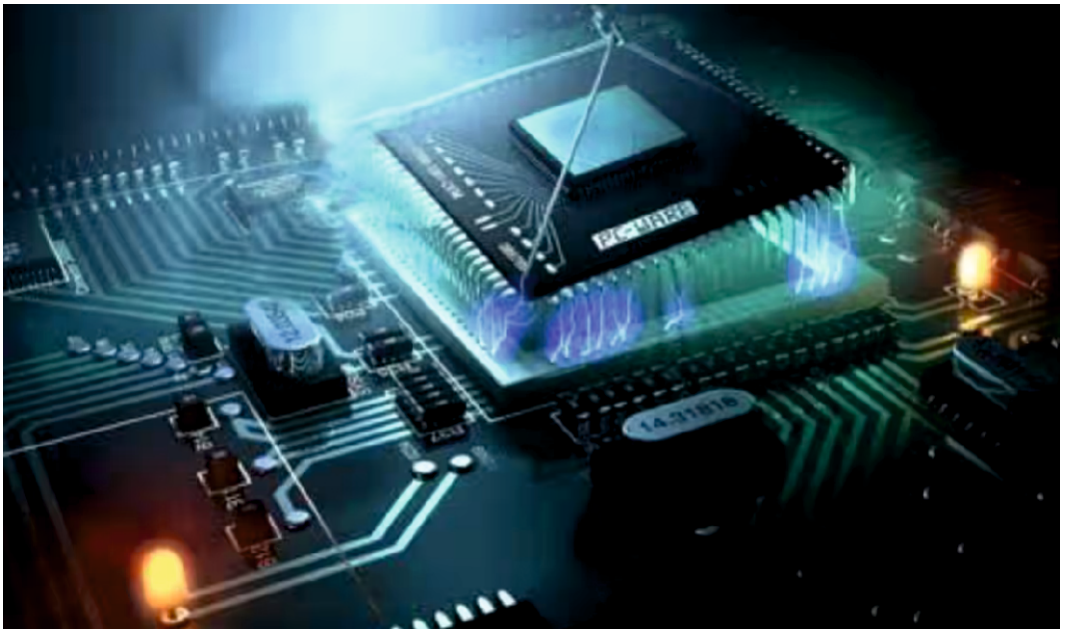
ARCHIVES OF VLSI TECHNOLOGY



Dr. Ashutosh Anand
Dr. Sachin Gupta

ALEXIS PRESS
JERSEY CITY, USA

ARCHIVES OF VLSI TECHNOLOGY

ARCHIVES OF VLSI TECHNOLOGY

Dr. Ashutosh Anand

Dr. Sachin Gupta

ALEXIS PRESS

Published by: Alexis Press, LLC, Jersey City, USA
www.alexispress.us

First Published 2022

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication Data

Includes bibliographical references and index.

Archives of VLSI Technology by *Dr. Ashutosh Anand, Dr. Sachin Gupta*

ISBN 978-1-64532-877-3

CONTENTS

CHAPTER 1

INTRODUCTION TO VLSI TECHNOLOGY

Dr. Ashutosh Anand, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id- ashutoshanand@presidencyuniversity.in

ABSTRACT:

Very Large Scale Integration (VLSI) technology is a field of microelectronics that deals with the fabrication of integrated circuits (ICs) by combining thousands or even millions of transistors and other components on a single chip. The miniaturization of electronic devices has led to the development of VLSI technology, which allows for the creation of complex and powerful ICs. VLSI technology has revolutionized the electronics industry by enabling the production of smaller, faster, and more powerful devices. The development of VLSI technology has made it possible to integrate more functionality into a single chip, leading to a wide range of applications such as microprocessors, memory chips, digital signal processors, and application-specific integrated circuits (ASICs).

KEYWORDS:

ASIC, Integrated Circuit, Electronic Circuit, memory chips, VLSI.

INTRODUCTION

A CPU, RAM, ROM, and other components are often included in electrical circuits on a single PCBA. However, an IC designer has the option to combine all of these onto a single chip thanks to very large-scale integration (VLSI) technology. We can find evidence of the electronics industry's explosive expansion if we look over the previous several decades. Enhanced functionality, enhanced miniaturisation, and higher performance are all advantages. There is less room for mistake as a result of the need to insert more components while using less space.With that in mind, it makes sense why silicon (CMOS) technology has emerged as the preferred method of manufacture for VLSI circuits that are reasonably priced and of great performance during the last several years.

The technique of embedding or integrating tens of millions of semiconductors onto a single silicon semiconductor microchip is known as very large-scale integration. VLSI technology was first developed in the late 1970s, at the same time when high level processor (computer) microchips have been beginning to take shape. The microcontroller and the microprocessor are two of the most popular VLSI components.VLSI is an acronym for a kind of integrated circuit that has several devices on a computer device [1] . Naturally, the phrase dates back to the 1970s, along with a variety of other grade assimilation categorization depending on the quantity of gated or semiconductors in an integrated circuit (IC)

The development of large-scale integration technologies is principally responsible for the electronics industry's impressive expansion. The capabilities for ICs in controller, communication, slightly elevated computing, and electrical goods however are expanding thanks to the introduction of VLSI designs. Due to VLSI technology, technological advances like telephones and cellphone telecommunications provide never-before-seen levels of mobility, processing power, and application access. Forecasts for this trend point to a fast rise

as demand keeps rising.The main benefits of VLSI technology are as follows circuit size reduction increased device cost-effectiveness an increase in the engine speed of circuits and better performance less power is used than separate components increased device dependability less room is needed, and downsizing is encouraged.

The VLSI IC Design Process

VLSI IC design generally consists of two main stages or components:

a) Digital design utilising an arduino ide software, such as Verilog, System Computer architecture, and VHDL, is referred to as front-end design. Additionally, this step includes design assurance via emulation and other types of verification. Designing, which begins with the valves and continuing thorough design with testing process, is also a part of the complete process[2], [3].

b) Qualification and CMOS database design allows up the back-end design. Formal definition plus failure simulation are also included.

c) The whole front architecture steps are as follows, and the complete design process is step-by-step:

d) Problem Description: This is a system's high-level interpretation. We discuss the important factors, including design methods, usability, performance, production methods, and physical dimensions. The horsepower, versatility, performance, and volume of the VLSI technology are all included in the final specs.

e) Architecture Definition: This covers basic requirements like floating-point units, the choice of system to employ, such as RISC or CISC, and the size of the ALU cache.

f) Functional Design: This identifies the key functional components of a system, allowing for the determination of each component's physical and electrical requirements as well as connectivity needs.

g) Control flow, Boolean operations, word format, and bank allocation are all part of the logic design process.

Circuit Design: In this phase, the realization of the circuit as a netlist is carried out. Even though this is a programming process, the output is checked using simulation. Tangible Model: In this phase, the layout is created by transforming the circuit into a geometry representation. Additionally, this stage adheres to certain predetermined static criteria, such as the omega rules, which provide accurate information on the ratio, component spacing, and size.

VLSI architecture design is concerned with deciding on the necessary hardware resources for solving problems from data and/or signal processing and with organizing their interplay in such a way as to meet target specifications defined by marketing. The foremost concern is to get the desired functionality right. The second priority is to meet some given performance target, often expressed in terms of data throughput or operation rate [4] .

A third objective, of economic nature this time, is to minimize production costs. Assuming a given fabrication process, this implies minimizing circuit size and maximizing fabrication yield so as to obtain as many functioning parts per processed wafer as possible. Another general concern in VLSI design is energy efficiency. Battery-operated equipment, such as hand-held cellular phones, laptop computers, digital hearing aids, etc.

DISCUSSION

Obviously imposes stringent limits on the acceptable power consumption. It is perhaps less evident that energy efficiency is also of interest when power gets supplied from the mains. The reason for this is the cost of removing the heat generated by high-performance high-density ICs. While the VLSI designer is challenged to meet a given performance figure at minimum power in the former case, maximizing performance within a limited power budget is what is sought in the latter.

The ability to change from one mode of operation to another in very little time, and the flexibility to accommodate evolving needs and/or to upgrade to future standards are other highly desirable qualities and subsumed here under the term agility [5] . Last but not least, two distinct architectures are likely to differ in terms of the overall engineering effort required to work them out in full detail

Driven by dissimilar applications and priorities, hardware engineers have, over the years, devised a multitude of very diverse architectural concepts which we will try to put into perspective how their respective strengths can be combined into one architecture. After the necessary groundwork for architectural analysis has been laid in subsequent sections will then discuss. In Figure 1 illustrate the VLC design principles.

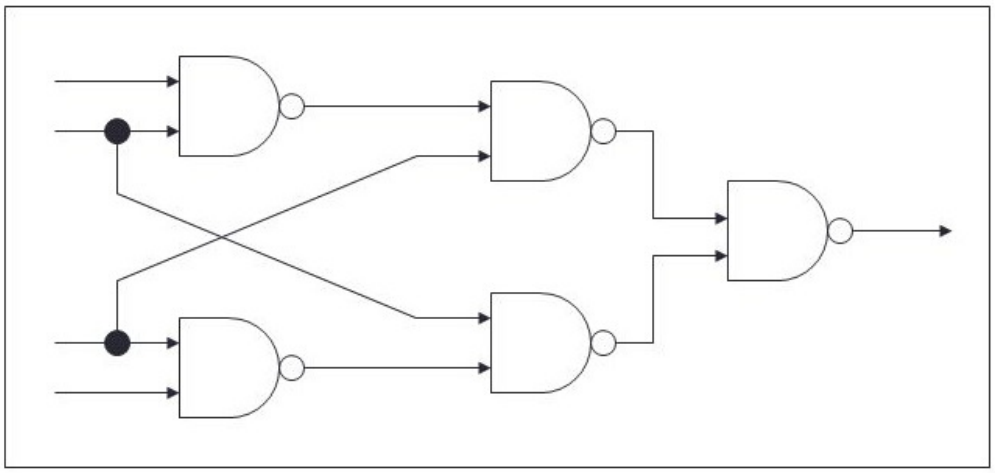


Figure 1: Illustrate the principle of VLC design.

To select, arrange, and improve the necessary hardware resources in an efficient way with a focus on dedicated architectures is concerned with organizing computations of combinational nature. Extends our analysis to no recursive sequential computations before time wise recursive computations are addressed other than word-level computations on real numbers. Inserted in between is section 2.5 that discusses the options available for temporarily storing data and their implications for architectural decisions. Given some computational task, one basically has the choice of writing program code and running it on a program-controlled machine, such as a microprocessor or a digital signal processor (DSP), or of coming up with a hardwired electronic circuit that carries out the necessary computation steps.This fundamental dichotomy, which is described implies that a systems engineer has to make a choice:

a) Select a processor-type general-purpose architecture and write program code for it, or
b) Tailor a dedicated hardware architecture for the specific computational needs.

Deciding between a general-purpose processor and an architecture dedicated to the application at hand is a major decision that has to be made before embarking on the design of a complex circuit.A great advantage of commercial microprocessors is that developers can focus on higher-level issues such as functionality and system-level architecture right away. There is no need for them to address all those exacting chores that burden semi- and even more so full-custom design.

There is no need for custom fabrication masks opting for commercial instruction-set processors and/or FPL sidesteps many technical issues that absorb much attention when a custom IC is to be designed instead. Conversely, it is precisely the focus on the payload computations, and the absence of programming and configuration overhead together with the full control over every aspect of architecture, circuit, and layout design that make it possible to optimize performance and energy efficiency.

Such as power distribution, clock preparation and distribution, input/output design, physical design and verification, signal integrity, electrical overstress protection, wafer testing, and package selection, all to be discussed in forthcoming chapters. Setting up a working CAE/CAD design flow typically also is a major stumbling block, to say nothing of estimating sales volume, hitting a narrow window of opportunity, finding the right partners, and providing the necessary resources, in-house expertise, and investments.

Also note that field-programmable logic (FPL) frees developers from dealing with many of these issues too. General purpose Special purpose Algorithm any, not known a priori fixed, must be known Architecture instruction set processor, dedicated design, Harvard style no single established pattern Execution model fetch–load–execute–store cycle process data item and pass on. Datapath universal operations, specific operations only, ALU(s) plus memory customized design Controller with program microcode typically hardwired Performance instructions per second, data throughput, indicator run time of various can be anticipated benchmark programs analytically Paradigm from craftsman in his machine shop division of labor in a factory manufacturing working according to set up for smooth production different plans every day of a few closely related goods

Possible hardware standard µC|DSP components ASIC of dedicated architecture implementations or ASIC with on-chip µC|DSP or FPL (FPGA|CPLD) Engineering effort mostly software design mostly hardware design Strengths highly flexible, room for max performance, immediately available, highly energy-efficient, routine design flow, lean circuitry. Upon closer inspection, one finds that dedicated architectures fare much better in terms of performance and/or dissipated energy than even the best commercially available general-purpose processors in some situations, whereas they prove a dreadful waste of both hardware and engineering resources in others.

Algorithms that are very irregular, highly data-dependent, and memory-hungry are unsuitable for dedicated architectures. Situations of this kind are found in electronic data processing such as databank applications, accounting, and reactive systems like industrial control, user interfaces. A system is said to be reactive if it interacts continuously with an environment, at a speed imposed by that environment. The system deals with events and the mathematical formalisms for describing them aim at capturing the complex ordering and causality relations between events that may occur at the inputs and the corresponding reactions events themselves at the outputs. Examples: elevators, protocol handlers, anti-lock brakes, process controllers, graphical user interfaces, operating systems.

As opposed to this, a transformation system accepts new input values often at regular intervals uses them to compute output values, and then rests until the subsequent data items

arrive. The system is essentially concerned with arithmetic/logic processing of data values. Form alism s for describing transform systems capture the numerical dependencies between the various data items involved. Examples: filtering, data compression, ciphering, pattern recognition, and other applications colloquially referred to as number crunching but also compilers and payroll programs.

Program-controlled general-purpose processor (a) and dedicated (special-purpose) hardware structure as architectural antipodes and others. In search of optimal architectures for such applications, one will invariably arrive at hardware structures patterned after instruction set processors. Writing code for a standard microcomputer either bought as a physical part or incorporated into an ASIC as a megacell or as a virtual component is more efficient and more economic in this case.

Situations where data streams are to be processed in fairly regular ways offer far more room for coming up with dedicated architectures. Impressive gains in performance and energy efficiency over solutions based on general-purpose parts can then be obtained. Generally speaking, situations that favor dedicated architectures are often found in real-time applications from digital signal processing and telecommunications such as

DSPs are at their best for sustained multiply–accumulate operations and offer word widths of 32 bit or so. However, as the Viterbi algorithm can be arranged to make no use of multiplication and to make do with word widths of 6 bit or less, DSPs cannot take advantage of these resources. A pipeline of tailor-made stages optimized for branch metric computation, path metric update, and survivor path trace back operations, in contrast, makes it possible to exploit the parallelism inherent in the Viterbi algorithm. In Figure 2 illustrate the design of VLCC.

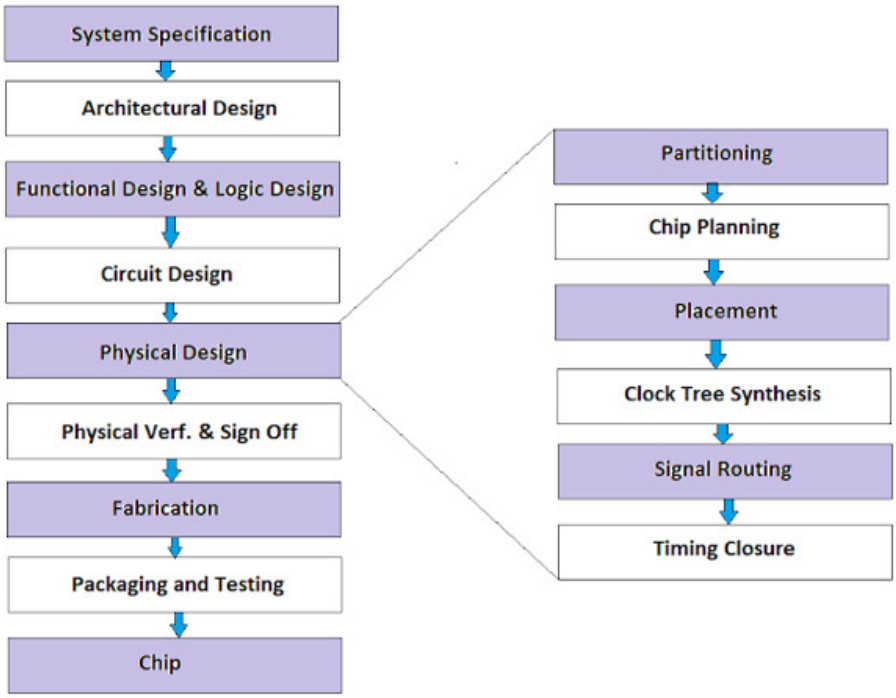


Figure 2: Illustratethe VLCC design Flow.

Diverse throughput requirements can be accommodated by trading the number of computational units in each stage for throughput. Sophisticated DSPs, such as the C6455, include an extra coprocessor to accelerate path metric update and survivor trace back. The

process of creating a VLSI chip involves several steps, including design, fabrication, testing, and packaging. The design phase involves creating a circuit layout using computer-aided design (CAD) software, which is then used to manufacture the chip using photolithography and other fabrication techniques. The fabricated chip is then tested to ensure its functionality and performance, and finally packaged for use in electronic devices.

The development of VLSI technology has driven the growth of the electronics industry and has had a significant impact on many aspects of modern life. The ongoing advancement of VLSI technology continues to bring new and exciting innovations to the field of electronics, including the development of faster and more powerful computers, the Internet of Things, and smart devices. There are many different duties that electronic devices today carry out in everyday life.

Certain mechanical, hydraulic, or other mechanisms have been replaced by electronic systems because they are often smaller, more adaptable, and simpler to maintain. In certain instances new uses have been made possible by electronic systems. Electronic systems carry out a range of activities, some of which are more obvious than others.

a) Portable MP3 players and DVD players are examples of personal entertainment devices that use very little energy to execute complex algorithms.
b) Electronic systems in automobiles regulate fuel injection systems, adjust suspensions for uneven terrain, and carry out the control tasks necessary for anti-lock braking (ABS) systems. They also run sound systems and displays.
c) Digital electronics compress and decompress video, even at highdefinition data rates, on-the-fly in consumer gadgets.
d) Despite their specific purpose, low-cost terminals for Web surfing nonetheless need complex circuitry.
e) Workstations and personal computers provide text processing, financial analysis, and gaming. Central processing units (CPUs) and specialised hardware for speedier screen display, disc access, etc. are both found in computers. IP-Based Design.
f) Medical electronic devices assess body activities and perform complex processing algorithms to alert about unexpected circumstances.

Far from overwhelming customers, the availability of these sophisticated systems merely increases demand for more complex systems. The difficulty of designing and producing integrated circuits and electronic systems is continuously rising as a result of the increasing sophistication of applications. The diversity of this group of systems is arguably its most astounding feature; as systems become more complicated, we develop fewer general-purpose computers and an increasing number of systems with specialised functions. While the increased needs of customers continue to push the boundaries of design and manufacturing, our capacity to do so is a credit to our developing mastery of both integrated circuit production and design. The qualities of integrated circuits what we can and cannot put in an integrated circuit efficiently largely dictate the design of the overall system, even though we shall focus on integrated circuits in this book. Integrated circuits significantly enhance several key system properties. Compared to digital circuits made from discrete components, ICs offer three major advantages:

In contrast to the millimetre or centimetre scales of discrete components, transistors and wires in integrated circuits are substantially smaller. As smaller components have less parasitic resistances, capacitances, and inductances, they are faster and use less power. Compared to switching between chips, switching signals between logic 0 and logic 1 inside a semiconductor can be done far more quickly. As compared to communication between chips

on a printed circuit board, chip-to-chip communication may happen hundreds of times quicker. Due to fewer components and connections having lower parasitic capacitances to slow down the signal, on-chip circuits operate at fast speeds.

Moreover, a chip uses substantially less power during logic processes. Again, smaller circuits on the chip result in reduced power consumption because parasitic capacitances and resistances are driven by less power owing to their smaller size.

System and VLSI At the system level, these advantages of integrated circuits translate into benefits: Physically smaller. Consider handheld mobile phones or portable TVs, where size is often a benefit. A single chip may minimise overall power usage by taking the place of many common components. A smaller, less expensive power supply may be utilised; as less heat is generated, a fan may no longer be required; and a simpler cabinet with less electromagnetic shielding may also be conceivable. Lowering power usage has a knock-on impact on the remainder of the system. A lower price. Reduced component counts, power supply needs, cabinet prices, and other factors all result in lower system costs. Although while bespoke ICs individually cost more than the conventional components they replace, the cost of a system made of them might be lower due to the cascading effects of integration.

Understanding IC manufacturing technology as well as the economics of ICs and digital systems is necessary to comprehend why integrated circuit technology has such a significant impact on the design of digital systems. The foundation of integrated circuit technology is in our capacity to produce enormous quantities of very tiny components; now, California produces more transistors annually than the state receives in precipitation. This section provides a quick overview of VLSI production. Technology the majority of industrial processes have a close relationship with the product they are producing. For instance, a Buick assembly line would need a modest rearrangement to establish a Chevrolet assembly line; equipment like sheet metal molds would need to be changed, and certain machines could even require modification. Moreover, neither assembly line would come close to being able to create electric drills[6], [7].

On the other side, integrated circuit fabrication technology is incredibly adaptable. CMOS, bipolar, and other circuit types have various manufacturing procedures, yet any of those circuit types may be produced on a production line by just switching out a few common tools called masks. One CMOS manufacturing facility, for instance. By altering the masks that create the patterns of wires and transistors on the chips, Digital Systems and VLSI are able to produce both microprocessors and microwave oven controllers. The base material for making integrated circuits is silicon wafers. Wires and transistors are created on the wafer using patterns created during the manufacturing process. A wafer is patterned with a succession of identical chips, with some room left over for test circuit designs that enable manufacturers to gauge the effectiveness of the manufacturing process. Since we can process a single wafer to create several identical chips, the IC manufacturing method is effective. We control the digital circuit that will be produced by altering the masks that dictate what patterns are imprinted on the chip. The IC fabrication line is a versatile manufacturing platform that can be swiftly retooled to produce large numbers of a different kind of chip using the same production techniques as the line's prior output.

With the help of common components, we could construct a breadboard circuit just from this description. We need to take it a step further and create the layout, or patterns on the masks, in order to produce it on an IC manufacturing line. Transistors and wires that adhere to the circuit in the schematic are created by the rectangular forms in the layout, which is shown here as a stick diagram. Layout creation takes a lot of time, but it is crucial since the size of

the layout affects how much it will cost to construct a circuit as well as how quickly it will operate. In the production process, the layout patterns are transferred from the masks to the wafer using a photolithographic photographic printing technique. To modify the wafer just partially, the patterns the mask leaves behind are used: Impurities are introduced to the wafer at certain areas, insulating and A A' p-type transistor A A.

IP-Based Design, Fourth Edition, Modern VLSI Design Going Back to the Table of Contents On top of the wafer, conducting materials for digital systems and VLSI are also added. High temperatures, limited quantities of very poisonous chemicals, and ultra-clean surroundings are needed for these fabrication procedures. The wafer is split into a number of chips at the conclusion of processing. Manufacturing faults some of the chips on the wafer may not function since no manufacturing process is faultless. The smallest chip that can currently be reliably made is 1.5 to 2 cm on a side, whereas a wafer is in the range of 30 to 45 cm.

Because at least one flaw is nearly certain to occur on each wafer, wafers are chopped into smaller, functional chips. When the wafer is divided into chips, each chip is separately examined; those that pass the test are kept. The familiar to digital designer's packages include the functioning chips. Although the package body shields the chip from handling and the environment in some packages, in others, solder bumps directly link the chip to the packaging. In certain packages, small wires connect the chip to the package's pins [8].

A few different types of transistors and wires can be used to create all circuits, making integrated circuit manufacturing a potent technology. Additionally, any combination of transistors and wires can be built on a single fabrication line by simply changing the masks that determine the pattern of the components on the chip. Integrated circuits operate incredibly quickly because the circuits are relatively tiny. Moreover, we are not limited to creating just a few common chip kinds; we are free to create any function. We can create more complicated, quicker digital systems with the freedom provided by IC fabrication.A lot of work has been put into improving integrated circuit manufacturing because it has so much potential a large number of components can be produced using only a few common production techniques. The cost of developing a chip, however, rises as processors get more complicated and has a significant role in the final cost of the chip.

Moore's Law Gordon Moore anticipated that the number of transistors that could be produced on a chip would increase exponentially in the 1960s. His forecast, which is now referred to as Moore's Law, proved extremely accurate. The most important forecast made by Moore was that the number of transistors will double every two years. The International Technological Roadmap for Semiconductors (ITRS), which outlines methods to sustain Moore's Law's speed, is now maintained by an industry body. Charting the dates of the launch of significant goods that advanced the state of the manufacturing art demonstrates improvements in manufacturing capabilities. The black dots represent random-access memory, usually dynamic RAMs or DRAMs, while the squares represent different logic circuits, principally CPUs and digital signal processors (DSPs). Logic chips and memory chips both follow Moore's Law, with memory chips having more transistors per unit area at any given moment [9], [10] .

Terminology The minimum channel length of a transistor is the most fundamental manufacturing process parameter. A technological node is a manufacturing innovation at a certain channel length. Micron, submicron, deep submicron, and now nanometer technologies are members of a family of technologies with comparable feature sizes that we often discuss. For technologies smaller than 100 nm, the phrase "nanometer technology" is often used.

CONCLUSION

VLSI (Very Large Scale Integration) technology has revolutionized the field of electronics and computing by allowing for the production of complex integrated circuits with millions or even billions of transistors on a single chip. VLSI technology has enabled the creation of smaller, faster, and more power-efficient devices that have transformed the way we live and work. The development of VLSI technology has been driven by continuous advances in semiconductor manufacturing processes, which have allowed for the fabrication of increasingly dense and complex integrated circuits. This has been accompanied by advances in design tools and techniques, which have enabled the creation of highly optimized and efficient designs.

REFERENCES

[1] S. Tanaka and W. Kosonocky, "Welcome to the 2020 Symposia on VLSI Technology and Circuits," *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*. 2020. doi: 10.1109/VLSICircuits18222.2020.9162814.

[2] "2018 International Symposium on VLSI Technology, Systems and Application, VLSI-TSA 2018," *2018 International Symposium on VLSI Technology, Systems and Application, VLSI-TSA 2018*. 2018.

[3] M. LI and R. HUANG, "Device and integration technologies for VLSI in post-Moore era," *Sci. Sin. Informationis*, 2018, doi: 10.1360/n112018-00114.

[4] R. D. Clark, "Emerging applications for high K materials in VLSI technology," *Materials*. 2014. doi: 10.3390/ma7042913.

[5] D. L. Critchlow, "MOSFET Scaling-The Driver of VLSI Technology," *Proc. IEEE*, 1999, doi: 10.1109/JPROC.1999.752521.

[6] S. Nicolay A., "Features of VLSI Components in Fully-depleted SOI CMOS Technology," *Nanoindustry Russ.*, 2018, doi: 10.22184/1993-8578.2018.82.46.48.

[7] M. C. Martínez-Rodríguez, M. A. Prada-Delgado, P. Brox, and I. Baturone, "VLSI design of trusted virtual sensors," *Sensors (Switzerland)*, 2018, doi: 10.3390/s18020347.

[8] "2019 Symposium on VLSI Technology, VLSI Technology 2019 - Digest of Technical Papers," *Digest of Technical Papers - Symposium on VLSI Technology*. 2019.

[9] C. V. S. Chaitanya, C. Sundaresan, P. R. Venkateswaran, K. Prasad, and V. Siva Ramakrishna, "Design of high-speed multiplier architecture based on vedic mathematics," *Int. J. Eng. Technol.*, 2018, doi: 10.14419/ijet.v7i2.4.11228.

[10] A. B. Kahng, L. Wang, and B. Xu, "TritonRoute: An initial detailed router for advanced VLSI technologies," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, 2018. doi: 10.1145/3240765.3240766.

CHAPTER 2

HARDWARE DEVELOPMENT OF VLSI

Dr. K Bhanu Rekha, Associate Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id- bhanurekha@presidencyuniversity.in

ABSTRACT:

Hardware development of VLSI (Very Large Scale Integration) refers to the process of designing and fabricating integrated circuits that contain a large number of transistors and other electronic components on a single chip. VLSI technology has revolutionized the electronics industry by enabling the development of faster, smaller, and more complex electronic systems.

KEYWORDS:

Diamond Crystal, Electronic Components, Transistors, Hardware Development, Integrated Circuits.

INTRODUCTION

Pure silicon that has been melted at 1400 degrees Celsius is used in the wafer processing process. Thereafter, a small seed with the deemed essential crystal orientation is inserted into the liquid silicon and gradually removed at a pace of 1mm per minute. A cylindrical ingot is used to make the diamond crystal, which is subsequently separated into smaller ingots or platters before being polished and orientated.

a) Photolithography: Both photographic and photo etching masks are used throughout this process. We next apply a photoresist layer on the wafer. A photo aligner is then used to align the wafer to a mask. The footprints that have made it through the mask are then shown when the wafer has been subjected to ultraviolet light.
b) Etching: In this procedure, we degrade the surface of the wafer to produce patterns. We use more plasma liquid chemicals to remove the remaining photoresist while utilising an etching mask to safeguard the material's crucial regions [1].

Using a process called ion implantation, we introduce dopants into the semiconductor in order to produce the required electrical characteristic. The technique uses a beam of more dopant ions to precisely target certain areas of the wafer. The energy level of the beam determines how deeply the wafer is cut. Metallization: At this stage, a thin layer of aluminium is applied to the whole wafer.

Many hundred chips are included on each wafer for assembly and packaging. As a consequence, we use a diamond saw to cut the wafers into individual chips. These are subsequently put through electrical testing, and the unsuccessful ones are thrown away. Those that succeed, however, get a careful visual inspection under a microscope. The chosen chips that pass the visual inspection are next packed and scrutinised one more.

VLSI technology is ideally adapted to meet the needs of modern electronic devices and systems. Because to the growing need for functionality, reliability, performance, portability, and shrinking, VLSI innovation will continue to drive electronics progress [2]–[4].

For safety-critical real-time embedded systems (such satellite and surveillance systems), where dependability is equally as important as energy efficiency, reliability conscious energy management is necessary. Some early work has been done for the investigation of general system dependability for SoC. (system-on-a-chip). As part of the architecture level design of microprocessors, RAMP is the first instrument for modelling long-term processor reliability. In their subsequent published work, dynamic voltage, the authors proposed frequency scaling and the dynamic reliability management (DRM) paradigm. From a dependability perspective, these works of art showed that it wasn't sufficient to just manage the temperature or power.

The goal of many contemporary embedded system initiatives is to use less energy while maintaining the deadlines of all real-time job models. Power management techniques that leverage static or dynamic slack are currently being studied [5].

The reliability of a CPU will naturally rise with decreased power usage for the larger computer. Nevertheless, there is currently no single knob that can control the two objectives of increasing longevity and reducing power. Recent ideas for management jobs using low power techniques like DVFS include power with some dependability awareness. The bulk of presently available publications, however, focus on momentary flaws rather than wear-out failures over time. Recently, a method of work allocation and scheduling that takes dependability into account was proposed for non-linear, non-embedded processors [6], [7].

A semiconductor is placed on a Board using VLSI technology. It would not have been possible to make this Board using semiconductors without VLSI technology. Using state-of-the-art PCB To effectively perform the work while designing for the little margin of error available in VLSI technology, finite element analysis software is required. Allegro by Cadence is one such application that provides all the capabilities and analytical tools needed for both simple and intricate circuit designs [8], [9].

Comparison of architectural alternatives for a secret-key block encryption/decryption algorithm (AES cipher, block size 128 bit, key length 128 bit). The Rijndael algorithm makes extensive use of a so-called S-Box function and its inverse; the three hardware implementations include multiple look-up tables (LUTs) for implementing that function. Also, (de)ciphering and subkey preparation are carried out concurrently by separate hardware units. On that background, the throughput of the assembly language program running on a Pentium III is indeed impressive. This largely is because the Rijndael algorithm has been designed with the Pentium architecture in mind (MMX instructions, LUTs that fit into cache memory, etc.). Power dissipation remains daunting. Architecture General Purpose Special purpose Key component RISC Proc. CISC Proc.

DISCUSSION

Algorithms and hardware architectures are intimately related. While dedicated architectures outperform program-controlled processors by orders of magnitude in many applications of predominantly transformatorial nature, they cannot rival the agility and economy ofprocessor-type designs in others of more reactive nature. More precise criteria for finding out whether a dedicated architecture can be an option or not from a purely technical point of view puts various applications fromsignal and data processing into perspective.

Costs in hardware are not the same as those in software. As an example, permutations of bits within a data word are time-consuming operations in software as they must be carried out sequentially. In hardware, they reduce to simple wires that cross while running from one subcircuit to the next[10]. Look-up tables (LUTs) of almost arbitrary size, on the other hand, have become an abundant and cheap resource in any microcomputer while large on-chip

RAMs and ROMs tend to eat substantial proportions of the timing and area budgets of ASIC designs.

In an attempt to provide some guidance, we have collected ten criteria that an information processing algorithm should ideally meet in order to justify the design of a special-purpose VLSI architecture and to take full advantage of the technology. Of course, very few real-world algorithms satisfy all of the requirements listed. It is nevertheless safe to say that designing a dedicated architecture capable of outperforming a general-purpose processor on the grounds of performance and costs will prove difficult when too many of these criteria are violated. The list below begins with the most desirable characteristics and then follows their relative significance.

1. Loose coupling between major processing tasks. The overall data processing lends itself to being decomposed into tasks that interact in a simple and unmutable way. Whether those tasks are to be carried out consecutively or concurrently is of secondary importance at this point; what counts is to come up with a well-defined functional specification for each task and with manageable interaction between them. Architecture design, functional verification, optimization, and reuse otherwise become real nightmares.
2. Simple control flow. The computation's control flow is simple. This key property can be tracked down to two more basic considerations**:** Figure 1 illustrate the specialization and model in VLSI.

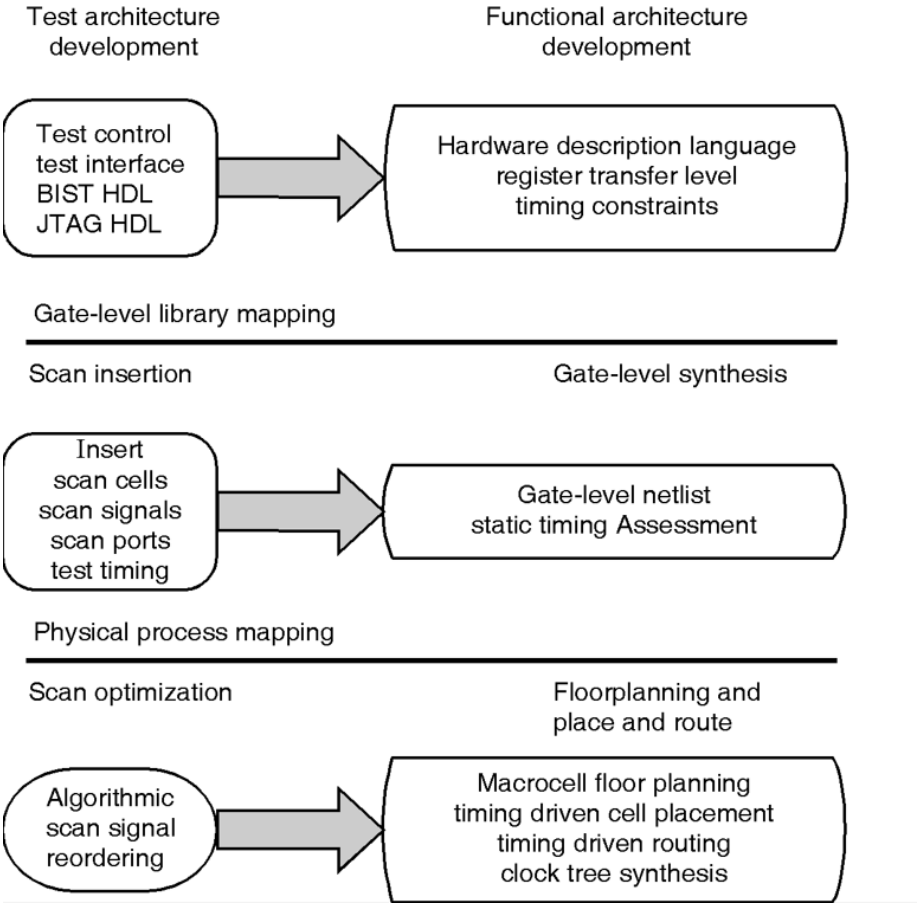


Figure 1: illustrate the specialization and model in VLSI.

Exact meaning of operation and data item left unspecified; 16 bit-by-16 bit multiply–accumulate (MAC) operations on 16 bit samples are often considered typical in a context of digital signal processing).

a) The course of operation does not depend too much on the data being processed; for each loop the number of iterations is a priori known and constant.
b) The application does not ask for computations to be carried out with overly many varieties, modes of operations, data formats, distinct parameter settings, and the like.

The benefit of a simple control flow is twofold. For one thing, it is possible to anticipate the datapath resources required to meet a given performance goal and to design the chip's architecture accordingly. There is no need for statistical methods in estimating the computational burden or in sizing data memories and the like. For another thing, data path control can be put in different terms, the target algorithm is virtually free of branching and loops such as if...then [...else], while...do, and repeat...until that include data items in their condition clauses.

Architectures of VLSI Circuits

Handled by counters and by simple finite state machines (FSMs) that are small, fast, energyefficient and most important easy to verify. An overly complicated course of operations, on the other hand, that involves much datadependent branching, multitasking, and the like, favors a processor-type architecture that operates under control of stored microcode. Most control operations will then translate into a sequence of machine instructions that take several clock cycles to execute.

The flow of data is regular and their processing is based on a recurrence of a fairly small number of identical operations; there are no computationally expensive operations that are called only occasionally. Regularity opens a door for sharing hardware resources in an efficient way by applying techniques such as iterative decomposition and timesharing. Conversely, multiple data streams that are to be processed in a uniform way lend themselves to concurrent processing by parallel functional units. A regular data flow further helps to reduce communications overhead in terms of both area and interconnect delay as the various functional units can be made to exchange data over fixed local links. Last but not least, regularity facilitates reuse and reduces design and verification effort.

As opposed to this, operations that are used infrequently either will have to be decomposed into a series of substeps to be executed one after the other on a general-purpose datapath, which is slow, or will necessitate dedicated functional units bound to sit idle for most of the time, which inflates chip size. Irregular data flow requires long and flexible communication busses which are at the expense of layout density, operating speed, and energy efficiency.

Reasonable storage requirements. Overall storage requirements are modest and have a fixed upper bound.7 Memories that occupy an inordinate amount of chip area, say more than half or so, cannot be incorporated into ASICs in an economic way and must, therefore, be implemented off-chip from standard parts, see subsection. Massive storage requirements in conjunction with moderate computational burdens tend to place dedicated architectures at a disadvantage. Compatible with finite precision arithmetics. The algorithm is insensitive to effects from finite precision arithmetics. That is, there is no need for floating-point arithmetics; fairly small word widths of, say, 16 bit or less suffice for the individual computation steps. Standard microprocessors and DSPs come with datapaths of fixed and often generous width (24, 32, 64 bit, or even floating-point) at a given price. No extra costs arise unless the programmer has to resort to multiple precision arithmetics.

As opposed to this, ASICs and FPL offer an opportunity to tune the word widths of datapaths and on-chip memories to the local needs of computation. This is important because circuit size, logic delay, interconnect length, parasitic capacitances, and energy dissipation of addition, multiplication, and other operations all tend to grow with word width, combining into a burden that multiplies at an overproportional rate. Which precludes the use of dynamic data structures. 8 Processor datapaths tend to be fast and area effi cient because they are typically hand-optimized at the transistor level (e.g. dynamic logic) and implemented in tiled layout rather than built from standard cells. These are only rarely options for ASIC designers.

THE ARCHITECTURAL ANTIPODES

Nonrecursive linear time-invariant computation. The processing algorithm describes a nonrecursive linear time-invariant system over some algebraic field. Each of these properties opens a door to reorganizing the data processing in one way or another, see sections through for details. High throughputs, in particular, are much easier to obtain from nonrecursive computations as will become clear in section.

The algorithm does not make use of roots, logarithmic, exponential, or trigonometric functions, arbitrary coordinate conversions, translations between incompatible number systems, and other transcendental functions as these must either be stored in large look-up tables (LUT) or get calculated on-line in lengthy and often irregular computation sequences. Such functions can be implemented more economically provided that modest accuracy requirements allow approximation by way of lookups from tables ofreasonable size, possibly followed by interpolation.

Extensive usage of data operations unavailable from standard instruction sets. Of course, there exist many processing algorithms that cannot do without costly arithmetic- logic operations. It is often possible to outperform traditional program-controlled processors in cases where such operations need to be assembled from multiple instructions. Dedicated datapaths can then be designed to do the same computation in a more efficient way. Examples include complex-valued arithmetics, add compare select operations, and many ciphering operations. It also helps when part of the arguments are constants because this makes it possible to apply some form of preprocessing. Multiplication by a variable is more onerous than by a constant, for instance. Throughput rather than latency is what matters. This is a crucial prerequisite for pipelined processing.

No divisions and multiplications on very wide data words. Multiplications involving wide arguments are not being used the algorithm does not make extensive use of multiplications and even less so of divisions as their VLSI implementation is much more expensive than that of addition/subtraction when the data words involved are wide. There is plenty of land between the architectural antipodes most markets ask for performance, agility, low power, and a modest design effort at the same time. In the face of such contradictory requirements, it is highly desirable to combine the throughput and the Recursiveness. Linear is meant to imply the principle of superposition $f(x(t) + y(t)) \equiv f(x(t)) + f(y(t))$ and $f(c\,x(t)) \equiv cf(x(t))$. T im e-invariant means that the sole effect of delaying the input is a delay of the output by the same amount of time: if $z(t) = f(x(t))$ is the response to $x(t)$ then $z(t - T)$ is the response to $x(t - T)$. Fields and other algebraic structures are compared. Dropping unit factors and/or zero sum terms (both at word and bit levels), substituting integer powers of 2 as arguments in multiplications and divisions, omitting insignificant contributions, special number representation schemes, taking advantage of symmetries, precomputed look-up tables, and distributed arithmetic, see subsection are just a few popular measures that may help to lower the computational burden in situations where parts of the arguments are known ahead of time.

Architectures of VLSI Circuits

Energy efficiency of a dedicated VLSI architecture for demanding but highly repetitive computations with the convenience and flexibility of an instruction set processor for more control-oriented tasks. This is because those parts of a system that ask for maximum computation rate are not normally those that are subject to change very often, and vice versa. The finding immediately suggests a setup where a software-controlled microcomputer cooperates with one or more dedicated hardware units. Separating the quest for computational efficiency from that for agility makes it possible to fully dedicate the various functional units to their respective tasks and to optimize them accordingly. Numerous configurations are possible and the role of the instruction set microcomputer varies accordingly.

Some digital systems and the computing requirements of major subfunctions thereof. Subfunctions primarily characterized by irregular control flow and/or repetitive control flow and Application need for flexibility need for comput. EfficiencyDVD player user interface, track seeking, 16-to-8 bit demodulation, tray and spindle control, error correction, processing of non-video data MPEG-2 decompression (directory, title, author, (discrete cosine transform), subtitles, region codes) video signal processing Cellular phone user interface, SMS, intermediate frequency directory management, filtering, (de)modulation, battery monitoring, channel (de)coding, communication protocol, error correction (de)coding, channel allocation, (de)ciphering, roaming, accounting speech (de)compression. Pattern recognition pattern classification, image stabilization, (e.g. as part of a object tracking, redundancy reduction,defensive missile) target acquisition, image segmentation,triggering of actions feature extraction[11].

CONCLUSION

Hardware development in VLSI technology has been critical to the creation of highly integrated and efficient electronic systems. The continued development of VLSI technology will play a significant role in the advancement of modern society and the growth of many industries. The hardware development in VLSI technology has led to the creation of a wide range of electronic devices, from microprocessors and memory chips to sensors and communication devices. It has also enabled the development of advanced systems such as autonomous vehicles, artificial intelligence, and the internet of things.

REFERENCE

[1] L. Louis, “Working Principle of Arduino and Using it as a Tool for Study and Research,” *Int. J. Control. Autom. Commun. Syst.*, 2016, doi: 10.5121/ijcacs.2016.1203.

[2] K. M. Cham, S. Y. Oh, and J. L. Moll, “Computer-Aided Design in VLSI Device Development,” *IEEE J. Solid-State Circuits*, 1985, doi: 10.1109/JSSC.1985.1052335.

[3] L. Louis, “Working Principle of a Arduino and Using It,” *Int. J. Control. Autom. Commun. Syst.*, 2016.

[4] M. Rudan, *Physics of Semiconductor Devices*. 2015. doi: 10.1007/978-1-4939-1151-6.

[5] X. Chen, G. Liu, N. Xiong, Y. Su, and G. Chen, “A Survey of Swarm Intelligence Techniques in VLSI Routing Problems,” *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.2971574.

[6] J. Mostow and B. Balzer, “Application of a transformational software development methodology to VLSI design,” *J. Syst. Softw.*, 1984, doi: 10.1016/0164-1212(84)90021-9.

[7] S. T and J. R. M, “Survey on Power Optimization Techniques for Low PowerVLSI Circuitsin Deep Submicron Technology,” *Int. J. VLSI Des. Commun. Syst.*, 2018, doi: 10.5121/vlsic.2018.9101.

[8] M. C. Stamm, M. Wu, and K. J. R. Liu, “Information forensics: An overview of the first decade,” *IEEE Access*, 2013, doi: 10.1109/ACCESS.2013.2260814.

[9] M. A. Islam, P. K. Datta, and H. Myler, “VLSI structures for DNA sequencing—a survey,” *Bioengineering*. 2020. doi: 10.3390/bioengineering7020049.

[10] H. Chen, S. Saïghi, L. Buhry, and S. Renaud, “Real-time simulation of biologically realistic stochastic neurons in VLSI,” *IEEE Trans. Neural Networks*, 2010, doi: 10.1109/TNN.2010.2049028.

[11] T. Acharya and C. Chakrabarti, “A survey on lifting-based discrete wavelet transform architectures,” *J. VLSI Signal Process. Syst. Signal Image. Video Technol.*, 2006, doi: 10.1007/s11266-006-4191-3.

CHAPTER 3

ADVANCEMENTS IN PHYSICS-BASED EM MODELING: TECHNIQUES AND APPLICATIONS FOR CHARACTERIZING ELECTROMAGNETIC PHENOMENA

Dr. Manikandan M, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id- manikandan.m@presidencyuniversity.in

ABSTRACT:

Electromagnetic (EM) modeling is a fundamental component of many fields, including electrical engineering, physics, and materials science. In particular, EM modeling is used to understand and predict the behavior of electromagnetic waves and fields in different materials and structures. Physics-based EM modeling is a technique that uses fundamental physical principles to describe the behavior of EM waves and fields, and it is often used to design and optimize complex devices and systems.

KEYWORDS:

Devices, Electrical Engineering, Electromagnetic Phenomena, Physics, EM waves.

INTRODUCTION

We look at some of the most recent advancements in EM modelling and evaluation techniques. We explain specifically on the EM thermodynamics, present malfunction patterns, and their limitations. In the next section, we look at the key Korhonen's nonlinear evolution equation for void production and void growth in the constrained region as well as the popular stress-based modelling of EM failures of metal wires used for connections. The most recent compact two-phase EM proposal models are provided here. The shortcomings of these three various EM models are then shown. First off, the recommended nucleation time formula, which according to the experiment's results should be near to, does not adequately forecast the current exponent. We suggest the four distinct EM frameworks, which address the problem by include "void nucleation," to minimize this.

We provide a clearer explanation of the terms "void growth phase" and "void incubation phase" and show how to use compact models to estimate the nucleation time and determine how long each of the three phases of nucleation, incubation, and failure last. Moreover, variations in wire resistance patterns may be predicted by the three-phase electromagnetic model, which is more in accordance with the information obtained from experiments. Moreover, we show how this study may be used to multi-segment connections using the rapid saturation volume estimation approach for connecting, which is crucial for determining the EM failure time to failure.

Electro migration (EM) is the term used to describe the directed movement of electrically charged particles. Owing to their momentum, copper (Cu) atoms migrate in the direction of the electrical field that is placed between them and the conducting electrons. Atoms move down the route, either with conducting lattice atoms or impurity electrons, towards the anode terminal of something like the metal wire. This coordinated atomic mobility at the cathode and a corresponding metal accumulation at the metal's anode end wire result in a decrease in metal density. The so-called diffuser barriers are holding the metal wires within, causing a depletion and growth of the metal volume since atoms can't just escape.[1]

Often, metallic objects are used to insert or contain Cu damascene linking wire barriers like Ta and are then covered with either a metallic or a dielectric layer. As a result, the wire volume changes as a result of atom depletion. Stress-Based EM and Stress Diffusion Modeling In addition to the Black and Blech formulations, which are based on a combination of semi-physics and physics, novel connecting wire has been developed employing electro migration phenomenon structures. First up in this part are the most sophisticated Korhonen's equation-based stress evolution-based EM models.

Next, we will show that the newly developed physics-based Cu damascene observed values are better consistent with the EM model by connecting the wires to the new three-phase EM model. The first time that the end of current-carrying strips was discovered was by Blech. The following motion is the Nernst-Einstein velocity, which is the drift speed for flow, where Fem is the EM-induced force caused by the electronic wind is the chemical diffusion coefficient and is computed using the formula Da = D0 exp Ea, where D0 is the absorption coefficient the metal's flux.[2]

The electron wind-induced force (Fem) and the back-force brought on by changes in atom concentration (or stress gradients brought on by atoms being depleted somewhere at the cathode end and the atoms accumulating at the electrochemical end) are the two main forces acting on atoms during electromagnetic radiation (EM). Migration-induced buildup results in hydrostatic forces over the conductor, tightness at the anode, and torsion at the cathode end of the wire. The persistent, unidirectional stream flow both of these stresses, as well as the gradient of stresses along the metal line rise, actually act as a counterforce to reduce the metal corrosion caused by EM migration movement in some circumstances, even when a circuit is long or a significant signal is present.[3], [4] The basis of EM modeling is Maxwell's equations, which describe the behavior of electric and magnetic fields in the presence of charges and currents. These equations are fundamental to the understanding of EM waves and fields, and they provide a theoretical framework for predicting the behavior of EM phenomena in different materials and structures. Physics-based EM modeling involves using Maxwell's equations and other fundamental physical principles to develop models that accurately predict the behavior of EM waves and fields in a wide range of systems. There are several different methods used for physics-based EM modeling, including analytical techniques, numerical methods, and hybrid approaches that combine both analytical and numerical methods. In this article, we will review some of the most common methods used in physics-based EM modeling.

DISCUSSION

Analytical Techniques

Analytical techniques are used to develop closed-form solutions to Maxwell's equations for simple systems with known geometries and boundary conditions. These solutions can provide insights into the behavior of EM waves and fields in the system, and they can be used to develop simplified models that can be used for more complex systems [5] one of the most common analytical techniques used in EM modeling is the method of images. This technique involves using a mirrored image of a charged object to represent its effect on the EM field. The method of images can be used to solve a wide range of problems, including the calculation of the electric field around a charged plane, the capacitance of parallel plates, and the force between two charged particles. Another analytical technique used in EM modeling is the Green's function method. This technique involves using the Green's function of a system to find the solution to Maxwell's equations for an arbitrary source distribution. The Green's function is a mathematical function that describes the response of a system to a point

source, and it can be used to find the response to any source distribution using convolution. The Green's function method is commonly used in the study of scattering and diffraction phenomena, such as the behavior of electromagnetic waves in the presence of obstacles or discontinuities.

Numerical Methods

Numerical methods are used to develop solutions to Maxwell's equations for more complex systems that cannot be solved analytically. Numerical methods involve discretizing the system into a finite number of elements, such as cells or finite elements, and solving Maxwell's equations numerically for each element. This allows for the development of detailed and accurate models of the behavior of EM waves and fields in a wide range of systems. One of the most common numerical methods used in EM modeling is the finite-difference time-domain (FDTD) method. This method involves discretizing the system into a 3D grid of cells and solving Maxwell's equations for each cell at discrete time steps. The FDTD method is widely used in the study of complex systems, including the design of antennas, microwave circuits, and electromagnetic compatibility (EMC) issues. Another numerical method commonly used in EM modeling is the method of moments (MoM). The MoM involves discretizing the system into a set of surface or volume elements, and solving Maxwell's equations for each element using a set of integral equations. The MoM is commonly used in the study of antenna design, electromagnetic scattering, and other electromagnetic problems in which the geometry of the system is known[6], [7]. In three dedicated and one program-controlled processing units are arranged in a chain. Figure 1 illustrate the EM Algorithm starts with trajectory estimation based on the basic physics.

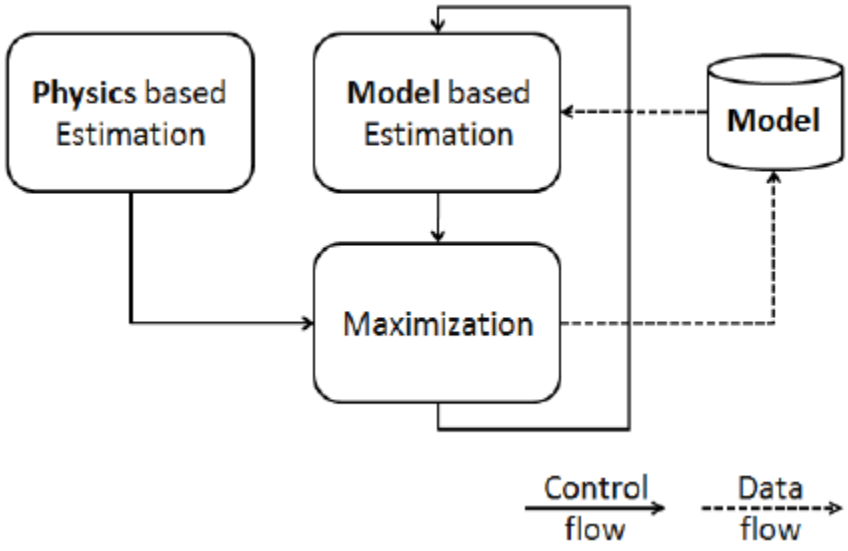


Figure 1: Illustrate the EM Algorithm starts with trajectory estimation based on the basic physics.

Each unit does its data processing job and passes the result to the downstream unit. While offering ample room for optimizing performance, this structure cannot accommodate much variation if everything is hardwired and tailor-made. Making the specialized hardware units support a limited degree of parametrization (e.g. wart data word width, filter order, code rate, data exchange protocol, and the like) renders the overall architecture more versatile while, at the same time, keeping the overhead in terms of circuit complexity and energy dissipation fairly low. The term weakly programmable satellites has been coined to reflect the idea. All

specialized hardware units now operate under control of a software-programmable host. A bidirectional bus gives the necessary liberty for transferring data and control words back and forth. Each coprocessor, or helper engine as it is sometimes called, has a rather limited repertoire of instructions that it can accept. It sits idle until it receives a set of input data along with a start command. As an alternative, the data may be kept in the host's own memory all the time but get accessed by the coprocessor via direct memory access (DMA). Once local computation has come to an end, the coprocessor sets a status flag and/or sends an interrupt signal to the host computer. The host then accepts the processed data and takes care of further action. Application-specific instruction set processors Patterning the overall architecture after a program-controlled processor affords much more flexibility. Application-specific features are largely confined to the data processing circuitry itself. That Architectures of VLSI Circuits is, one or more data paths are designed and hardwired so as to support specific data manipulations while operating under control of a common microprogram. The number of ALUs, their instruction sets, the data formats supported, the capacity of local storage, etc. are tailored to the computational problems to be solved. What's more, the various data paths can be made to operate simultaneously on different pieces of data, thereby providing a limited degree of concurrency. The resulting architecture is that of an application-specific instruction set processor (ASIP).

Application-specific instruction set processor (ASIP) (a), multiple cooperating ASIPs (b).The hardware organization of an ASIP bears much resemblance to architectural concepts from general-purpose computing. As more and more concurrent data path units are added, what results essentially is a very-long instruction word (VLIW) architecture. An open choice is that between a multiple-instruction multiple-data (MIMD) machine, where an individual field in the overall instruction word is set apart for each data path unit, and a single-instruction multiple-data (SIMD) model, where a bunch of identical data paths works under control of a single instruction word. Several data items can thus be made to undergo the same operation at the same time. In an effort to better serve high-throughput video and graphics applications, many vendors enhanced their microprocessor families in the late 1990s by adding special instructions that provide some degree of concurrency.

During each such instruction, the processor's data path gets split up into several smaller subunits. A data path of 64 bit can be made to process four 16 bit data words at a time, for instance, provided the operation is the While the mono-ASIP architecture of affords flexibility, it does not provide the same degree of concurrency and modularity as the multiple processing units. A multiprocessor system built from specialized ASIPs, therefore, an interesting extension. In addition, this approach facilitates the design, interfacing, reuse, test, and on-going update of the various building blocks involved. However, always keep in mind that defining a proprietary instruction set makes it impossible to take advantage of existing compilers, debugging aids, assembly language libraries, experienced programmers, and other resources that are routinely available for industry-standard processors. Industry provides us with such a vast selection of micro- and signal processors that only very particular requirements justify the design of a proprietary CPU.

Example While generally acknowledged to produce more realistic renderings of 3D scenes than industry standard raster graphics processors, ray tracing algorithms have long been out of reach for real-time applications due to the myriad floating-point computations and the immense memory bandwidth they require. Hardwired custom architectures do not qualify either as they cannot be programmed or as ray tracing necessitates many data-dependent recursions and decisions. Same for all of them. The technique is best described as sub-word parallelism, but is better known under various trademarks such as multimedia extensions

(MMX), streaming SIMD extensions (SSE) (Pentium family), Velocity Engine, AltiVec, and VMX (PowerPC family). Reports on an interesting approach to expedite ASIP development whereby assembler, linker, simulator, and RTL synthesis code are generated automatically by system -level software tools. Product designers can thus essentially focus on defining the most appropriate instruction set for the processor in view of the target application.

Architectures of VLSI Circuits

Ray tracing may finally find more general adoption in multi-ASIP architectures that combine multiple ray processing units (RPUs) into one powerful rendering engine. Working under control of its own program thread, each RPU operates as a SIMD processor that follows a subset of all rays in a scene. The independence of light rays allows a welcome degree of scalability where frame rate can be traded against circuit complexity. The authors of [19] have further paid attention to defining an instruction set for their RPUs that is largely compatible with pre-existing industrial graphics processors.

Configurable computing

Another crossbreed between dedicated and general-purpose architectures did not become viable until the late 1990s but is now being promoted by FPL manufacturers and researchers The IEEE 1532 standard has also been created in this context. The idea is to reuse the same hardware for implementing sub functions that are mutually exclusive in time by reconfiguring FPL devices on the fly. As shown in fig.2.5, the general hardware arrangement bears some resemblance to the coprocessor approach of fig.2.3b, yet in-system configurable (ISC) devices are being used instead of hardwired logic. As a consequence, the course of operations is more sophisticated and requires special action from the hardware architects. For each major subtask, the architects must ask themselves whether the computations involved

a) Qualify for being delegated to in-system configurable logic,
b) Never occur at the same time or can wait until the FPL device becomes free, and
c) Whether the time for having the FPL reconfigured in between is acceptable or not.

Typically this would be the case for repetitive computations that make use of sustained, highly parallel, and deeply pipelined bit-level operations. When designers have identified some suitable sub function, they devise a hardware architecture that solves the particular computational problem with the resources available in the target FPGA or CPLD, prepare a configuration file, and have that stored in a configuration memory. In some sense, they create a large hardware procedure instead of programming a software routine in the customary way. Whenever the host computer encounters a call to such a hardware procedure, it configures the FPL accordingly by downloading the pertaining configuration file. From now on, all the host has to do[8].

THE ARCHITECTURAL ANTIPODES

Is to feed the “new” coprocessor with input data and to wait until the computation is completed.The host then fetches the results before proceeding with the next subtask.It thus becomes possible to support an assortment of data processing algorithms each with itsoptimum architecture or almost so from a single hardware platform. What often penalizesthis approach in practice are the dead times incurred whenever a new configuration is being loaded.Another price to pay is the extra memory capacity for storing the configuration bits for all operationmodes. Probably the most valuable benefit, however, is the possibility of being able to upgradeinformation processing hardware to new standards and/or modes of operation even after the systemhas been fielded.

Examples Transcoding video streams in real time is a good candidate for reconfigurable computing because of the many formats in existence such as DV, AVI, MPEG-2, DivX, and H.264. For each conversion scheme, a configuration file is prepared and stored in local memory, from where it is transferred into the reconfigurable coprocessor on demand. And should a video format or variation emerge that was unknown or unpopular at the time when the system was being developed, extra configuration files can be made available in a remote repository from where they can be fetched much like software plug-ins get downloaded via the Internet. The results from a comparison between Lempel–Ziv data compression with a reconfigurable coprocessor and with software execution on a processor have been summarized. A related application was to circumvent the comparatively slow PCI bus in a PC.

Extendable instruction set processors

This latest and most exotic approach pioneered by Stretch borrows from ASIPs and from configurable computing. Both a program-controlled processor and electrically reconfigurable logic are present on a common hardware platform. The key innovation is a suite of proprietary EDA tools that allows system developers to focus on writing their application program in C or C++ as if for a regular general purpose processor. Those tools begin by profiling the software code in order to identify sequences of instructions that are executed many times over. For each such sequence, reconfigurable logic is then synthesized into a dedicated and massively parallel computation network that completes within one clock cycle ideally at least. Finally, each occurrence of the original computation sequence in the machine code gets replaced by a simple function call that activates the custom-made data path logic.

In essence, the base processor gets unburdened from lengthy code sequences by augmenting his instruction set with a few essential additions that fit the application and that get tailor-made as an extension to the general procedure described here, an extra optimization step can be inserted before the coprocessor is configured. During this stage, the host would adapt a predefined generic configuration to take advantage of particular conditions of the specific situation at hand. Consider pattern recognition, for instance, where the template remains unchanged for a prolonged lapse of time, or secret-key (de)ciphering, where the same holds true for the key. As stated item 1, it is often possible to simplify arithmetic and almost on the fly. Yet, the existence of reconfigurable logic and the business of coming up with a suitable hardware architecture are hidden from the system developer. The fact that overall program execution remains strictly sequential should further simplify the design process.

Digest

Program execution on a general-purpose processor and hardwired circuitry optimized for one specific flow of computation are two architectural antipodes. Luckily, many useful compromises exist in between, and this is reflected. A general piece of advice is this: Rely on dedicated hardware only for those sub functions that are called many times and are unlikely to change; keep the rest programmable via software, via reconfiguration, or both. The architectural solution space viewed as a globe. While there are many ways to trade agility for computational efficiency and vice versa, the two seem to be mutually exclusive as we know of no architecture that would meet both goals at the same time. Their conceptual differences notwithstanding, many techniques for obtaining high performance at low cost are the same for general- and special-purpose architectures.

As a consequence, much of the material presented in this chapter applies to both of them. Yet, the emphasis is on dedicated architectures as the a priori knowledge of a computational problems offers room for a number of ideas that do not apply to instruction-set processor

architectures. Most data and signal processing algorithms would lead to grossly inefficient or even infeasible solutions if they were implemented in hardware as they are. Adapting processing algorithms to the technical and economic conditions of large-scale integration is one of the intellectual challenges in VLSI design. Basically, there is room for remodeling in two distinct domains, namely in the algorithmic domain and in the architectural domain. There exists an excellent and comprehensive literature on general-purpose architectures including.The historical evolution of the microprocessor is summarized in along with economic facts and trends.

Architectures of VLSI Circuits

There is room for remodeling in the algorithmic domain. In the algorithmic domain, the focus is on minimizing the number of computational operations weighted by the estimated costs of such operations. A given processing algorithm thus gets replaced by a different one better suited to hardware realization in VLSI. Data structures and number representation schemes are also subject to optimizations such as subsampling and/or changing from floating-point to fixed-point arithmetic's. All this implies that alternative solutions are likely to slightly differ in their functionality as expressed by their input-to-output relations.

Six examples when designing a digital filter, one is often prepared to tolerate a somewhat lower stopband suppression or a larger passband ripple in exchange for a reduced computational burden obtained, for instance, from substituting a lower order filter and/or from filling in zeros for the smaller coefficients. Conversely, a filter structure that necessitates a higher number of computations may sometimes prove acceptable in exchange for less stringent precision requirements imposed on the individual arithmetic operations and, hence, for narrower data words. In a decoder for digital error-correction, one may be willing to sacrifice 0.1 dB or so of coding gain for the benefit of doing computations in a more economical way. Typical simplifications to the ideal Viterbi algorithm include using an approximation formula for branch metric computation, truncating the dynamic range of path metrics, rescaling them when necessary, and restricting trace back operations to some finite depth. The autocorrelation function (ACF) has many applications in signal processing, yet it is not always needed in the form mathematically defined.

$$\mathrm{ACF}x\ x\ (k) = rx\ x\ (k) = \infty$$

$$n = -\infty$$

$$X\ (n) \cdot x(n + k)$$

Many applications offer an opportunity to relax the effort for multiplications because one is interested in just a small fragment of the entire ACF, because one can take advantage of symmetry, or because modest precision requirements allow for a rather coarse quantization of data values. It is sometimes even possible to substitute the average magnitude difference function (AMDF) that does away with costly multiplication altogether.

$$\mathrm{AMDF}x\ x\ (k) = r$$

$$x\ x\ (k) =$$

$$N$$

$$-1$$

$$n = 0$$

$$|x\ (n) - x(n + k)|$$

Code-excited linear predictive (CELP) coding is a powerful technique for compressing speech signals, yet it has long been left aside in favor of regular pulse excitation because of its prohibitive computational burden. CELP requires that hundreds of candidate excitation sequences be passed through a cascade of two or three filters and be evaluated in order to pick the one that fits best[9]. In addition, the process must be repeated every few milliseconds. Yet, experiments have revealed that the usage of sparse (up to 95% of samples replaced with zeros), of ternary (+1, 0, −1), or of overlapping excitation sequences has little negative impact on auditory perception while greatly simplifying computations and reducing memory requirements. In designing computational hardware that makes use of trigonometric functions, look-up tables (LUTs) are likely to prove impractical because of size overruns. Executing a lengthy algorithm [10], [11].

CONCLUSION

Physics-based EM modeling is a powerful tool for understanding and predicting the behavior of electromagnetic waves and fields in a wide range of systems. The use of fundamental physical principles and mathematical techniques allows for the development of accurate and detailed models that can be used to design and optimize complex devices and systems. Analytical techniques, such as the method of images and the Green's function method, provide closed-form solutions to Maxwell's equations for simple systems with known geometries and boundary conditions. These solutions can be used to develop simplified models for more complex systems.

REFRENCES

[1] B. Falkner and G. F. Schröder, “Cross-validation in cryo-EM-based structural modeling,” *Proc. Natl. Acad. Sci. U. S. A.*, 2013, doi: 10.1073/pnas.1119041110.

[2] J. Ismer, A. S. Rose, J. K. S. Tiemann, and P. W. Hildebrand, “A fragment based method for modeling of protein segments into cryo-EM density maps,” *BMC Bioinformatics*, 2017, doi: 10.1186/s12859-017-1904-5.

[3] T. Kim, Z. Liu, and S. X. D. Tan, “Dynamic reliability management based on resource-based EM modeling for multi-core microprocessors,” *Microelectronics J.*, 2018, doi: 10.1016/j.mejo.2018.01.024.

[4] N. Azzouza, K. Akli-Astouati, and R. Ibrahim, “Twitterbert: Framework for twitter sentiment analysis based on pre-trained language model representations,” in *Advances in Intelligent Systems and Computing*, 2020. doi: 10.1007/978-3-030-33582-3_41.

[5] U. Ependi, “PEMODELAN SISTEM INFORMASI MONITORING INVENTORY SEKRETARIAT DAERAH KABUPATEN MUSI BANYUASIN,” *KLIK - Kumpul. J. ILMU Komput.*, 2018, doi: 10.20527/klik.v5i1.124.

[6] S. Incerti, V. Ivanchenko, and M. Novak, “Recent progress of Geant4 electromagnetic physics for calorimeter simulation,” *J. Instrum.*, 2018, doi: 10.1088/1748-0221/13/02/C02054.

[7] X. Jia, J. Wang, Y. Cai, and Q. Zhou, “Electromigration design rule aware global and detailed routing algorithm,” in *Proceedings of the ACM Great Lakes Symposium on VLSI, GLSVLSI*, 2018. doi: 10.1145/3194554.3194567.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” *J. Mach. Learn. Res.*, 2003, doi: 10.1016/b978-0-12-411519-4.00006-9.

[9] D. Damayanti, M. F. Akbar, and H. Sulistiani, "Game Edukasi Pengenalan Hewan Langka Berbasis Android Menggunakan Construct 2," *J. Teknol. Inf. dan Ilmu Komput.*, 2020, doi: 10.25126/jtiik.2020721671.

[10] E. Karabelas *et al.*, "Towards a computational framework for modeling the impact of aortic coarctations upon left ventricular load," *Front. Physiol.*, 2018, doi: 10.3389/fphys.2018.00538.

[11] S. X. D. Tan, H. Amrouch, T. Kim, Z. Sun, C. Cook, and J. Henkel, "Recent advances in EM and BTI induced reliability modeling, analysis and optimization (invited)," *Integration, the VLSI Journal*. 2018. doi: 10.1016/j.vlsi.2017.08.009.

CHAPTER 4

MEMS-BASED IMU FOR POSE ESTIMATION

Dr. Ashutosh Anand, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id- ashutoshanand@presidencyuniversity.in

ABSTRACT:

Inertial measurement units (IMUs) are important sensors used in a variety of applications, including robotics, virtual reality, and navigation systems. An IMU typically consists of a combination of accelerometers and gyroscopes that measure linear and angular motion, respectively. MEMS-based IMUs have gained popularity due to their small size, low power consumption, and low cost. These sensors are commonly used for pose estimation, which involves estimating the orientation and position of a rigid body in space. In this paper, we will discuss the design and implementation of a MEMS-based IMU for pose estimation.

KEYWORDS:

Accelerometers, Angular Motion, Inertial Measurement, Gyroscopes, Sensors.

INTRODUCTION

The inertial measurement unit (IMU), an electrical device, uses a three-axis barometer and three-axis gyroscope to detect accelerations and angular velocities gyroscope to calculate the position and location of an object. IMUs have long been used in applications related to navigation and aerospace. Those were the first big, heavy IMUs. Ultimately, as MEMS-based technology developed. IMU is reduced, as is the quantity of power utilized. The MEMS-based IMU sensor may be utilized for pose estimation. It is widely used in the robotics, navigation, and consumer electronics industries. Orientation and position are vital in robots that climb, traverse terrain, navigate, etc., but the MEMS-based IMU's results are intricate yet turbulent and unstable. Because of the noise in the accelerometer data from the IMU It is susceptible to external vibration contamination when subjected to uniform acceleration. It is difficult to measure gravitational acceleration precisely in a vibrating environment. In order to get accurate data over the long term, the accelerometer alone cannot be utilized for orientation evaluation even if it provides constant data that is free from drifts[1].

A gyroscope is used to monitor angular acceleration and four-dimensional motion, and unlike accelerometers, its outputs are less vulnerable to external noise. Hence, tri-axis hardware and software based on gyroscopes IMU MEMS sensor MEMS IMU sensors work the best for determining orientation. The disadvantage of employing a gyroscope is that it introduces bias into angular velocity measurements, which leads to data fluctuations over time. For a short while, the gyroscope's information may be believed. The combination of IMU and accelerometer data is essential for weighing the advantages and disadvantages of gyroscopes in order to get an accurate orientation estimation. The noisy accelerometer data are sent to the time series filter. When the position is smoothed and the double acceleration integration is used to produce the jarring variations in the gathered data. One of the most popular methods for data fusion in the IMU to determine an object's orientation is the complimentary filter, which estimates movement using information from the accelerometer and gyroscope. In this research, the IMU's accelerometer measurement is stabilized and the position is determined using a moving average filter. The complementary filter has a relatively simple methodology and few computations.

In order to evaluate direction and movement, IMU data is paired with a complementary filter in a comparison study. As a consequence, it may be employed in embedded systems. Gyroscope data is single integrated, while accelerometer data from the IMU is double integrated for position. The accelerometer data has an average filter applied. Data on inclination within static analysis parameters for each instance are taken into account. a comparison of the experimental results generated using these various methods. The complementary filter was initially developed by Shane Colton in 2007. In contrast to the pedometer, which is instantaneously obtained by integrating the output of the gyroscope, a filter utilizes data to execute low-pass filtering on moderate heading estimations and high-pass screening on rough approximations. A full estimation. The orientation is established by merging the two kinds of information. In order to rectify the integrated angle from the magnetometer over short periods of time and the drift of the angle over long periods of time, the supplementary filter employs the low-pass filter data from the accelerometer. The offset of the gyroscope is continually updated and corrected. This demonstrates how quickly changing and drift-free the computed angle[2].

MEMS-Based IMU Design

The design of a MEMS-based IMU involves selecting the appropriate accelerometers and gyroscopes, and integrating them into a system that can accurately measure linear and angular motion. In general, MEMS-based accelerometers and gyroscopes use a micro-electromechanical system (MEMS) to sense motion[3], [4]. The MEMS device consists of a proof mass that is suspended by a spring, which moves in response to motion. The displacement of the proof mass is detected using capacitive or piezo resistive sensors.

Accelerometers

Accelerometers are used to measure linear acceleration, and they typically have a range of ±2g to ±16g. The output of an accelerometer is proportional to the acceleration experienced by the sensor. The sensitivity of an accelerometer is defined as the output change per unit acceleration, typically given in units of mV/g. The sensitivity can vary due to the manufacturing process and environmental factors such as temperature.

Gyroscopes

Gyroscopes are used to measure angular velocity, and they typically have a range of ±250°/s to ±2000°/s. The output of a gyroscope is proportional to the angular velocity experienced by the sensor. The sensitivity of a gyroscope is defined as the output change per unit angular velocity, typically given in units of mV/ (°/s). The sensitivity can vary due to the manufacturing process and environmental factors such as temperature[5].

DISCUSSION

Sensor Fusion

Sensor fusion involves combining the measurements from multiple sensors to improve the accuracy and reliability of the IMU. In general, the accelerometer measures the gravity vector, while the gyroscope measures the rotational motion. The gravity vector can be used to estimate the orientation of the sensor with respect to the gravity direction, which can be used to correct for the drift of the gyroscope. [6], [7] A complementary filter is commonly used to combine the measurements from the accelerometer and gyroscope, which improves the accuracy of the IMU.

Pose Estimation

Pose estimation involves estimating the orientation and position of a rigid body in space. The orientation can be represented using Euler angles, quaternion, or rotation matrix. The position can be represented using Cartesian coordinates or spherical coordinates. There are several methods for pose estimation using IMUs, including the Kalman filter, the extended Kalman filter, and the unscented Kalman filter. The Kalman filter is a recursive algorithm that estimates the state of a system based on noisy measurements. The filter works by predicting the state of the system based on a model, and then updating the state based on measurements. The Kalman filter is commonly used in navigation systems to estimate the position and velocity of a vehicle. The extended Kalman filter (EKF) is a variant of the Kalman filter that is used for nonlinear systems. The EKF approximates the nonlinear system with a linear system by using the first-order Taylor series expansion. The EKF is commonly used in robotics to estimate the pose of a robot.

The unscented Kalman filter (UKF) is a variant of the Kalman filter that is used for nonlinear systems. The UKF uses a deterministic sampling technique called the unscented transform to propagate the state and covariance through the nonlinear function. The CORDIC (coordinate rotation digital computer) family of algorithms is one such Compromise that was put to service in scientific pocket calculators in the 1960s and continues to find applications in DSP. Note that CORDIC can be made to compute hyperbolic and other transcendental functions too. Computing the magnitude function $m = \sqrt{a2 + b2}$ is a rather costly proposition in terms of circuit hardware. Luckily, there exist at least two fairly precise approximations based on add, shift, and compare operations exclusively. Better still, the performance of many optimization algorithms used in the context of demodulation, error correction, and related applications does not suffer much when the computationally expensive 2 -norm gets replaced by the much Simpler 1 - or ∞-norm. See for an example.

The common theme is that the most obvious formulation of a processing algorithm is not normally the best starting point for VLSI design. Departures from some mathematically ideal algorithm are almost always necessary to arrive at a solution that offers the throughput and energy efficiency requested at economically feasible costs. Most algorithmic modifications alter the input-to-output mapping and so imply an implementation loss, that is a minor cut-back in signal-to-noise ratio, coding gain, bit-error-rate, mean time between errors, stopband suppression, passband ripple, phase response, false-positive and false-negative rates, data compression factor, fidelity of reproduction, total harmonic distortion, image and color definition, intelligibility of speech, or whatever figures of merit are most important for the application. Figure 1 illustrate the MEMS-Based IMU for pose estimation.

Experience tells us that enormous improvements in terms of throughput, energy efficiency, circuit size, design effort, and agility can be obtained by adapting an algorithm to the peculiarities and cost factors of hardware. Optimizations in the algorithmic domain are thus concerned with “How to tailor an algorithm such as to cut the computational burden, to trim down memory requirements, and/or to speed up calculations without incurring unacceptable implementation losses.”

What the trade-offs are and to what extent departures from the initial functionality are acceptable depends very much on the application. It is, therefore, crucial to have a good command of the theory and practice of the computational problems to be solved. Digital signal processing programs often come with floating-point arithmetic’s. Re-Implementing them in fixed-point arithmetic’s, with limited computing resources, and with

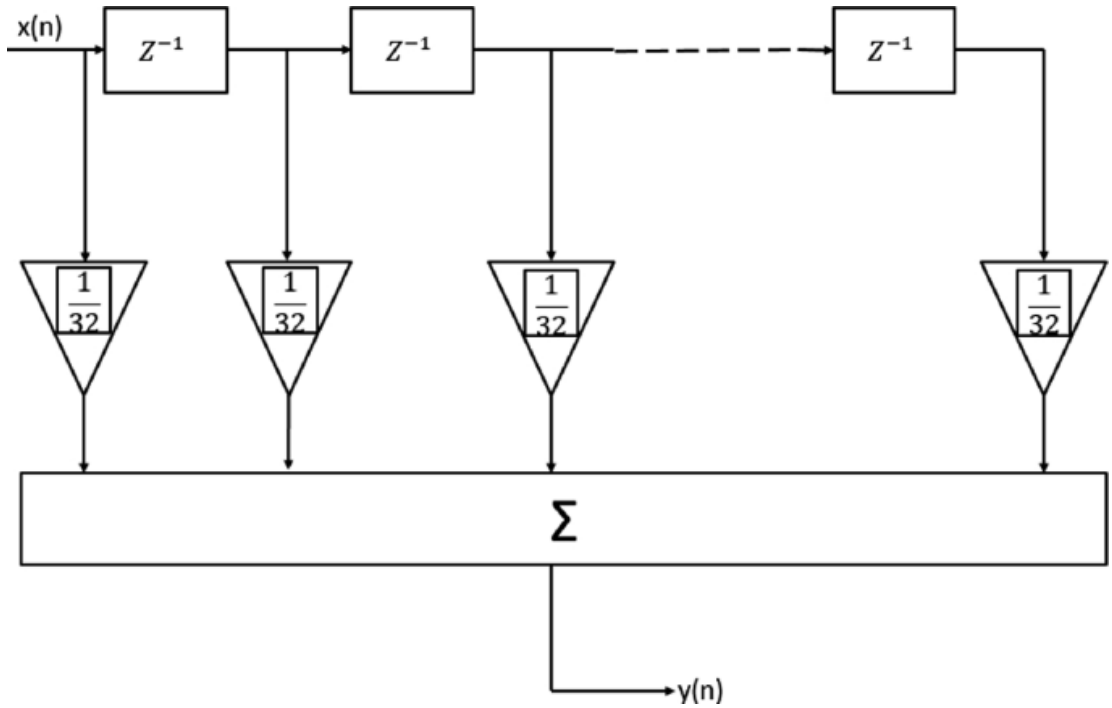


Figure 1: illustrate the MEMS-Based IMU for pose estimation.

Minimum memory results in an implementation loss. The effort for finding a good compromise between numerical accuracy and hardware efficiency is often underestimated. The necessity to validate trimmed-down implementations for all numerical conditions that may occur further ads to the effort. It is not uncommon to spend as much time on issues of numerical precision as on all subsequent VLSI design phases together and there is room in the architectural domain in the architectural domain, the focus is on meeting given performance targets for a specific data processing algorithm with a minimum of hardware resources. The key concern is "How to organize data paths, memories, controllers, and other hardware resources for implementing some given computation flow such as to optimize throughput, energy efficiency, circuit size, design effort, agility, overall costs, and similar figures of merit while leaving the original input-to-output relationship unchanged except, possibly, for latency."

As computations are just reorganized, not altered, there is no implementation loss at this point. Given some data or signal processing algorithm, there exists a profusion of alternative architectures although the number of fundamental options available for reformulating it is rather limited. This is because each such option can be applied at various levels of detail and can be combined with others in many different ways. Our approach is based on reformulating algorithms with the aid of equivalence transforms. The remainder of this chapter gives a systematic view on all such transforms and shows how they can be applied to optimize VLSI architectures for distinct size, throughput, and energy targets.

Systems engineers and VLSI designers must collaborate

Systems theorists tend to think in purely mathematical terms, so a data or signal processing algorithm is not much more than a set of equations to them. To meet pressing deadlines or just for reasons of convenience, they tend to model signal processing algorithms in floating-point arithmetic's, even when a fairly limited numeric range would amply suffice for the application. This is typically unacceptable in VLSI architecture design and establishing a lean bit-true software model is a first step towards a cost-effective circuit.

Generally speaking, it is always necessary to balance many contradicting requirements to arrive at a working and marketable embodiment of the mathematical or otherwise abstracted initial model of a system. A compromise will have to be found between the theoretically desirable and the economically feasible. So, there is more to VLSI design than just accepting a given algorithm and turning that into gates with the aid of some HDL synthesis tool. Algorithm design is typically carried out by systems engineers whereas VLSI architecture is more the domain of hardware designers. The strong mutual interaction between algorithms and architectures mandates a close and early collaboration between the two groups.

We propose a more precise definition of "void growth phase," "void incubation phase," and illustrate the compact models for the nucleation time estimation formula and calculating the durations of the three stages' nucleation, incubation, and failure. In addition, the three-phase Electromagnetic model can forecast changes in wire resistance patterns, which are more in line with the data provided via experimentation. Added to We also demonstrate the application of this research to the multi-segment connection the quick saturation volume estimate technique for multi-segment helps to connect, which are important for the investigation of the EM failure time to failure Oriented movement of electrically charged particles is known as electro migration (EM).

Due to the momentum, copper (Cu) atoms move in the direction of the applied electrical field between the conducting electrons and the atoms. Atoms go toward the anode terminal of something like the metal wire following the path either with lattice atoms or impurities electrons that conduct. Metal density is reduced as a consequence of this directed atomic movement at the cathode and a comparable metal buildup at the metal's anode end wire. Because atoms can't simply leave, there is this depletion and buildup the metal volume because the so-called diffuser barriers are keeping the metal wires within typically, metallic objects are used to contain or insert a Cu damascene connecting wire barriers like Ta and topped with either a metallic or a dielectric.

The atom depletion causes a change in wire volume as a consequence. In complement to the semi-physics-based Black and Blech formulations, EM many more physics models of early experimental results-supported models, new connection wire has been designed using the electro migration phenomena structures. The most advanced stress evolution-based EM models as described by the Korhonen's equation are presented first in this section. Next, we will demonstrate the new three-phase EM model as well as newly established physics-based Cu damascene measured values are more compatible with the EM model connect the wires. Blech made the first discovery that the termination of strips carrying current The next movement, which is the Nernst-Einstein velocity, is the drift speed for flow where Fem is the EM-induced force brought about by the electronic wind is the chemical diffusion coefficient and is calculated using the formula Da = D0 exp Ea is the EM absorption coefficient, while D0 is the absorption coefficient the metal's flux[8].

The two primary forces acting on atoms during electromagnetic radiation (EM) are the electron wind-induced force (Fem) and the back-force caused by the concentration of atoms variations (or stress gradients brought on by the atoms being depleted somewhere at cathode end and the atoms accumulating at the electrochemical end. A high-level block diagram of the outcome shows data paths, controllers, memory, interfaces, and important signals. Also, a rough layout is being created. Typically, simulations are used to verify an architecture, with each key component being represented by a unique behavioral model. The work is then taken down to the register transfer level (RTL), which is a more detailed level, where the circuit is treated as a group of storage components linked by just combinational sub circuits. How to

implement arithmetic and logic units is a crucial question at this point (e.g. ripple-carry, carry-look ahead, carry-select)[9].

The result is a collection of more intricate diagrams that show each and every register, memory, and significant combinational logic block. Combinational functions are stated in behavioral rather than structural terms, in contrast to gate-level schematics. The RTL code is mostly debugged using simulations. Based on the more complete data that is now available, the floorplan is adjusted and compared to the die size and cost objectives for the finished product. Now is also the time to choose the best design level for each circuit block, such as synthesis, schematic entry, or manual layout.

VLSI circuit architectures and logic design. The gate-level netlist's Boolean optimization and translation are essentially automated. A fabrication depth (e.g., full-custom vs. semi-custom vs. FPL), one or more cell libraries (e.g., by Artisan vs. LSI Logic vs. Xilinx), a circuit style (e.g., static vs. dynamic CMOS logic), a fabrication technology (e.g., CMOS vs. BiCMOS), and a manufacturing process are now being firmly committed (e.g. L130 by UMC vs. HCMOS9gp by ST). Calculations are being made to determine the delays and energy losses incurred by the different computational and storage procedures. Where feasible, sub circuits that were shown to be performance-limiting during pre-layout study are rebuilt or re-optimized. A whole set of gate-level schematics and/or netlists that have been verified by electrical rule check (ERC), logic simulation, timing verification, and power estimate are the end result. Increase in testability Design problems, manufacturing issues, or both might lead to an IC that isn't operating properly.

To ensure the proper functioning of millions of transistors packed into a device with little more than a few hundred pins, certain safeguards must be made. The idea behind design for test (DFT) is to build auxiliary circuitry on top of the payload logic to increase the controllability and observability of inner circuit nodes. A test vector set is also created to help differentiate between bad and good circuits. Usually, such a vector set consists of tens of thousands or millions of stimuli and anticipated reactions [10], [11].

Testability is graded using a process known as fault grading, which compares the number of manufacturing flaws that can actually be found using the test vector set in question to the total number of potential flaws. Up until a suitable fault coverage is attained, both the test circuits and the test patterns are repeatedly improved. Physical layout. The arrangement of several sub circuits, devices, and their connections on a piece of semiconductor material are all concerns of physical design. The goal of floor planning is to fit the principal circuit blocks into a rectangular area that is as compact as feasible while minimizing the performance-impacting consequences of interconnect delays. Distribution of clocks and electricity at the chip level must also be addressed. To hold the bond pads and the top-level layout blocks, a pad frame has to be created. Each cell is given a precise position on the die during the ensuing place and route (P&R) procedures before the courses of countless metal wires that will transmit electrical impulses between those cells are established. The estimated connection delays that become available throughout the procedure often need re-optimizing the circuit logic. The last stage Block isolation, scan testing, and BIST are common strategies. With the use of additional multiplexers, block isolation makes the majority of circuit blocks accessible from outside a chip so that stimuli may be applied and responses can be assessed through package pins while in test mode.

Built-in self-test (BIST) is intended to relocate stimulus creation and response checking onto the chip itself, and to effectively produce a "go/no go" answer. Block isolation and BIST are often used to evaluate on-chip memory. The reader is directed to the specialist literature, such

as, since DFT, test vector preparation, and automated test equipment (ATE) are not covered in this article. Similar to how layout design is a component of physical design, floor planning is. So what makes a difference? In contrast to layout design, which is focused with creating intricate geometric patterns on carpet, floor planning is concerned with dividing a flat into rooms and halls. Cells are often abstracted up to this point to their outlines since the interior layout specifics of the cells don't actually relevant for floor planning, location, and route. For such abstract perspectives, detailed layout data must be put in in order to be ready for IC fabrication. The end result is a large collection of polygons made up of each mask layer. To prevent deadly accidents, the whole layout data must be thoroughly examined before manufacture. The verification of physical designs uses a variety of software technologies.

Sign-off. An IC vendor promises to supplying circuits that operate like the post-layout simulation model (same functionality for the test vector set given by the client, same or better speed, same or lower power) by approving a design for prototype manufacturing. No client will pay for manufactured components that do not meet this criteria, thus the vendor wants to make sure the design complies with both company-specific standards and good engineering practice before moving further. Regular checks are made for DRC, manufacturability, ERC, LVS, post-layout simulation, and fault coverage. Timing verification, clocking discipline, power and clock distribution, circuit design aesthetic, test architecture, and other topics are often covered by inspection[12].

After this pretty broad review, a few remarks are required. In practice, the division into distinct subtasks is rarely as neat and obvious. The majority of software tools must operate across several layers of abstraction due to the consequences of deep submicron technologies and the pursuit of optimal solutions. Because of the associated layout parasitic and connection delays, it is no longer feasible to install and route a gate-level netlist without modifying the circuit logic.

This is shown in the illustration by the collaborative improvement of layout data and netlists. Design only ever happens as a linear series of stages in an ideal world. For a genuinely good outcome, some back and forth between the many subtasks is unavoidable. Moreover, not every IC development project specifically covers every design step. Depending on the kind, depth, and degree of manufacture of the circuit, certain design steps are omitted or outsourced, or given to experts at other businesses.

As there are no algorithmic or architectural issues to be resolved, the design of a straightforward glue logic chip, for example, starts at the logic level. Where industrial partnership model discussions will take place. While angular boxes refer to construction activities, the rounded ones stand for analysis and verification steps. A backward sorrow implies that any problem uncovered during such an analysis triggers corrective action say the designer. The results from construction steps are subject to immediate verification, which is typical for VLSI. The reason is that correcting a mistake becomes more and more onerous the further the design process has progressed. Correcting a minor functional bug after layout design, for instance, would require redoing several design stages and would waste many hours of labor and computer time. Also, a functional bug can be uncovered more effectively from a behavioral or RTL model than from a post-layout transistor-level netlist because simulation speed is orders of magnitude higher and because automatic response checking is much easier to implement for logic and numeric data types than for analogue waveforms. A critical point is reached when first silicon is going to be produced.

While it is possible stop cut and add wires using advanced and expensive equipment such as focused ion-beam (FIB) technology to patch a malfunctioning prototype, there is virtually no

way to fix bugs in volume production. Depending on the circuit's size, fabrication depth, process, and manufacturer, expenses somewhere between 12 kUSD and 1 MUSD are involved with preparation soft photomasks, tooling, wafer processing, and preparation of probe cards and evaluation of preproduction samples. Any design flaw found after prototype fabrication thus implies the waste soft important sums of money. To make things worse, with turnaround times ranging between two weeks and three months, as product's arrival on the market is delayed so much that the chip is likely to miss its window soft opportunity.

Redesigns are so devastating for the business that the entire semiconductorindustry has committed itself to "first-time-right" design as a guiding principle. To avoid them, VLSI engineers typically spend much more time verifying a circuit than actually designing it. They suggest how electronic design automation, cell libraries, and purchased know-how help speed up the design process. Keeping pace with the breathtaking progress of fabrication technology is in fact one of the major challenges for today's VLSI designers. While there is not too much of a difference in the front-end flow, back-end design for field programmable logic (FPL) differs somewhat from that depicted. The preliminary state-level netlist obtained from HDL synthesis is mapped onto configurable blocks available in the target FPGA or CPLD device. After the EDA software has decided how to run all necessary interconnects using the wires, switches, and drivers available, the result is converted into a configuration bit stream for download into the FPL device. As FPGAs and CPLDs some with many diverse architectures, product-specific back-end tools made available by the sepal vendor are used for this procedure. Whoever has learned to design full-custom ICs is in an excellent position for designing semi-custom ICs and to design with field-programmable logic, but not necessarily the other \sway round. Library development occurs quite separately from actual IC design as cell-based circuits largely dominate VLSI.

Once the set of prospective library cells has been defined functionally, library development proceeds in three major phases. Electrical design deals with implementing logic functions as transistor-level networks and with sizing the individual devices such as to find an optimum trade-off between performance, circuit complexity, and energy efficiency. During the subsequent layout design, the locations and geometric shapes of individual devices are defined along with the shapes of the wires running in between. The goal is to obtain leaf cells that are compact, fast, energy-efficient, suitable for automatic place and route (P&R), and that can be \manufactured with maximum yield. Verification includes the customary ERC, DRC, manufacturability analysis, extraction, and LVS procedures. Next the electrical and timing parameters that are to be included in data sheets and stimulation models of the cells are determined. This library characterization step typically relies Semi-custom ICs and FPL rely on prefabricated primitives anyway.

DESIGN FLOW IN DIGITAL VLSI

On repeated continuous-time continuous-value simulations under varying load, ramp, and operating conditions. Designing, characterizing, documenting, and maintaining a cell library is a considerable effort as multiple design views must be prepared for each cell, including a datasheet with functional, electrical, and timing specifications.

In order to protect their investments, most library vendors consider their library cells to be proprietary and are not willing to disclose how they are constructed internally. They supply datasheets, scions, simulation models, and abstracts, but no transistor-level schematics and no layouts. Under this scheme, detailed layouts are to be substituted for all cell abstracts by the vendor before mask preparation can begin.

CONCLUSION

REFERENCES

[1] J. Zhao, "A Review of Wearable IMU (Inertial-Measurement-Unit)-based Pose Estimation and Drift Reduction Technologies," in *Journal of Physics: Conference Series*, 2018. doi: 10.1088/1742-6596/1087/4/042003.

[2] S. S. Bangera, T. D. Shiyana, G. K. Srinidhi, Y. R. Vasani, and M. Sukesh Rao, "MEMS-Based IMU for Pose Estimation," in *Lecture Notes in Electrical Engineering*, 2020. doi: 10.1007/978-981-15-0626-0_1.

[3] Y. Wu, X. Niu, J. Du, L. Chang, H. Tang, and H. Zhang, "Artificial marker and MEMS IMU-based pose estimation method to meet multirotor UAV landing requirements," *Sensors (Switzerland)*, 2019, doi: 10.3390/s19245428.

[4] T. McGrath and L. Stirling, "Body-worn imu human skeletal pose estimation using a factor graph-based optimization framework," *Sensors (Switzerland)*, 2020, doi: 10.3390/s20236887.

[5] Z. Zhang, C. Wang, W. Qin, and W. Zeng, "Fusing wearable IMUs with multi-view images for human pose estimation: A geometric approach," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. doi: 10.1109/CVPR42600.2020.00227.

[6] S. Su, Y. Zhou, Z. Wang, and H. Chen, "Monocular Vision-and IMU-Based System for Prosthesis Pose Estimation during Total Hip Replacement Surgery," *IEEE Trans. Biomed. Circuits Syst.*, 2017, doi: 10.1109/TBCAS.2016.2643626.

[7] D. Gautam, A. Lucieer, C. Watson, and C. McCoull, "Lever-arm and boresight correction, and field of view determination of a spectroradiometer mounted on an unmanned aircraft system," *ISPRS J. Photogramm. Remote Sens.*, 2019, doi: 10.1016/j.isprsjprs.2019.06.016.

[8] L. Han, Y. Lin, G. Du, and S. Lian, "DeepVIO: Self-supervised Deep Learning of Monocular Visual Inertial Odometry using 3D Geometric Constraints," in *IEEE International Conference on Intelligent Robots and Systems*, 2019. doi: 10.1109/IROS40897.2019.8968467.

[9] J. D. Hol, F. Dijkstra, H. Luinge, and T. B. Schöny, "Tightly coupled UWB/IMU pose estimation," in *Proceedings - 2009 IEEE International Conference on Ultra-Wideband, ICUWB 2009*, 2009. doi: 10.1109/ICUWB.2009.5288724.

[10] Y. Xiao, X. Ruan, J. Chai, X. Zhang, and X. Zhu, "Online IMU self-calibration for visual-inertial systems," *Sensors (Switzerland)*, 2019, doi: 10.3390/s19071624.

[11] C. Malleson, J. Collomosse, and A. Hilton, "Real-Time Multi-person Motion Capture from Multi-view Video and IMUs," *Int. J. Comput. Vis.*, 2020, doi: 10.1007/s11263-019-01270-5.

[12] Y. Zhang *et al.*, "Brain-inspired computing with memristors: Challenges in devices, circuits, and systems," *Applied Physics Reviews*. 2020. doi: 10.1063/1.5124027.

CHAPTER 5

DATA FLOW VERIFICATION IN SOC USING FORMAL TECHNIQUES

Dr. Joseph Anthony Prathap, Associate Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id- joseph.anthony@presidencyuniversity.in

ABSTRACT:

The design of a System-on-Chip (SOC) is a complex process that involves integrating various components, such as processors, memories, and input/output (I/O) interfaces, onto a single chip. The verification of the data flow between these components is critical to ensure that the SOC functions correctly. Formal techniques are increasingly being used to verify the data flow in SOC designs. In this article, we will discuss the use of formal techniques for data flow verification in SOC designs.

KEYWORDS:

Chip, Data Flow, Input, Output, Techniques.

INTRODUCTION

Modern civilization places a high priority on security, which has a big impact on system-on-chip architectures (SoCs). Expanding using intellectual properties (IPs). Systems demand more time to market while requiring engineers to do less design effort. As a SoC often involves a large number of IPs, IP component security that incorporates the necessary designs and requirements is seen as a critical design need. It is critical to lessen or simply eliminate the chance of losing important designs and qualities to fraud. In line with the design criteria, it is important to check data transfer in the idea under all constraints and needs. Accidental functional pathways that might expose sensitive information due to design weaknesses in security should be carefully analyzed and validated to assure secure data flow. Designing SoCs with IP is essential for the VLSI industry. Early in the 1990s, the semiconductor industry started using VLSI IP recycling and composting design concepts. Intellectual property protection and preventing data and information theft have always been challenging issues to handle. The capacity of available circuits rose faster as technology evolved at a Moore's Law pace, outpacing the ability of VLSI architects and EDA tools to handle the ensuing design challenge[1], [2].

The efficiency gap that followed prevented engineers from producing each transistor on a separate die. To fill this gap in reuse-based design, industries employed IP. These IPs were the most valuable assets of the companies. The exponential growth of Fagan IP recycling methods forced the birth of Patent rights engineering in order to tackle various security risks, such as tampering and reverse engineering. Patents, Treaties, trademarks, copyrights, and trade secrets are a few examples of common IP protection deterrents. They do not, however, invariably avoid violating IP privacy since the owner of the IP might lose money if you are discovered. IP protection has been identified as one of the key elements. Technology that enable the reuse-based design process as a result of globalization increase the potential for dishonest IP suppliers to introduce dangerous code into the SoC design pipeline. A verification technique may be used to examine the third-party IPs for the leakage of harmful

information. To relax time-to-market constraints Engineers developing fabless SoCs blend internal IP cores with third-party IPs to cut test and production expenses.

The design might fail or accidentally create a backdoor that would let an attacker to access the device without authorization if a dishonest IP provider added damaging circuits or instructions during production. Contrarily to formal verification methods, which give complete coverage, counterexamples offer an additional explanation. An Internet governance system based on formal property inspection (FPV) may be used for verification and rapid prototyping. Information concealment suggests that IP reuse and sharing, although beneficial, may provide serious security risks. So, it may be beneficial to include information masking of an inherent ownership evidence as part of the system architecture.

In the beginning of the IP system life cycle, more information may be added to the design, which helps with copyright identification. The system security design, if safeguards are made during the installation of upgraded techniques. By looking for security weaknesses, it's possible to breach laws protecting confidential information and intellectual property. Protocol-level protection is one of the defending tactics. Pattern watermarking and scan chain systems are expected to be used to offer security, while testing the logic obfuscation method may protect the design from overbuilding and IP or IC reverse engineering. The original netlist is changed by the obfuscation approach, which when used with the correct key is logically equivalent to the original design.

Moreover, it has been shown that even while attackers may use circuit extraction to do reverse engineering in order to get the gate level netlist, they are unable to determine the Logic functions that are hidden. To cut down on design time and TTM, IPs are used as fundamental design elements. expanded systems Pre-verified IPs eliminate the need for each module verification and functioning portions, allowing designers to concentrate only on the connections between the modules. Verification is vital for defending IPs and its data produced from scams since critical information like password and account PINs may also be included in modern SoCs. The VLSI industry long ago became entirely dependent on electronic design automation (EDA) software. There is not one single step that could possibly be brought to an end without the assistance of sophisticated computer programmers. The sheer quantity of data necessary to describe a multi-million \transistor chip makes this impossible.

The design flow outlined in the previous section gives a rough side of the variety of CAE/CAD programmers that are required to pave the way for VLSI and FPL design. While a few vendors can take pride in offering a range of products that covers all stages from system level decision making down to physical layout, much of their effort tends to focus on relatively small portions of the overall flow for reasons of market penetration and profitability. Frequent mergers and acquisitions are another characteristic trait of the EDA industry [3].

Truly integrated design environments and seamless design flows are hardly available off the shelf. Also, the idea of integrating numerous EDA tools over a common design database and with a consistent user interface, once promoted as front-to-back environments, aka frameworks, has lost \momentum in the marketplace in favor of point tools and the “best in class” approach. Design slows are typically pieced together from software components of various origins. The presence of \software tools, design kits, and cell libraries from multiple sources in conjunction with the absence of \agreed-on standards adds a lot of complexity to the maintenance of a coherent design environment.

DISCUSSION

SoC designs involve multiple components that interact with each other through various data transfer mechanisms, such as buses, bridges, and DMA controllers. The correct operation of these components is crucial to ensure the overall functionality of the SoC. The traditional approach to verifying the data flow in SoC designs involves simulation-based techniques. However, simulation-based verification can be time-consuming and does not guarantee complete coverage of the design space. Formal verification techniques can overcome these limitations and provide a higher degree of confidence in the correctness of the design.

Formal techniques for SoC verification involve the use of mathematical models to verify the correctness of the design. These techniques are based on mathematical reasoning and do not rely on simulation-based testing. Formal techniques can be used to verify a wide range of properties, such as data integrity, deadlock freedom, and absence of race conditions. There are several formal techniques that can be used for SoC verification, such as model checking, theorem proving, and equivalence checking[4], [5]. These techniques differ in their underlying algorithms and the types of properties they can verify. Data Flow Verification in SoC using Formal Techniques Data flow verification in SoC involves verifying the correctness of the data transfer mechanisms between the various components of the SoC. This involves ensuring that the data is transferred correctly and that there are no errors, such as data corruption or data loss. Data flow verification can be performed at different levels of abstraction, such as RTL (Register Transfer Level), gate level, or system level. There are several formal techniques that can be used for data flow verification in SoC designs, such as model checking and theorem proving.

Model Checking for Data Flow Verification

Model checking is a formal verification technique that involves the automatic exploration of a state space to verify the correctness of a design. Model checking involves the creation of a formal model of the design, which is then used to explore the state space of the design. The state space of a design is the set of all possible states that the design can be in. Model checking can be used to verify various properties of the design, such as data integrity and deadlock freedom. Model checking can be used for data flow verification in SoC designs[6] by creating a formal model of the data transfer mechanisms between the various components of the SoC. The formal model can be created using a hardware description language, such as Verilog or VHDL. The model can then be used to explore the state space of the design to verify the correctness of the data transfer mechanisms.

Theorem Proving for Data Flow Verification

Theorem proving is a formal verification technique that involves the use of mathematical logic to prove the correctness of a design. Theorem proving involves the creation of a set of axioms and theorems, which are then used to prove the correctness of the design. Theorem proving can be used to verify various properties of the design, such as data integrity and absence of race conditions. Theorem proving can be used for data flow verification in SoC designs by creating a set of axioms and theorems that describe the data transfer mechanisms between the various components of the SoC[7]. The axioms and theorems can be written in a formal language, such as first-order logic. Many of the practical difficulties with setting up efficient design flows are left to EDA. Architectures of VLSI Circuits and can sometimes become a real nightmare. It is to be hoped that this trend will be reversed one day when customers are willing to pay more attention to design productivity than to layout density and circuit performance.

1.4 Field-programmable logic

The general idea behind programmable logic has been introduced. The goal of this section is to explain the major differences that separate distinct product families from each other. Key properties of any FPL device are fixed by decisions along two dimensions taken at development time. A first choice refers to how the device is being configured and how its configuration is stored electrically while a second choice is concerned with the overall organization of the hardware resources available to customers. Customers, in this case, are design engineers who want to implement their own circuits in an FPL device[8].

Configuration Technologies

Static memory. The key element here is an electronic switch such as a transmission gate, a pass transistor, or a three-state buffer that gets turned "on" or "off" under control of a configuration bit. Unlimited programmability is obtained from storing the configuration data in SRAM cells or in similar on-chip sub circuits built from two cross-coupled inverters. As a major drawback, the circuit must (re)obtain its entire configuration from outside whenever it is being powered up. The problem is solved in one of three possible ways, namely (a) by reading from a dedicated bit-serial or bit-parallel off-chip ROM, (b) by downloading a bit stream from a host computer, or (c) by long-term battery backup. Re configurability is very helpful for debugging.

It permits one to probe inner nodes, to alternate between normal operation and various diagnostic modes, and to patch a design once a flaw has been located. Many RAM-based FPL devices further allow reconfiguring of their inner logic during operation, a capability known as in-system configuration (ISC) that opens a door towards configurable computing. UV-erasable memory. Electrically programmable read-only memories (EPROM) rely on special MOSFETs where a second gate electrode is sandwiched between the transistor's bulk material underneath and a control gate.

The name floating gate captures the fact that this gate is entirely surrounded by insulating silicon dioxide material. An electrical charge trapped there determines whether the MOSFET, and hence the programmable link too, is "on" or "off". More precisely, the presence or absence of an electrical charge m modifies the MOSFET's threshold voltage and \so determines whether the transistor will conduct or not when a voltage is applied to its control gate during \memory readout operations. Figure 1 illustrate the Data Flow Verification in SoC Using Formal Techniques.

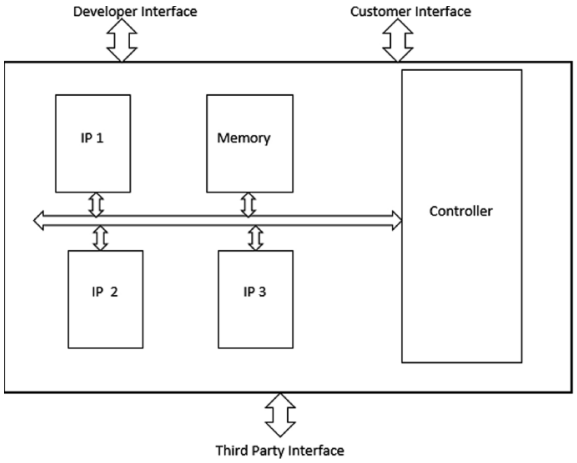


Figure 1: illustrate the Data Flow Verification in SoC Using Formal Techniques.

FPL configuration technologies (simplified, programming circuitry not shown) (simplified, programming circuitry not shown). Switch steered by static memory cell, MOSFET controlled by a charge trapped on a floating gate, fuse, and santifuse.Charging occurs by way of hot electron injection from the channel. That is, a strong lateral field applied between source and drain accelerates electrons to the point where they get injected through the thin dielectric layer into the floating gate. The necessary programming voltage in the order of 5 to 20 V is typically generated internally by an on-chip charge pump. Erasure of the charge is obtained by shining ultraviolet (UV) radiation on the chip, thereby causing the charges to leak away from the floating gate. The necessary quartz window in the plastic or ceramic package gives UV-erasable devices their unmistakable appearance but also \senders the package rather expensive.

UV-erasable devices are non-volatile and immediately live at power-up, thereby doing away \with the need for any kind of configuration-backup apparatus. Reprogramming necessitates removing the component from the circuit board and placing it into a special UV eraser, however, which is undesirable and often altogether impossible. This explains why EPROM based FPL devices much like the memories themselves have been superseded by parts that are more convenient to reconfigure. Electrically erasable memory. EEPROM technology borrows from UV-erasable memories. The difference is that the electrons trapped on the floating gate are removed electrically by having them tunnel through the oxide layer underneath the floating gate without exposure to ultraviolet light, thereby making it possible to manufacture FPL devices that are non-volatile but nevertheless reconfigurable through their package pins. The secret is a quantum-mechanical effect known as Fowler–Northeim tunneling that comes into play when a strong vertical field (8–10 MV/cm or so) is applied across the gate oxide[9]–[11].

Architectures of VLSI Circuits

Early electrically erasable devices were penalized by the fact that an EEPROM cell occupies about twice as much area as its UV-erasable counterpart because each bit cell includes a select transistor connected in series with the storage transistor. The flash memory technology prevalent today manages with a single floating-gate transistor per bit. The fact that erasure must occur in chunks, that is to say many bits at a time, is perfectly adequate in the context of FPL. Data retention times vary between 10 and 40 years. Endurance of flash FPL is typically specified with 100 to 1000 configure erase cycles, which is much less than for flash memory chips. Fuse or antiques. Fuses, which were used in earlier bipolar PROMs and SPLDs, are narrow bridges of conducting material that blow in a controlled fashion when a programming current is forced through.

Antifuses, such as those employed in today's FPGAs, are thin dielectrics separating two conducting layers that are made to rupture upon applying a programming voltage, thereby establishing a conductive path of low impedance. In either case, programming is permanent. Whether this is desirable or not depends on the application. Full factory testing prior to programming of one-time programmable links is impossible for obvious reasons. Special circuitry is incorporated to test the logic devices and routing tracks at the manufacturer before the unprogrammed devices are being shipped. On the other hand, antifuses are only about the size of a contact or via and, therefore, allow for higher densities than reprogrammable links. Antifuse-based FPL is also less sensitive to radiation effects, offers superior protection against unauthorised cloning, and does not need to be configured following power-up.

Non- Live at Reconfi- Unlimited Radiation Area Extra Configuration vola- power- gurable endu- tolerance occupation fair. Technology tile up rance of config. Per link steps SRAM no no in circuit yes poor large 0 EPROM yes yes out of no good small 3 circuit in array Electr. erasable yes yes in circuit good >5 EEPROM no 2·EPROM Flash memory no ≈EPROM Antifuse PROM yes yes no n.a. best small 3 Organization of hardware resources Simple programmable logic devices (SPLDs). Historically, FPL has evolved from purely combinational devices with just one or two programmable levels of logic such as ROMs, PALs, and PLAs. Flip-flops and local feedback paths were added later to allow for the construction of finite state machines. Products of this kind continue to be commercially available for glue logic applications. Classic SPLD examples include the 18P8 (combinational) and the 22V10 (sequential).

FIELD-PROGRAMMABLE LOGIC

Equivalent to one SPLD programmable interconnect CPLD c) AND plane OR plane PLA inputs outputs a) logic programmable AND plane OR plane SPLD flip-flops & feedback inputs outputs b) programmable feedback logic programmable evolution technological evolution technological flip-flops & feedback AND plane OR plane configurable I/O cell. General architecture of CPLDs (c) along with precursors (am).

The rigid two-level-logic-plus-register architecture in conjunction with the limited numbers of inputs, outputs, product terms, and flip-flops always restricted SPLDs to small applications. More scalable and flexible architectures had thus to be sought, and the spectacular progress of VLSI technology has made their implementation economically feasible from the late 1980s onwards. Two broad classes of hardware organization prevail today.

Complex programmable logic devices (CPLDs) expand the general idea behind SPLDs by providing many of them on a single chip. Up to hundreds of identical sub circuits, each of which conforms to a classic SPLD, are combined with a large programmable interconnect matrix or network. A difficulty with this type of organization is that a partitioning into a bunch of cooperating SPLDs has to be imposed artificially on any given computational task, which benefits neither hardware nor design efficiency.

Depending on the manufacturer, products are known as complex programmable logic device (CPLD), programmable large-scale integration (PLSI), erasable programmable logic device (EPLD), and the like in the commercial world. Field-programmable gate arrays (FPGAs) have their overall organization patterned after that of gate arrays[12]. Many configurable logic cells are arranged in a two-dimensional array with bundles of parallel wires in between. A switchbox is present wherever two wiring channels intersect.

Depending on the product, each logic cell can be configured so as to carry out some not-too-complex combinational operation, to store a bit or two, or both. While it is correct to think of alternating cells and wiring channels from a conceptual point of view, you will hardly be able to discern them under a microscope. The reason is that logic and wiring resources are superimposed for the sake of layout density in modern FPGA chips. Architectures of VLSI Circuits logic cell config. Switch box conf. configurable I/O cell wires FPGA wires. General architecture of FPGAs. As opposed to traditional gate arrays, it is the state of programmable links rather than fabrication masks that decides on logic functions and signal routing.

Parts with this organization are being promoted under names such as field-programmable gate array (FPGA), logic cell array (LCA), and programmable multilevel device (PMD). The number of configurable logic cells greatly varies between products, with typical figures

ranging between a few dozens and hundreds of thousands. FPGA architectures are differentiated further depending on the granularity and capabilities of the configurable logic cells employed.

One speaks of a fine-grained architecture when those cells are so simple that they are capable of implementing no more than a few logic gates and/or one bitable. As opposed to this, cells that are designed to implement combinational functions of four to six input variables and that are capable of storing two or more bits at a time are referred to as coarse-grained. The logic cell of fig.1.18b has 16 inputs and 11 outputs, and includes two programmable look-up tables (LUTs), two generic bistables that can be configured either into a latch or a flip-flop, a bunch of configurable multiplexers, a fast carry chain, plus other gates. Of course, the superior functional capabilities offered by a coarse-grained cell are accompanied by a larger area occupation. The gate-level netlists produced by automatic synthesis map more naturally onto fine-grained architectures.

The fact that fine-grained FPGAs and semi-custom ICs provide similar primitives further supports extensive reuse of design flows, HDL code, building blocks, and design 2 6 Incidentally note that FPL vendors refer to configurable logic cells by proprietary names. “Logic tile” is Actel’s term for their fine-grained cells whereas Xilinx uses the name “configurable logic block” (CLB) for their coarsegrained counterparts. Depending on the product family, one CLB consists of two or three LUTs plus two flip-flops or of several “slices”, each of which includes one LUT and one bistable. “Module” and “eCell”are commercial names used by other vendors.

FIELD-PROGRAMMABLE LOGIC

It thus becomes practical to move back and forth between field- and mask programmed circuits with little overhead and to postpone any final commitment until fairly late in the design cycle. Conversely, fine-grained FPGAs tend to be more wasteful in terms of configuration bits and routing resources. Another reason that contributed to the popularity of coarse-grained FPGAs is that on-chip RAMs come at little extra cost when that architectural concept is combined with configuration from static memory.

In fact, a reprogrammable LUT is nothing else than a tiny storage array. It is thus possible to bind together multiple logic cells in such a way as to make them act collectively like a larger RAM. As opposed to many other types of FPGAs, there is no compelling need to set aside special die areas for embedded SRAMs. In the occurrence of fig.1.18b, each of the two larger LUTs in each logic tile contributes another 16 bits of storage capacity. These are not the only features that distinguish the numerous commercial products from each other, however.

The next logical step was the extension to mixed-signal applications. Advanced products that combine configurable analogue building blocks with a micro- or digital signal processor and with analog-to-digital and digital-to-analog converters come quite close to the vision of field programmable systems on a chip. Vendors of field-programmable analogue and mixed-signal arrays include Anadigm, Actel, Cypress, Lattice, and Zetex FAS. Technical details on commercial FPL devices are distributed over thousands of datasheets, help to keep track of products and manufacturers. More condensed background information is available from references such as. Capacity figures of semi-custom ICs and FPL may be confusing.

As opposed to full-custom ICs, manufactured gates, usable gates, and actual gates are not the same. Manufactured gates indicate the total number of GEs that are physically present on a silicon die. A substantial fraction thereof is not usable in practice because the combinational functions in a given design do not fit into the available look-up tables exactly, because an

FPL device only rarely includes combinational and storage resources with the desired proportions, and because of limited interconnect resources. The percentage of usable gates thus depends on the application. The actual gate count, finally, tells how many GEs are indeed put to service by a given design.

In a deliberate attempt to make one product look better than its competitors in advertisements, product charts, and datasheets. Some FPL vendors prefer to specify the available resources using their own proprietary capacity units rather than in gate equivalents. Locate them in the Y-chart of Think of some industrial product family of your own liking (record player/MP3 player, mobile phone, (digital) camera, TV set/video recorder; car, locomotive, airplane; computer, photocopier, building control equipment, etc.)[13].

Discuss what microelectronics has contributed towards making these products possible in their present form. How has the microelectronic content evolved over the years? Where do you see challenges for improving these products and their microelectronic content? Choosing the hardware resources required to solve challenges arising from data and/or signal processing and planning how they interact to achieve marketing goal criteria are the two main concerns of VLSI architecture design. Getting the needed functionality properly is the main priority. The achievement of a predetermined performance goal, sometimes stated in terms of data throughput or operating rate, takes second place. Minimizing manufacturing expenses is a third goal that is this time of an economic nature. This entails lowering circuit size and increasing fabrication yield in order to produce the most functional components per processed wafer, assuming a certain fabrication technique. Energy efficiency is a common issue in VLSI design. The permissible power consumption is clearly severely constrained by battery-operated devices like mobile phones on the go, laptop computers, digital hearing aids, etc. The interest in energy efficiency when electricity is provided from the mains is maybe less obvious. The expense of cooling down high-performance, high-density ICs is the cause of this[14], [15]. Whereas the first example presents a challenge to the VLSI designer to fulfil a particular performance number at minimal power, the later situation seeks to maximize performance within a constrained power budget. Other highly desired attributes that are included here under the phrase agility include the flexibility to adapt changing demands and/or upgrade to future standards, as well as the quickness with which one can switch between one style of operation and another. Last but not least, two different architectures may vary in terms of the total technical work needed to fully develop them and, therefore, in the durations necessary to bring them to market.

CONCLUSION

The verification of data flow in System-on-Chip (SoC) designs is critical to ensure the correct operation of the various components of the SoC. Formal techniques, such as model checking and theorem proving, can be used to verify the correctness of the data transfer mechanisms between these components. Model checking involves the creation of a formal model of the design, which is used to explore the state space of the design and verify various properties, such as data integrity and deadlock freedom. Theorem proving involves the use of mathematical logic to prove the correctness of a design by creating a set of axioms and theorems that describe the data transfer mechanisms.

REFRENCES

[1] K. S. Asha, O. S. Mendonca, R. Bhandarkar, and R. Srinivas, “Data Flow Verification in SoC Using Formal Techniques,” in *Lecture Notes in Electrical Engineering*, 2020. doi: 10.1007/978-981-15-0626-0_2.

[2] P. Musinguzi, P. Ebanyat, J. S. Tenywa, T. A. Basamba, M. M. Tenywa, and D. Mubiru, "Precision of farmer-based fertility ratings and soil organic carbon for crop production on a Ferralsol," *Solid Earth*, 2015, doi: 10.5194/se-6-1063-2015.

[3] D. S. Vidya and M. Ramachandra, "An optimized method towards formal verification of mixed signals using differential fed neural network over FFNN," *Int. J. Electr. Comput. Eng.*, 2019, doi: 10.11591/ijece.v9i5.pp3423-3431.

[4] M. Driss, A. Aljehani, W. Boulila, H. Ghandorh, and M. Al-Sarem, "Servicing Your Requirements: An FCA and RCA-Driven Approach for Semantic Web Services Composition," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.2982592.

[5] R. Sinha, A. Girault, G. Goessler, and P. S. Roop, "A formal approach to incremental converter synthesis for system-on-chip design," *ACM Trans. Des. Autom. Electron. Syst.*, 2014, doi: 10.1145/2663344.

[6] A. Fotouhi *et al.*, "Lithium-Sulfur Cell Equivalent Circuit Network Model Parameterization and Sensitivity Analysis," *IEEE Trans. Veh. Technol.*, 2017, doi: 10.1109/TVT.2017.2678278.

[7] G. Tadesse *et al.*, "Active case detection of malaria in pregnancy using loop-mediated amplification (LAMP): A pilot outcomes study in South West Ethiopia," *Malar. J.*, 2020, doi: 10.1186/s12936-020-03380-9.

[8] N. Pandeya, G. M. Williams, A. C. Green, P. M. Webb, and D. C. Whiteman, "Do low control response rates always affect the findings? Assessments of smoking and obesity in two Australian case-control studies of cancer," *Aust. N. Z. J. Public Health*, 2009, doi: 10.1111/j.1753-6405.2009.00401.x.

[9] R. Van Os, "SOC-CMM: Designing and Evaluating a Tool for Measurement of Capability Maturity in Security Operations Centers," *Forum Incid. Response Secur. Teams*, 2016.

[10] Z. Lu and X. Zhao, "xMAS-based qos analysis methodology," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2018, doi: 10.1109/TCAD.2017.2706561.

[11] E. Singh, D. Lin, C. Barrett, and S. Mitra, "Logic Bug Detection and Localization Using Symbolic Quick Error Detection," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2018, doi: 10.1109/TCAD.2018.2834401.

[12] S. Kundu, S. Lerner, and R. Gupta, "High-level verification," *IPSJ Trans. Syst. LSI Des. Methodol.*, 2009, doi: 10.2197/ipsjtsldm.2.131.

[13] M. Luckey and G. Engels, "High-quality specification of self-adaptive software systems," in *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, 2013. doi: 10.1109/SEAMS.2013.6595501.

[14] P. Subramanyan, B. Y. Huang, Y. Vizel, A. Gupta, and S. Malik, "Template-Based Parameterized Synthesis of Uniform Instruction-Level Abstractions for SoC Verification," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2018, doi: 10.1109/TCAD.2017.2764482.

[15] D. P. Little, "Combinatorial aspects of the Lascoux-Schützenberger tree," *Advances in Mathematics*. 2003. doi: 10.1016/S0001-8708(02)00038-5.

CHAPTER 6

MITIGATING CYBER SECURITY RISKS WITH INTRUSION PREVENTION SYSTEMS AND INTELLECTUAL PROPERTY PROTECTION

Ms. Akshaya M Ganorkar, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id- akshayamganorkar@presidencyuniversity.in

ABSTRACT:

Intellectual Property (IP) is an asset that is created from the creative and innovative efforts of individuals or organizations. It can be protected under various forms of IP laws, such as patents, trademarks, copyrights, and trade secrets. IP rights provide exclusive ownership and legal protection to the owner, and they can be a valuable asset for businesses, organizations, and individuals. However, the unauthorized use or theft of IP can result in significant financial losses and damage to the reputation of the owner. Intrusion Prevention Systems (IPS) are security mechanisms that are used to protect IP and prevent unauthorized access, theft, and misuse of IP assets.

KEYWORDS:

Business, Intellectual System, IP laws, Intrusion Prevention, Mechanism.

INTRODUCTION

Verified, tested, and useable reusable modules have been created to shorten the design process to be utilised. Intellectual properties are the terms used to describe these components (IPs). Intellectual Property (IPs) is a catch-all phrase for a variety of legal rights. It is a kind of real estate it also includes intangible works of human creativity that are shielded from IP regulations.IP encompasses innovations, patents, service marks, and literary and creative works. Hidden information and other intangible assets. IP for semiconductors also falls under this category. Engineers spend less time creating each module from scratch when IPs are used. Since creating ICs costs a significant amount of time and money, it is crucial to safeguard these IPs from different dangers. Threats include me sending spoof emails, data that is not permitted. Tampering: unauthorized changes to the material. Repudiation: Refusal to acknowledge that certain activities were authorized. Knowledge disclosure: sniffing interfaces and inserting malicious code. Attack of service: locking the debug port and stopping the controller. Privilege elevation: get admission as anon-authorized features, developer [1], [2].

Among the most significant assaults against the SoC are malicious changes to a design known as assaults using hardware Trojans, theft of hardware IPs via illicit sales or the use of ICs or cores with soft intellectual property, or physical assaults such side-channel assault on data encryption during fault-based attack, scan-based attack. The SoC needs to be examined for various security flaws. As the beginning of a threat analysis, a functional overview should always be created, and every weak area I It is necessary to name the idea. The necessary assets are then added to the SoCto keep the secure IPs safe. The additional resources may

consist of I decryption process codes. Encryption keys are used to secure debug ports. Key codes: obtaining encrypted codes the ability to use extra features. Configuration information to enable a certain group of Characteristics of a product variation. Secure boot is a need for a secure SoC, since it ensures that harmful software the system with is loaded. Memory types safe and insecure. Cryptography and Support for decryption and Error checking port security[3]. Private buses are used in a secure IP architecture to transport components and keys. In a public bus, a normal key never is made accessible every crucial.

The design's parameters are kept in secure mind that will never be accessed the unlocked debug ports. Passwords are always required to access debug ports, and The SoC can only be debugged by the user and the engineer using a special password. All Before being scanned, the registers relating to private buses are cleared mode to guarantee that no sensitive information is accessed through to the development ports. Verifying Data Flow in SoC Using Formal Techniques SoC Validation Implementation, manufacturing, and testing typically come after the design phase in an IC design and testing. If a design fault materializes at a later stage, to correct the design, additional time and work will be needed. Additionally, any security flaws.

Unintentional functional routes, crucial parameters, or data may exist, similar to backdoors.be forfeited to frauds. Consequently, design verification is carried out to record the design. Early stages of the design and testing to eliminate design flaws before they are implemented. Functional verification is done to ensure that the functionality is accurate [4]. Immediately after the design process. Where its plans are officially validated, formal verification procedures are more beneficial. Predictive analysis is provided via verification.to guarantee that the synthesized design will carry out the specified function when builtI/O operation it is in charge of the design's quality. Hardware engineers have developed a plethora of quite distinct architectural ideas over the years as a result of divergent applications and goals, which we will attempt to put into perspective in this chapter.

The antipodes of architecture

One generally has two options when faced with a computational task: either write programme code and execute it on a machine that can be programmed, such a microprocessor or a digital signal processor (DSP), or create a hardwired electrical circuit that performs the required computing processes. A systems engineer is forced to choose between two options as a result of this fundamental dichotomy: a) choose a processor-type general-purpose architecture and write programme code for it, or b) design a dedicated hardware architecture specifically for the required computational needs. Before starting the construction of a complicated circuit, it is important to choose between a general-purpose processor and an architecture tailored to the current application [5].

DISCUSSION

Commercial microprocessors have the tremendous benefit of allowing engineers to immediately concentrate on higher-level concerns like functionality and system-level architecture. They don't need to take care of all those difficult tasks that semi-custom and, much more so, full-custom design must deal with. Moreover, masks made specifically for each user are not required. Using commercial instruction-set processors and/or FPL avoids a number of technical problems that would otherwise need a lot of attention if a bespoke IC

were to be created. Instead, it is feasible to maximise speed and energy efficiency thanks to the payload calculations' primary emphasis, the lack of programming and setup overhead, and the complete control over all aspects of architecture, circuit design, and layout [6] .

Types of IPS

Intrusion Prevention Systems (IPS) are security mechanisms that are used to protect computer networks and systems from unauthorized access, theft, and misuse. There are several types of IPS that can be used to protect IP assets, including:

1. **Network-based IPS:** Network-based IPS monitors the traffic flowing over the network and identifies and blocks any suspicious or unauthorized activity. It uses various techniques, such as signature-based detection, anomaly detection, and behavioral analysis to identify and prevent intrusions.
2. **Host-based IPS:** Host-based IPS operates at the operating system or application level and monitors the behavior of individual hosts or devices. It can identify and prevent unauthorized access or activity on the host or device.
3. **Wireless IPS:** Wireless IPS is used to protect wireless networks and can identify and prevent unauthorized access, rogue access points, and other wireless security threats.
4. **Cloud-based IPS:** Cloud-based IPS is a security mechanism that is used to protect cloud-based applications and data. It can identify and prevent unauthorized access, data theft, and other security threats in the cloud environment.
5. **Physical IPS:** Physical IPS is used to protect physical assets, such as buildings, equipment, and devices. It can include physical security mechanisms, such as access controls, surveillance systems, and alarms, to protect physical assets from theft or unauthorized access.

Importance of IP Protection

IP protection is important for several reasons, including:

1. **Economic Value:** IP assets can have significant economic value for businesses and individuals. Protecting IP can ensure that the owner has exclusive ownership and control over their IP assets, and can prevent unauthorized use or theft that can result in significant financial losses.
2. **Competitive Advantage:** IP assets can provide a competitive advantage to businesses and individuals. Protecting IP can ensure that the owner maintains their competitive advantage and prevents competitors from using their IP assets to gain an advantage.
3. **Reputation:** IP assets can also be important for the reputation of businesses and individuals. Protecting IP can ensure that the owner's reputation is not damaged by unauthorized use or misuse of their IP assets.
4. **Legal Protection:** IP protection is also important for legal reasons. IP laws provide legal protection to the owner of the IP asset and can be used to enforce legal rights against unauthorized use or misuse of the IP asset.

Methods and Strategies for Protecting IP Assets

There are several methods and strategies that can be used to protect IP assets, including:

1. **Patents:** Patents are a form of legal protection for inventions and innovative ideas. They provide the owner with exclusive ownership and the right to prevent others from making, using, selling, or importing the invention without permission.
2. **Trademarks:** Trademarks are a form of legal protection for logos, brand names, and other distinctive marks that are used to identify a business or product. They provide the owner with exclusive ownership and the right to prevent others from using similar marks that may cause confusion among consumers [7].

These are circuit instances where specialised architectures outperform computers with instruction sets. As an example, future chapters will cover topics like power distribution, clock preparation and distribution, input/output design, physical design and verification, signal integrity, electrical overstress prevention, wafer testing, and packaging selection. In addition to estimating sales volume, hitting a small window of opportunity, finding the proper partners, and providing the required resources, in-house experience, and investments, setting up a functional CAE/CAD design pipeline is often a significant roadblock. But keep in mind that many of these problems may be avoided by developers by using field-programmable logic (FPL). Architecture instruction set processor, a von Neumann or Harvard-style dedicated design not a single recognised pattern Execution models that are "instruction-oriented" and "dataflow-oriented" retrieve data, load it, process it, and then store it. Datapath universal operations, limited operations only, and specialised ALU(s) and memory design.

Usually hardwired controller with programme microcode Analytical benchmarking of different programmes may be used to predict performance in terms of instructions per second, data throughput, and indicator run time. Division of labour in a factory manufacturing functioning in accordance with setup for efficient production of various plans each day of a few closely linked items is based on a paradigm from a craftsman in his machine shop potential hardware norm ASICs with on-chip FPL (FPGA|CPLD) or C|DSP components, or ASICs with specific architecture implementations largely software design in engineering, mostly hardware design. Strengths Lean circuitry, regular design flow, maximum performance space, fast availability, and great energy efficiency low initial costs. In certain cases, specialised designs outperform even the greatest commercially available general-purpose processors in terms of performance and/or energy dissipated, while in other cases, they prove to be a terrible waste of hardware and technical resources.

For specialised architectures, algorithms that are extremely erratic, data-dependent, and memory-intensive are inappropriate. Such circumstances may be encountered in electronic data processing applications, accounting, and reactive systems. user interfaces, industrial controls, and three When a system continually interacts with its surroundings at a pace dictated by that environment, it is said to be reactive . The system deals with events, and mathematical formalisms used to describe them try to capture the intricate ordering and causality relationships between events that could happen at the inputs and the associated reactions events themselves at the outputs. Elevators, protocol handlers, anti-lock brakes, process controllers, graphical user interfaces, and operating systems are a few examples[8], [9]. Figure 1 illustrate the Public vs. Private IP Addresses.

Figure 1: Illustrate the Public vs. Private IP Addresses.

In contrast, a transformatorial system takes fresh input values, often at regular intervals, processes them to produce new values, and then stops until fresh data items are received. The system's primary focus is on processing data values using logic and arithmetic. The numerical connections between the many data items involved are captured by formalisms for modelling transformatorial systems. Examples include payroll software, filtering, data compression, ciphering, pattern recognition, and other programmes that are referred to as number crunching informally. Control in the sense of "programmierte Steuerungen" rather than "Regelungstechnik," as in German.

THE ARCHITECTURAL ANTIPODES

A dedicated (special-purpose) hardware structure and a program-controlled general-purpose processor as architectural antipodes one will inevitably find hardware structures modelled after instruction set processors while searching for the best designs for these kinds of applications. It is more efficient and cost-effective to write the code for a typical microcomputer in this scenario, whether it is purchased as a physical component, included as a megacell in an ASIC, or exists just as a virtual component.

There is much more flexibility for developing specialised architectures in situations where data streams must be handled in reasonably predictable ways. In real-time applications from digital signal processing and telecommunications, such as Source coding (i.e. data, audio, and video (de)compression), (De)ciphering (primarily for secret key cyphers), Channel coding (i.e. error correction), and Digital (de)modulation, situations that favour dedicated architectures are frequently encountered (for modems, wireless communication, and disc drives),

specialised unit subtask B is devoted to specialised unit subtask A is dedicated to specialised unit subtask C is dedicated to specialised unit subtask D is dedicated to specialised unit subtask E Dedicated to a specialised unit subtask (D b) C input data output data Program storage controller data memory general output data input data program-controlled processor input data output data are examples of programme storage controller data memory.

DSPs have word lengths of around 32 bits and work best for prolonged multiply-accumulate operations. DSPs cannot utilise these resources, however, since the Viterbi algorithm may be configured to not require multiplication and to work with word lengths of 6 bits or less. Contrarily, the Viterbi algorithm's inherent parallelism may be taken advantage of via a pipeline of specially designed stages that are optimised for branch metric calculation, path metric updating, and survivor route traceback operations. By sacrificing throughput for the amount of computing units at each step, different throughput demands may be met. A second coprocessor is included in more complex DSPs, such the C6455, to speed up route metric updating and survivor traceback[10], [11].

Comparing several architectural options for a secret-key block encryption/decryption technique. (IDEA cypher, block size 64 bit, key length 128 bit). Due to the high degree of parallelism in its datapath and, in particular, the existence of four pipelined computational units for multiplication modulo (216 + 1) constructed in a full-custom configuration that work simultaneously and constantly, the VINCI ASIC clearly has an advantage. The more modern IDEA kernel combines four highly efficient arithmetic units with a deep submicron manufacturing technique. The necessity for a fully customised layout to attain higher performance has vanished. Contrasts the architectural alternatives to the LempelZiv-77 method, which mainly depends on string matching procedures, for lossless data compression. A reconfigurable coprocessor board constructed around four field-programmable gate-array components houses the specialised hardware architecture. The string comparison subfunctions are created to be performed concurrently by 512 special-purpose processing components. As the content-addressed symbol memory is effectively set up as a shift register, all entries may be accessed simultaneously. Naturally, the two software implementations produced by compiling C source code are unable to provide as much concurrency.

CONCLUSION

Intrusion Prevention Systems (IPS) are security mechanisms that are used to protect intellectual property (IP) and prevent unauthorized access, theft, and misuse of IP assets. IPS can include network-based, host-based, wireless, cloud-based, and physical IPS, which can monitor and identify suspicious activity and prevent intrusions. The protection of IP assets is crucial for businesses and individuals to maintain their economic value, competitive advantage, reputation, and legal protection. Methods and strategies for protecting IP assets can include patents, trademarks, copyrights, trade secrets, contracts, and employee training.

REFERENCES

[1] A. Basak, S. Bhunia, T. Tkacik, and S. Ray, “Security Assurance for System-on-Chip Designs with Untrusted IPs,” *IEEE Trans. Inf. Forensics Secur.*, 2017, doi: 10.1109/TIFS.2017.2658544.

[2] T. Vian, F. G. Feeley, S. Domente, A. Negruta, A. Matei, and J. Habicht, “Barriers to universal health coverage in Republic of Moldova: A policy analysis of formal and informal out-of-pocket payments,” *BMC Health Serv. Res.*, 2015, doi: 10.1186/s12913-015-0984-z.

[3] J. Zhang, Y. Lin, Y. Lyu, and G. Qu, “A PUF-FSM Binding Scheme for FPGA IP Protection and Pay-Per-Device Licensing,” *IEEE Trans. Inf. Forensics Secur.*, 2015, doi: 10.1109/TIFS.2015.2400413.

[4] Z. Huang and Q. Wang, “Enhancing architecture-level security of soc designs via the distributed security IPs deployment methodology,” *J. Inf. Sci. Eng.*, 2020, doi: 10.6688/JISE.202003_36(2).0016.

[5] R. S. Chakraborty and S. Bhunia, “HARPOON: An obfuscation-based SoC design methodology for hardware protection,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2009, doi: 10.1109/TCAD.2009.2028166.

[6] M. Onyango *et al.*, “O12.2 Understanding the relationship dynamics between female sex workers and their intimate partners in kumasi, ghana,” *Sex. Transm. Infect.*, 2015, doi: 10.1136/sextrans-2015-052270.143.

[7] A. Alanwar, M. A. Aboelnaga, Y. Alkabani, M. W. El-Kharashi, and H. Bedour, “Dynamic FPGA detection and protection of hardware trojan: A comparative analysis,” *Int. J. Comput. Digit. Syst.*, 2016, doi: 10.12785/IJCDS/050302.

[8] A. Ibrahim, C. Valli, I. McAteer, and J. Chaudhry, “A security review of local government using NIST CSF: a case study,” *J. Supercomput.*, 2018, doi: 10.1007/s11227-018-2479-2.

[9] E. N. Ceesay, K. Myers, and P. A. Watters, “Human-centered strategies for cyber-physical systems security,” *ICST Trans. Secur. Saf.*, 2018, doi: 10.4108/eai.15-5-2018.154773.

[10] C. R. G. Popescu and G. N. Popescu, “Risks of cyber attacks on financial audit activity,” *Audit Financ.*, 2018, doi: 10.20869/auditf/2018/149/140.

[11] K. B. Kelarestaghi, K. Heaslip, M. Khalilikhah, A. Fuentes, and V. Fessmann, “Intelligent Transportation System Security: Hacked Message Signs,” *SAE Int. J. Transp. Cybersecurity Priv.*, 2018, doi: 10.4271/11-01-02-0004.

CHAPTER 7

VERY LARGE-SCALE INTEGRATION VLSI SYSTEM

Dr. Sivaperumal S, Professor, Director
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id- sivaperumal@presidencyuniversity.in

ABSTRACT:

Very Large-Scale Integration (VLSI) is a technology that involves the integration of a large number of transistors and other electronic components onto a single chip. VLSI has revolutionized the electronics industry by making it possible to create complex integrated circuits (ICs) that are smaller, faster, and more power-efficient than their predecessors.

KEYWORDS:

Complex Hardware, Electronic Component, Greater Functionality, VLSI, Integrated Circuit.

INTRODUCTION

Greater functionality may be attained with less complex hardware. The cost of providing extra features is thereby lowered to the expenditure of the hardware required for storing the logic design since the necessary logic may be kept in memory. This is especially helpful in the field of mobile communication since it is simple to change the protocol to a newer one, save it in memory, and then modify the hardware to get the desired functionality. The advantages of higher speed and decreased energy and power use are compelling. Moving key software loops to reconfiguring hardware, according to a research, resulted in estimated efficiency improvements of 35% to 70% and average speedups of 3 to 7 times, depending on the exact device being utilized.

Embedded Quali-ties

In general-purpose computing, a standard silicon chip could be manufactured and modified to do any computational job. This meant that a single IC could be produced for a variety of applications to share underlying economic principles, and one IC could have been used to various challenges at various times. Engineers may programme the component to do tasks that the original IC makers had not even thought of thanks to general-purpose computing. Some of the machines the the government is putting. Reconfigurable computing has all of these "general-purpose" qualities. Reconfigurable computers calculate a purpose by designing functional components and wiring them together in space, as opposed to processing through some kind of set of procedures in time (like a processor). As with a customised ASIC, this enables the parallel processing of certain, specified processes. Additionally, it may be altered. To improve performance, the changeable hardware fabric may be swiftly and easily adjusted from a distance. It may be changed to serve a totally new purpose. As a result, reconfigurable computing has cheaper non-recurring engineering (NRE) expenses than a bespoke ASIC [1].

Decreased System Cost

By reducing the ASIC design reduced system cost for a low-volume device is obtained. The cost of manufacturing fixed hardware is really significantly cheaper for greater volume items.

Technical instability raises system costs for ASIC and broad sense embedded systems. Systems that can be upgraded and reconfigured have a longer usable lifespan. This lowers overall expenses.

Shorter Time to Market

Reconfigurable computing's last benefit is a shorter time to market. Reconfigurable computing requires far less development work now that ASIC is not utilised. Even though the product has been delivered, the logic design is still adaptable. Minimum criteria may be used to launch a design, and afterwards, other functionalities can be implemented without altering the actual product or system. Reconfigurable computing enables incremental design flow as a result.

These benefits make reconfigurable computers effective tools for several applications. Applications for advanced electronic systems like ASICs and system is directly board's development and development tools. There are no model tools for these systems. Additionally, creating prototypes takes time and money. Electronic design verification may be performed quickly, cheaply, and accurately using a reconfigurable computer [2].

Disadvantages

Reconfigurable computing has two significant drawbacks that may be seen. They are the length of time it takes a chip to adapt itself to a job and the challenge of programming such processors. Dynamic reconfigurable computing faces a number of intricate problems. They are development tools, design space, location, routing, timing, and consistency. The discussion of each of them follows.

Placing Problems

Having enough room to position additional hardware is necessary in order to modify it. If a component must be positioned close to certain resources like built-in memory, I/O pins, or DLLs on the FPGA, the placement problem gets more complicated.

Routing Problems

The newly reconfigured components must be attached to the existing components. New components must be able to interface via the ports. The prior arrangement must have used the same ports as well. The components' orientation should be in a practical way to do this.

Timing Problems

The time requirement must be met by newly configured hardware for the circuit to operate effectively. Timing may be impacted by longer cables connecting components. After the gadget is dynamically reconfigured, maximum speed should be possible. A new additional design that is timed incorrectly or too soon might provide an incorrect outcome [3], [4].

Continuity Problems

Device reconfiguration, whether static or dynamic, shouldn't compromise the computational consistency of the design. When the FPGA is partly redesigned and interfaced with an existing design, this problem becomes crucial. The present design of the device shouldn't be

erased or changed when new components are added using reconfigurable fabric. (Or recall). There should be several safe techniques to save the serial data to the memory.

Tools for Development

Commercial dynamic reconfigurable computing development tools are currently in the early stages of development. A bottleneck in web electronics is still the absence of commercially accessible tools for the definition to execution phases. To implement the whole system using the existing technologies, significant human involvement is needed.

Table 2.5 contrasts several architectures for a secret-key block encryption/decryption technique (AES cipher, block size 128 bit, key length 128 bit). The three hardware implementations incorporate several look-up tables (LUTs) for implementing the so-called S-Box function, which is heavily used by the Rijndael algorithm. Additionally, distinct hardware units simultaneously perform (de)ciphering and subkey preparation. In light of this, the assembly language program's throughput on a Pentium III is quite astounding [5] .

The Rijndael algorithm was created with the Pentium architecture in mind, which primarily explains this (MMX instructions, LUTs that fit into cache memory, etc.). Yet power dissipation still poses a challenge. After transmission across copper lines and optical fibres, there is adaptive channel equalisation and filtering (for noise cancellation, preprocessing, spectral shaping, etc.), Digital beamforming using phased-array antennas (Radar), multipath combiners in broadband wireless access networks (RAKE, MIMO), computer graphics and video rendering,

Transcoding, such as between different multimedia formats, multi-media (such as MPEG, HDTV), packet switching (such as ATM, IP), pattern recognition, and more. Processor algorithms and hardware designs are closely connected. Although dedicated architectures perform orders of magnitude better than program-controlled processors in many applications that are primarily transformatorial in nature, they are unable to compete with the agility and efficiency of processor-type designs in others that are more reactive.

Hardware costs are not the same as software costs. Software activities that require sequential execution, such permutations of bits inside a data word, take a long time to complete. In hardware, they are reduced to straightforward wires that cross as they travel between subcircuits. On the other hand, look-up tables (LUTs) of nearly any size have developed into a plentiful and affordable resource in every microcomputer, while big on-chip RAMs and ROMs often use a significant chunk of the timing and space budgets of ASIC designs.

DISCUSSION

We have compiled 10 criteria that an information processing algorithm should ideally satisfy in order to justify the construction of a special-purpose VLSI architecture and to fully use the technology in an effort to provide some direction. Naturally, relatively few algorithms in the actual world meet all of the above criteria. Yet, it is reasonable to assume that when too many of these requirements are broken, developing a specialised architecture that can beat a general-purpose processor in terms of performance and price would prove challenging. The qualities that are most desired are listed first, followed by a list of their relative importance [6].

Loose connection between the main activities of processing. It is possible to break down the total data processing into smaller, more manageable jobs. At this stage, it doesn't matter whether those activities are to be completed sequentially or simultaneously; what matters is that each task has a clear functional specification and there is reasonable interaction between them. Otherwise, functional verification, reuse, optimization, and design of the architecture become true nightmares.

Thinking Machines at the Top Digital dedicated high-performance VLSI designs towards high data rates low-cost hardware towards applications towards sophisticated high audio & video quality, (strong compression, secure connection, resilient transmission), multi-antenna (MIMO) wireless reception (4G), signal processors, and Intel Xeon (2006) BlueGene/L (Supercomputer).

Computing requirements for various signal and data processing applications (rough estimates; exact meaning of operation and data item left unspecified; 16 bit-by-16 bit multiply-accumulate (MAC) operations on 16 bit samples are frequently thought of as typical in a context of digital signal processing).

a) Since the number of iterations for each loop is predetermined and constant, the operation's path is not too dependent on the data being processed.
b) The programme does not request that calculations be performed using an excessive number of different types, modes of operation, data formats, unique parameter settings, etc.

A straightforward control flow has two advantages. One benefit is that it is feasible to plan the chip's architecture by anticipating the datapath resources needed to achieve a certain performance objective. In calculating the computing load or sizing data memories and the like, statistical approaches are not required. Moreover, datapath control may to put it another way, the target algorithm has almost no loops and branchings that include condition clauses that involve data items, such as if...then[...else], while...do, and repeat...until. Counters and straightforward finite state machines (FSMs) are used in VLSI circuit architectures because they are tiny, quick, energy-efficient, and, most importantly, simple to test. Figure 1 illustrate the Very Large Scale Integration System.

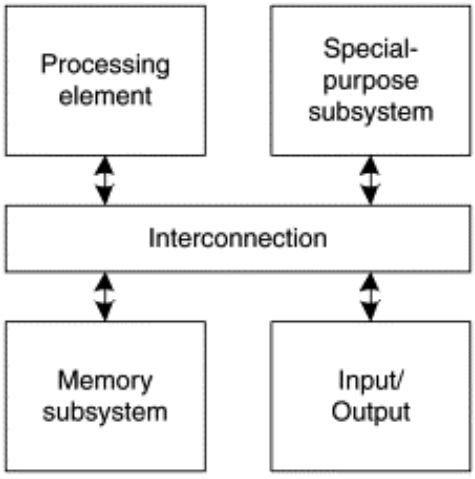


Figure 1: illustrate the Very Large Scale Integration System.

On the other hand, a processor-type architecture that runs under supervision of stored microcode supports an unduly complex course of operations that requires a lot of datadependent branching, multitasking, and the like. The majority of control activities will subsequently be converted into a series of machine instructions that run over the course of multiple clock cycles.

Consistent data flow. There are no computationally costly processes that are called only sometimes; instead, the processing of the data is based on a very limited number of repeated, similar actions that occur often. By using strategies like iterative decomposition and timesharing, regularity makes it possible to efficiently share hardware resources.

On the other hand, parallel functional units may handle many data streams concurrently when they must be treated uniformly. Although the different functional units may be designed to exchange data across fixed local lines, a regular data flow also aids in reducing communications overhead in terms of both area and interconnect latency. Last but not least, regularity makes reuse easier and requires less work for design and verification [7], [8].

In contrast, operations that are performed infrequently will either require dedicated functional units that must be constrained to be idle for the majority of the time, which increases chip size, or they will have to be divided into a series of substeps to be executed one after the other on a general-purpose datapath, which is slow. Long and flexible communication buses are needed for irregular data flow, which compromises layout density, operation speed, and energy efficiency. The total amount of storage needed is low and has a set maximum. 7 Since they cannot be economically integrated into ASICs, memories that take up too much chip space, say more than half or so, must be implemented off-chip using ordinary components. Dedicated architectures sometimes suffer from high storage needs combined with light computational demands.

5. Compatible with arithmetic of finite precision. Effects from arithmetic with limited precision have no influence on the procedure. That is, the individual computing steps may be performed with reasonably short word widths, such as 16 bits or fewer, without the requirement for floating-point arithmetic. Conventional microprocessors and DSPs are sold with datapaths that are fixed and often generous in width (24, 32, 64 bit, or even floating-point). Unless the programmer has to use multiple precision arithmetic, there are no additional charges.

In contrast, ASICs and FPL provide the option to adjust the word lengths of datapaths and on-chip storage to the specific computational demands of the area. This is significant because word width tends to increase the load at an excessive pace due to factors such as circuit size, logic latency, connection length, parasitic capacitances, and energy dissipation from addition, multiplication, and other operations. This makes using dynamic data structures impossible. 8 Since they are often manually optimised at the transistor level (e.g., dynamic logic) and implemented in tiled layout rather being constructed from conventional cells, processor datapaths are frequently quick and space-efficient. ASIC designers seldom ever have these possibilities[9], [10].

THE ARCHITECTURAL ANTIPODES

Nonrecursive processing that is linearly time-invariant. A nonrecursive linear time-invariant system over some algebraic field is described by the processing algorithm. Each of these

characteristics may be used to reorganise data processing in one way or another. Absence of transcendental operations. Roots, logarithmic, exponential, or trigonometric functions, arbitrary coordinate conversions, translations between incompatible number systems, and other transcendental functions are not used in the algorithm because they must either be stored in large look-up tables (LUT) or be calculated online in protracted, frequently irregular computation sequences. Such functions may be implemented more cheaply if the accuracy requirements are acceptable and allow for approximation by lookups from tables of a suitable size, potentially followed by interpolation.

8. Widespread use of data operations not supported by common instruction sets. In fact, there are numerous processing methods that need expensive arithmetic/logic operations in order to function. In situations where such tasks require the assembly of several instructions, it is often feasible to outperform conventional program-controlled computers. The same calculation may then be done more effectively by creating dedicated datapaths. Examples include add-compare-select operations, complex-valued arithmetic, and several ciphering processes.

The ability to conduct some kind of preprocessing is another benefit when some of the parameters are constants. For example, multiplying by a variable is more difficult than multiplying by a constant. Throughput, not delay, is important. No divisions or multiplications on extremely broad data words, which is a necessary condition for pipelined processing. There is no usage of wide-argument multiplications. The method uses addition and subtraction a lot more often than multiplications and much less frequently than divisions since their VLSI implementation is substantially more costly when the data words involved are broad. There is a lot of space between these two opposing architectural poles.

The majority of markets want simultaneous performance, agility, low power, and moderate design effort. With these incompatible demands, it is extremely desirable to combine the throughput and the 9 Section 2.7 is where recursiveness is to be defined. The term "linear" is intended to suggest the superpositional principles of "f (x(t) + y(t))" and "f (c x(t)" and "cf (x(t)". If z (t) = f (x(t)) is the response to x(t), then z (t T) is the response to x(t T), which indicates that the only consequence of delaying the input is a delaying of the output by the same amount of time. In section 2.11, fields and other algebraic structures are contrasted. 0 Dropping unit factors and/or zero sum terms (both at word and bit levels), replacing integer powers of 2 as arguments in multiplications and divisions, omitting insignificant contributions, special number representation schemes, utilising symmetries, precomputed look-up tables, and distributed arithmetic are just a few common measures that may help to reduce the computational burden in scenarios where parts of the arguments are known.

54 VLSI circuit architectures combine the ease and flexibility of an instruction set processor with the energy efficiency of a specialised VLSI design for complex yet highly repetitive calculations. This is so that a system's most computationally intensive components—and vice versa—are not often the ones that are susceptible to frequent modification. The discovery instantly points to a configuration in which a software-controlled microprocessor works in conjunction with one or more specialised hardware components. It is feasible to completely devote the different functional units to their respective duties and to optimise them in line with those tasks when the pursuit of computing efficiency and that of agility are separated. The function of the instruction set microcomputer might take on a variety of forms.

Subfunctions with irregular control flow, recurring control flow, and application-specific demand for flexibility and computer efficiency are the main characteristics of such subfunctions.

Track searching, 16-to-8 bit demodulation, spindle and tray management, error correction, and handling of non-video data are all features of DVD players. Directory, author, discrete cosine transform, subtitles, and region codes for MPEG-2 decompression processing of video signals

SMS, intermediate frequency directory management, filtering, (de)modulation, battery monitoring, channel (de)coding, communication protocol, error correction (de)coding, channel allocation, (de)ciphering, roaming, accounting, and speech (de)compression are all terms used in relation to mobile phones.Target acquisition, picture segmentation, triggering of operations, feature extraction, object tracking, image stabilisation as part of a defensive missile, and redundancy reduction are all examples of pattern recognition techniques.

THE ARCHITECTURAL ANTIPODES

Three program-controlled processing units and one specialised processing unit are placed in a chain. Each unit completes its data processing task and transmits the finished product to the subsequent unit. If everything is hardwired and custom-made, this structure can support a lot of variance while still providing plenty of opportunities for speed optimization. The overall architecture becomes more versatile while maintaining a relatively low overhead in terms of circuit complexity and energy dissipation when the specialised hardware units support a limited degree of parametrization (e.g. wrt data word width, filter order, code rate, data exchange protocol, and the like). To describe the concept, the phrase "weakly programmable satellites" was created. This expansion of the initial idea is suggested by an optional parametrization bus and is focused on specialised unit subtasks. B is devoted to a specialty unit subtask an assigned, specialised unit job D bus for parametrization

a) Manages C program-controlled processor output data input data.
b) Dedicated to Specialized Unit Subtask B an assigned, specialised unit job D inputs data, outputs data, exchanges data, and controls bus-programmed processors manage subtask C and data dispatch.

While is also based on segregation, the way the various components interact is different. Now, a software-programmable host is in charge of controlling all specialised hardware components. The freedom for back-and-forth data and control word transfers is provided by a bidirectional bus. There is a rather small range of instructions that any coprocessor, or assistance engine as it is often known, may take. Up until a set of input data and a start instruction are given, it is idle. As an alternative, the data might be continuously stored in the host's memory and accessible by the coprocessor via direct memory access (DMA). The coprocessor sets a status flag and/or sends an interrupt signal to the host machine once local processing is finished. The host then accepts the data that has been processed and handles the next steps .

CONCLUSION

Very Large-Scale Integration (VLSI) technology has revolutionized the electronics industry by enabling the integration of a large number of transistors and electronic components onto a

single chip. VLSI technology has enabled the creation of various electronic devices that are smaller, faster, and more power-efficient than their predecessors, and have found applications in various industries, including computing, telecommunications, automotive, and healthcare. VLSI technology has transformed the electronics industry, and its future prospects look bright. The continued development of VLSI technology and the creation of even more complex and efficient ICs will undoubtedly play a crucial role in shaping the future of electronics and advancing technological innovation in various fields.

REFERENCES

[1] "2020 Index IEEE Transactions on Very Large Scale Integration (VLSI) Systems Vol. 28," *IEEE Trans. Very Large Scale Integr. Syst.*, 2020, doi: 10.1109/tvlsi.2020.3041879.

[2] M. Alioto, "Editorial on the Opening of the New Editorial Year - The State of the IEEE Transactions on Very Large Scale Integration (VLSI) Systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2020. doi: 10.1109/TVLSI.2019.2955524.

[3] "2019 Index IEEE Transactions on Very Large Scale Integration (VLSI) Systems Vol. 27," *IEEE Trans. Very Large Scale Integr. Syst.*, 2019, doi: 10.1109/tvlsi.2019.2956852.

[4] K. Chakrabarty, "IEEE Transactions on Very Large Scale Integration (VLSI) Systems: Editorial," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2015. doi: 10.1109/TVLSI.2014.2385112.

[5] Y. I. Ismail, "IEEE Transactions on Very Large Scale Integration (VLSI) Systems: Editorial," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2011. doi: 10.1109/TVLSI.2011.2109790.

[6] "2012 Index IEEE Transactions on Very Large Scale Integration (VLSI) Systems Vol. 20," *IEEE Trans. Very Large Scale Integr. Syst.*, 2012, doi: 10.1109/tvlsi.2012.2212496.

[7] M. Kazemi and M. F. Bocko, "Design rules for scalability in spin-orbit electronics," *Sci. Rep.*, 2019, doi: 10.1038/s41598-019-49831-5.

[8] "IEEE Transactions on Very Large Scale Integration (VLSI) Systems society information," *IEEE Trans. Very Large Scale Integr. Syst.*, 2014, doi: 10.1109/tvlsi.2014.2306241.

[9] "2017 Index IEEE Transactions on Very Large Scale Integration (VLSI) Systems Vol. 25," *IEEE Trans. Very Large Scale Integr. Syst.*, 2018, doi: 10.1109/tvlsi.2018.2790008.

[10] "Search for Editor-In-Chief of the IEEE Transactions on Very Large Scale INtegration (VLSI) Systems," *IEEE Trans. Very Large Scale Integr. Syst.*, 2018, doi: 10.1109/tvlsi.2018.2813500.

CHAPTER 8

METAL–OXIDE–SEMICONDUCTOR FIELD-EFFECT TRANSISTOR WITH A VACUUM CHANNEL

Dr. S Riyaz Ahammed, Assistant Professor
Department of Electronics and Communication Engineering, Presidency University, Bangalore, India
Email Id- riyaz.ahammed@presidencyuniversity.in

ABSTRACT:

The Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) is a fundamental building block of modern electronics, used in a wide range of applications from microprocessors to power electronics. Recently, a new type of MOSFET has been proposed that uses a vacuum channel instead of a traditional semiconductor channel. The channel region is replaced with a vacuum, which eliminates many of the performance limitations of conventional MOSFETs. Because the electrons in the vacuum are not scattered by impurities or defects in the channel material, the vacuum MOSFET (VMOSFET) is predicted to have faster switching times and higher power handling capabilities than traditional MOSFETs.

KEYWORDS:

Electronics, Microprocessors, Metal-Oxide, Transistors, Vacuum Channel.

INTRODUCTION

A potential well with a width of 2 nm formed on the semiconductor side may maintain a quasi-two-dimensional electron system (2DES) in the metal-oxide-semiconductor (MOS) capacitor structure. Additionally, the metal side undergoes band stretching, allowing positive and negative charges to coexist in a small area nm at the oxide layer's interface. Now imagine a MOS structure with a limited lateral extent, for instance one with cleaved edges. Strong Columbic repulsion is anticipated in the local region surrounding the edge of 2DES in the scenario where prosecute neutrality is maintained by relatively distant indictments for example, opposite magnetic pole charges prompted throughout the oxide layer of a Reverse bias, and this can dramatically alter the electrostatic potential there.

Focused ion beam (FIB) etching of a silicon MOS structure (20 nm aluminum/23 nm SiO2/p- (or n-) silicon substrate; was utilised to create nanoscale void-channels. Square wells (0.5 0.5 mm2, 1 1 mm2, and 2 2 mm2) were etched to a depth of 1 or 2 mm. The oxide layer thickness in this vertical construction accurately determines the channel length between the anode and cathode, which was intended to be less than the mean free path of air (60 nm). In the log-log plots, the two-terminal I–V characteristics exhibit a rectifying behaviour with a forward slope of 1.5 and a reverse slope of 0.5–1.0. In the p-Si (n-MOS) example, when such aluminium passageway is negatively biassed, the channel exhibits a forward characteristic. This suggests that with the same bias voltage but opposite polarity, charged particle emission from the aluminum side is more effective than that of the silicon side. For instance, a 43 nA channel current is reported at 1 V bias in a 0.5 0.5 1.0 mm3 well created on p-Si, whereas 6 nA is achieved at -1 V sensitivity [1]–[3].

The stream throughput is related to the edge of the well not at all the width of the well according to a study of three experiments with wells of varied diameters (perimeters of 2, 4, or 8 mm). This implies that electron emission takes place at a well's edge surface, or peripheral, on its vertical sidewalls. However, samples with varied etch depths (1 or 2 mm) exhibit the same amounts of current, indicating that the corrosion products thickness effectively determines the channel length. When the platform is negatively biassed in the n-Si (p-MOS) situation, a forward characteristic is seen, indicating that silicon emits electrons more effectively than the other material.

The Child-Langmuir Space Charge Limited (SCL) flow of water in vacuum is represented by the forward typical with a slope of 1.5, or a V3/2 voltage dependence. This SCL current indicates a carrier injection with a minimal barrier height and scattering-free ballistic transit of electrons over the gap. The most often used semiconductor transistors in modern technology are metal-oxide semiconductor field-effect transistors (MOSFETS). MOSFETS are four-terminal electronic components with source, drain, gate, and ground connections. The field effect causes current to be permitted to stream from either the source towards the drain when a potential is put on the gate. Electrical signals may be amplified using MOSFETS, which are networked to provide electrical logic. Smaller MOSFET wavelengths and device widths made possible by advancements in manufacturing technology have directly contributed to the shrinking of integrated circuits [4] .

The same perpendicular channels pattern was created by cleaving a MOS wafer in order to confirm that the detected V3/2 dependency arises from electron transport via air (Nano scale vacuum) rather than surface conduction, which may be facilitated by etch residual or deal posit on the oxide surface. The rectifying I-V characteristic of the FIB samples is clearly seen in the cleaved samples as well. Prior to FIB etching, the leakage currently through into the oxide layer was measured at 20 pA at a partiality of 2 V, a far lower value than the aforementioned channel current threshold.

It is significantly more flexible to model the whole architecture after a program-controlled CPU. The data processing circuitry alone houses the majority of the application-specific functionalities. That is, one or more datapaths are built and hardwired in 56 Architectures of VLSI Circuits so as to facilitate certain data operations while running under the direction of a single microprogram. The amount of ALUs, their instruction sets, supporting data formats, local storage capacity, etc. are adapted to the computational issues at hand. Moreover, it is possible to provide some concurrency by having distinct datapaths work on various chunks of data concurrently. The resultant architecture is an application-specific instruction set processor (ASIP) [16]. Program storage Data Memory Program-controlled Processor Controller Handles Subtasks A, B, C, and D Using Several Specialized Datapaths.

An ASIP's hardware layout is very similar to architectural ideas used in general-purpose computing. A very-long instruction word (VLIW) architecture is what emerges as more concurrent datapath units are added. There is a choice between a single-instruction multiple-data (SIMD) model, where a number of identical datapaths operate under the control of a single instruction word, and a multiple-instruction multiple-data (MIMD) machine, where a specific field in the overall instruction word is set aside for each datapath unit [5] . Thus, it is possible to simultaneously process multiple data items with the same operation. Many vendors improved their microprocessor families in the late 1990s by including special

instructions that provide some level of concurrency in an effort to better support high-throughput video and graphics applications. The datapath of the processor is divided into several smaller subunits during each such instruction. Four 16-bit data words can be processed by a 64-bit datapath at once, for example, if the operation is the

THE ARCHITECTURAL ANTIPODES

While the mono-ASIP architecture affords flexibility, it does not provide the same degree of concurrency and modularity as the multiple processing units. A multiprocessor system built from specialised ASIPs, therefore, an interesting extension. In addition, this approach facilitates the design, interfacing, reuse, test, and on-going update of the various building blocks involved.

DISCUSSION

The Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) is a widely used device in modern electronic circuits. The MOSFET is used for switching and amplification of signals, and is a crucial component in modern electronic devices such as microprocessors, memory chips, and power electronics. The MOSFET is a voltage-controlled device that consists of a gate, source, drain, and a channel. The channel is made of a semiconductor material, which conducts current between the source and drain terminals. The flow of current in the channel is controlled by the voltage applied to the gate terminal.

The MOSFET has undergone various improvements over the years to improve its performance. However, conventional MOSFETs face several limitations, including low on-state current density, high off-state leakage current, and limited scalability [6], [7] . These limitations have prompted researchers to investigate alternative materials and structures to improve MOSFET performance. One such alternative is the MOSFET-VC, which employs a vacuum channel instead of a conventional semiconductor channel. The MOSFET-VC has several advantages over conventional MOSFETs, including higher current density, lower off-state leakage current, and improved scalability. The vacuum channel in MOSFET-VCs provides better control over the flow of current, resulting in improved device performance.

Working Principle:

The MOSFET-VC operates on the same principle as the conventional MOSFET. The device consists of a gate, source, drain, and a channel. The gate is separated from the channel by a thin layer of insulating material, typically silicon dioxide (SiO2). The source and drain terminals are doped regions in the semiconductor substrate. The key difference between a conventional MOSFET and a MOSFET-VC is the channel material. In a conventional MOSFET, the channel is made of a semiconductor material such as silicon, germanium, or gallium arsenide. In a MOSFET-VC, the channel is a vacuum, which provides a higher mobility for charge carriers. The vacuum channel in a MOSFET-VC provides better control over the flow of current through the device. The mobility of charge carriers in a vacuum is higher than in a semiconductor, resulting in higher current density. Additionally, the absence of a semiconductor material in the channel reduces the off-state leakage current, resulting in lower power consumption [8].

Fabrication:

The fabrication process of a MOSFET-VC is similar to that of a conventional MOSFET. However, additional steps are required to create the vacuum channel.

However, always keep in mind that defining a proprietary instruction set makes it impossible to take advantage of existing compilers, debugging aids, assembly language libraries, experienced programmers, and other resources that are routinely available for industry-standard processors. Industry provides us with such a vast selection of micro- and signal processors that only very particular requirements justify the design of a proprietary CPU. 12 Example

While generally acknowledged to produce more realistic renderings of 3D scenes than industrystandard raster graphics processors, ray tracing algorithms have long been out of reach for real-time applications due to the myriad floating-point computations and the immense memory bandwidth they require. Hardwired custom architectures do not qualify either as they cannot be programmed or as ray tracing necessitates many data-dependent recursions and decisions.

The technique is best described as sub-word parallelism, but is better known under various trademarks such as multimedia extensions (MMX), streaming SIMD extensions (SSE) (Pentium family), Velocity Engine, AltiVec, and VMX (PowerPC family) (PowerPC family). 1 2 [18] reports on an interesting approach to expedite ASIP development whereby assembler, linker, simulator, \sand RTL synthesis code are generated automatically by system -level software tools. Product designers can \sthus essentially focus on defining the most appropriate instruction set for the processor in view of the target application.

58 Architectures of VLSI Circuits

Ray tracing may finally find more general adoption in multi-ASIP architectures that combine multiple ray processing units (RPUs) into one powerful rendering engine. Working under control of its own programme thread, each RPU operates as a SIMD processor that follows a subset of all rays in a scene. The independence of light rays allows a welcome degree of scalability where frame rate can be traded against circuit complexity. The authors of [19] have further paid attention to defining an instruction set for their RPUs that is largely compatible with pre-existing industrial graphics processors.

2.2.6 Configurable computing

Another crossbreed between dedicated and general-purpose architectures did not become viable until the late 1990s but is now being promoted by FPL manufacturers and researchers. The IEEE 1532 standard has also been created in this context. The idea is to reuse the same hardware for implementing sub functions that are mutually exclusive in time by reconfiguring FPL devices on the fly.

The general hardware arrangement bears some resemblance to the coprocessor approach of yet in-system configurable (ISC) devices are being used instead of hardwired logic. As a consequence, the course of operations is more sophisticated and requires special actionfrom the hardware architects. For each major subtask, the architects must ask themselves whether the computations involved: Qualify for being delegated to in-system configurable logic, Never occur at the same time or can wait until the FPL device becomes free and Whether the

time for having the FPL reconfigured in between is acceptable or not. Figure 1 illustrate the Metal–oxide semiconductor field-effect transistor with a vacuum channel.

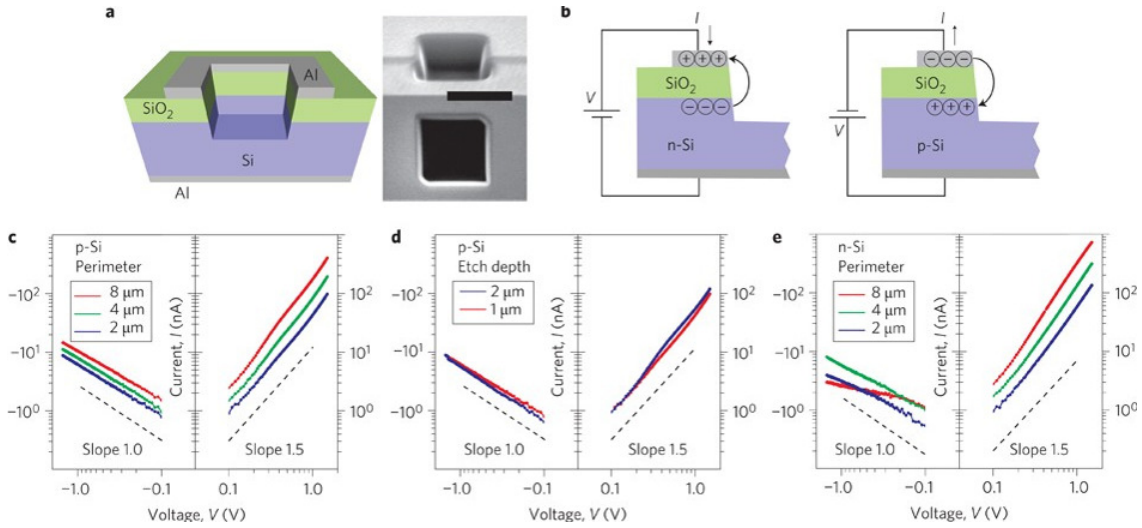


Figure 1: illustrate the Metal–oxide–semiconductor field-effect transistor with a vacuum channel.

Typically this would be the case for repetitive computations that make use of sustained, highly parallel, and deeply pipelined bit-level operations. When designers have identified some suitable subfunction, they devise a hardware architecture that solves the particular computational problem with the resources available in the target FPGA or CPLD, prepare a configuration file, and have that stored in a configuration memory. In some sense, they create a large hardware procedure instead of programming a software routine in the customary way. Whenever the host computer encounters a call to such a hardware procedure, it configures the FPL accordingly by downloading the pertaining configuration file. From now on, all the host has to do. The host then fetches the results before proceeding with the next subtask.

It thus becomes possible to support an assortment of data processing algorithms each with its optimum architecture or almost so from a single hardware platform. What often penalizes this approach in practise are the dead times incurred whenever a new configuration is being loaded. Another price to pay is the extra memory capacity for storing the configuration bits for all operation modes. Probably the most valuable benefit, however, is the possibility of being able to upgrade information processing hardware to new standards and/or modes of operation even after the system has been fielded.

Transcoding video streams in real time is a good candidate for reconfigurable computing because of the many formats in existence such as DV, AVI, MPEG-2, DivX, and H.264. For each conversion scheme, a configuration file is prepared and stored in local memory, from where it is transferred into the reconfigurable coprocessor on demand. And should a video format or variation emerge that was unknown or unpopular at the time when the system was being developed, extra configuration files scan be made available in a remote repository from where they can be fetched much like software plug-ins get downloaded via the Internet.

The results from a comparison between Lempel–Ziv data compression with a reconfigurable coprocessor and with software execution on a processor.

Extendable instruction set processors

This latest and most exotic approach pioneered by Stretch borrows from ASIPs and from configurable computing. Both a program-controlled processor and electrically reconfigurable logic are present on a common hardware platforms. The key innovation is a suite of proprietary EDA tools that allows system developers to focus on writing their application programme in C or C++ as if for a regular general purpose processor. Those tools begin by profiling the software code in order to identify sequences of instructions that are executed many times over. For each such sequence, reconfigurable logic is then synthesised into a dedictated and massively parallel computation network that completes within one clock cycle ideally at least. Finally, each occurrence of the original computation sequence in the machine code gets replaced by a simple function call that activates the custom-made datapath logic. In essence, the base processor gets unburdened from lengthy code sequences by augmenting his instruction set with a few essential additions that fit the application and that get tailor-made

As an extension to the general procedure described here, an extra optimization step can be inserted before the coprocessor is configured. During this stage, the host would adapt a predefined generic configuration to take advantage of particular conditions of the specific situation at hand. Consider pattern recognition, for instance, where the template remains unchanged for a prolonged lapse of time, or secret-key (de)ciphering, where the same holds true for the key. It is often possible to simplify arithmetic and logic hardware a lot provided that part of the operands have fixed values.

Yet, the existence of reconfigurable logic and the business of coming up with a suitable hardware architecture are hidden from the system developer. The fact that overall program execution remains strictly sequential should further simplify the design process. Program execution on a general-purpose processor and hardwired circuitry optimised for one specific flow of computation are two architectural antipodes. Luckily, many useful compromises exist in between, and this is reflected: Observation 2.4. Rely on dedicated hardware only for those sub functions that are called many times and are unlikely to change; keep the rest programmable via software, via reconfiguration, or both. While there are many ways to trade agility for computational efficiency and vice versa, the two seem to be mutually exclusive as we know of no architecture that would meet both goals at the same time.

A transform approach to VLSI architecture design

Let us now turn our attention to the main topic of this chapter: “How to decide on the necessary hardware resources for solving a given computational problem and show to best organise them.” Their conceptual differences notwithstanding, many techniques for obtaining high performance at slow cost are the same for general- and special-purpose architectures. As a consequence, much of the material presented in this chapter applies to both of them. Yet, the emphasis is on dedicated architectures as the a priori knowledge of a computational problems offers room for a number of ideas that do not apply to instruction-set processor architectures. Most data and signal processing algorithms would lead to grossly inefficient or even infeasible solutions if they were implemented in hardware as they are. Adapting processing algorithms to the technical and economic conditions of large-scale integration is one of the intellectual challenges in VLSI design.

Basically, there is room for remodelling in two distinct domains, namely in the algorithmic domain \sand in the architectural domain.There exists an excellent and comprehensive literature on general-purpose architectures including. The historical evolution of the microprocessor is summarised in along with economic facts and trends. Emphasise the impact of deep-submicron technology on high-performance microprocessor architectures.

Circuit Architectures in VLSI

In the algorithmic domain, the focus is on minimising the number of computational operations weighted by the estimated costs of such operations. A given processing algorithm thus gets replaced by a different one better suited to hardware realisation in VLSI. Data structures and number representation schemes are also subject to optimizations such as subsampling and/or changing from floating-point to fixed-point arithmetics. All this implies that alternative solutions are likely to slightly differ in their functionality as expressed by their input-to-output relations.

When designing a digital filter, one is often prepared to tolerate a somewhat lower stopband suppression or a larger passband ripple in exchange for a reduced computational burden obtained, for instance, from substituting a lower order filter and/or from filling in zeros for the smaller coefficients. Conversely, a filter structure that necessitates a higher number of computations may sometimes prove acceptable in exchange for less stringent precision requirements imposed on the individual arithmetic operations and, hence, for narrower data words.

In a decoder for digital error-correction, one may be willing to sacrifice 0.1 dB or so of coding gain for the benefit of doing computations in a more economic way. Typical simplifications to the ideal Viterbi algorithm include using an approximation formula for branch metric computation, truncating the dynamic range of path metrics, rescaling them when necessary, and restricting traceback operations to some finite depth. The autocorrelation function (ACF) has many applications in signal processing, yet it is not always needed in the form mathematically defined.

$$\text{ACF}x\ x\ (k) = r x\ x\ (k) = \infty\ n = -\infty\ x(n) \cdot x(n + k) \quad (2.1)$$

Many applications offer an opportunity to relax the effort for multiplications because one is interested in just a small fragment of the entire ACF, because one can take advantage of symmetry, or because modest precision requirements allow for a rather coarse quantization of data values. It is sometimes even possible to substitute the average magnitude difference function (AMDF) that does away with costly multiplication altogether.

$$\text{AMDF}x\ x\ (k) = r\ x\ x\ (k) = N - 1\ n = 0\ |x(n) - x(n + k)| \quad (2.2)$$

Code-excited linear predictive (CELP) coding is a powerful technique for compressing speech signals, yet it has long been left aside in favour of regular pulse excitation because of its prohibitive computational burden. CELP requires that hundreds of candidate excitation sequences be passed through a cascade of two or three filters and be evaluated in order to pick the one that fits best. In addition, the process must be repeated every few milliseconds. Yet, experiments have revealed that the usage of sparse up to 95% of samples replaced with zeros, of ternary (+1, 0, −1), or of overlapping excitation sequences has little negative impact

on auditory perception while greatly simplifying computations and reducing memory requirements [9], [10] .

CONCLUSION

The Metal–Oxide–Semiconductor Field-Effect Transistor with a Vacuum Channel (MOSFET-VC) is a novel device that has shown potential to improve the performance of conventional MOSFETs. The MOSFET-VC employs a vacuum channel instead of a conventional semiconductor channel, which provides several advantages, including higher current density, lower off-state leakage current, and improved scalability. The MOSFET-VC is a promising technology that has the potential to revolutionize the field of electronics. Further research and development are required to address the current challenges and improve the device's performance to make it a viable alternative to conventional MOSFETs.

REFERENCES

[1] J. Cho, S. Hwang, D. H. Ko, and S. Chung, "Transparent ZnO thin-film deposition by spray pyrolysis for high-performance metal-oxide field-effect transistors," *Materials (Basel).*, 2019, doi: 10.3390/ma12203423.

[2] S. Srisonphan, Y. S. Jung, and H. K. Kim, "Metal-oxide-semiconductor field-effect transistor with a vacuum channel," *Nat. Nanotechnol.*, 2012, doi: 10.1038/nnano.2012.107.

[3] W. T. Chang, H. J. Hsu, and P. H. Pao, "Vertical field emission air-channel diodes and transistors," *Micromachines*, 2019, doi: 10.3390/mi10120858.

[4] N. Miyata, "Study of Direct-Contact HfO2/Si Interfaces," *Materials (Basel).*, 2012, doi: 10.3390/ma5030512.

[5] K. Y. Lin *et al.*, "Molecular beam epitaxy, atomic layer deposition, and multiple functions connected via ultra-high vacuum," *J. Cryst. Growth*, 2019, doi: 10.1016/j.jcrysgro.2019.02.035.

[6] K. Ito, T. Kobayashi, and T. Kimoto, "Influence of vacuum annealing on interface properties of SiC (0001) MOS structures," *Jpn. J. Appl. Phys.*, 2019, doi: 10.7567/1347-4065/ab2557.

[7] D. Khim, Y. H. Lin, and T. D. Anthopoulos, "Impact of Layer Configuration and Doping on Electron Transport and Bias Stability in Heterojunction and Superlattice Metal Oxide Transistors," *Adv. Funct. Mater.*, 2019, doi: 10.1002/adfm.201902591.

[8] R. Chen and L. Lan, "Solution-processed metal-oxide thin-film transistors: A review of recent developments," *Nanotechnology*, 2019, doi: 10.1088/1361-6528/ab1860.

[9] H. Kawarada *et al.*, "C-H surface diamond field effect transistors for high temperature (400 °c) and high voltage (500V) operation," *Appl. Phys. Lett.*, 2014, doi: 10.1063/1.4884828.

[10] H. Park and J. Kim, "Enhancing ambipolar carrier transport of black phosphorus field-effect transistors with Ni-P alloy contacts," *Phys. Chem. Chem. Phys.*, 2018, doi: 10.1039/C8CP02285B.

CHAPTER 9

IMPORTANCE OF VLSI IN MODERN DAYS AND USES OF VLSI

Dr. Sachin Gupta, Chancellor,
Department of Management, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id- chancellor@sanskriti.edu.in

ABSTRACT:

Very Large-Scale Integration (VLSI) is a field of electronics that deals with the design and manufacturing of integrated circuits (ICs). ICs are the building blocks of modern electronic devices, from smartphones and laptops to medical equipment and automobiles. VLSI has revolutionized the electronics industry by enabling the creation of complex and powerful electronic devices that are smaller, faster, and more efficient than ever before. In this article, we will discuss the importance of VLSI in modern days and the uses of VLSI in various industries.

KEYWORDS:

Automobiles, Electronics, Integrated Circuit, VLSI, Medical Equipment.

INTRODUCTION

The technique of integrating or embedding thousands of transistors on a single silicon semiconductor microchip is known as VLSI. VLSI is important nowadays because it makes it possible to build large memory arrays, which are utilised in microcontrollers and microprocessors.VLSI is crucial since it helps with compact design in a big way. Compared to a circuit with discrete components, it uses less power. VLSI may be used in a small package for a variety of tasks.

Very large-scale integration is the process of anchoring or incorporating significant numbers of electronics onto a single made of semiconductor microchip. In the late 1970s, when high level computer (computer) tracking devices were starting to take form, VLSI innovation was first created. The two most common VLSI components are the microchip and the microprocessor. VLSI stands for a somewhat embedded system that contains several devices on a virtual machine. Naturally, the term and other other grade assimilation classifications based on the number of gates or devices in a single - chip (IC) date back to the 1970s.

The phenomenal growth of the electronics sector is mostly attributable to the establishment of better integration technologies. However, with the advent of VLSI designs, the possibilities for ICs in controlling, communication, somewhat raised computing, and electrical items are growing. VLSI technology enables previously unheard-of levels of mobility, processor speed, and access controls in modern devices like phones and cellular telephony. The demand for this trend is expected to increase quickly.Circuit size reduction, higher device cost-effectiveness, an improvement in circuit engine speed, and greater performance are the key advantages of VLSI technology. Less power is utilised than with individual components, and device reliability is also boosted[1], [2].

The Design Process for VLSI ICs

There are typically two primary stages or components in a VLSI IC design: Front-end design is the practise of creating digital designs using arduino ide program, such like Rtl, System Mainframe, and VHDL. This stage also involves architecture certification via emulators and other forms of verification. The whole process also includes designing, which starts with the buttons and continues through design with testing.

To detect odd circumstances, medical electronic equipment assess bodily activities and run intricate processing algorithms. The availability of such, or complicated systems, distant enough from consumers to overload them merely increases demand for ever more complex systems.By placing several MOS transistors on a single chip, a microcircuit is created using a method known as very large-scale integration. Widespread use of microcircuit chips has made it possible to create sophisticated semiconductor and communications technologies [3].

Benefits of VLSI

a) Circuit size is decreased.

b) Decreases the gadgets' actual cost.

c) increases circuit operating speed

d) Less electricity is used than with discrete components.

e) Increased Reliability

f) Uses a substantially lower amount of space.

g) How Partech assists in providing an understanding of VLSI and its significance

h) Partech eLearning provides an overview of VLSI and explains its significance. Pantech eLearning provides projects, workshops, internships, and courses on VLSI.

It is also about the fundamental ideas of Automation and control, which provides information in the form of info graphics. It is intended for self-learning and may support the development of economic competencies in the VLSI world. This concentrates on the fundamental elements and features of MOS Transistor.

In designing computational hardware that makes use of trigonometric functions, look-up tables (LUTs) are likely to prove impractical because of size overruns. Executing a lengthy algorithm, the other hand, may be just too slow, so a tradeoff among circuit size, speed, and precision must be found. The CORDIC (coordinate rotation digital computer) family of algorithms is one such compromise that was put to service in scientific pocket calculators in the 1960s and continues to find applications in DSP. Note that CORDIC can be made to compute hyperbolic and other transcendental functions too [4], [5].

Computing the magnitude function $m = \sqrt{a2 + b2}$ is a rather costly proposition in terms of circuit hardware. Luckily, there exist at least two fairly precise approximations based on add, shift, and compare operations exclusively, see table 2.8 and problem 1. Better still, the performance of many optimization algorithms used in the context of demodulation, error correction, and related applications does not suffer much when the computationally expensive 2 -norm gets replaced by the much simpler 1 - or ∞-norm.

The common theme is that the most obvious formulation of a processing algorithm is not normally the best starting point for VLSI design. Departures from some mathematically ideal algorithm are almost always necessary to arrive at a solution that offers the throughput and energy efficiency requested at economically feasible costs. Most algorithmic modifications alter the input-to-output mapping and so imply an implementation loss, that is a minor cut-back in signal-to-noise ratio, coding gain, bit-error-rate, mean time between errors, stop band suppression, passband ripple, phase response, false-positive and false-negative rates, data compression factor, fidelity of reproduction, total harmonic distortion, image and colour definition, intelligibility of speech, or whatever figures of merit are most important for the application.

Experience tells us that enormous improvements in terms of throughput, energy efficiency, circuit size, design effort, and agility can be obtained by adapting an algorithm to the peculiarities and cost factors of hardware. Optimizations in the algorithmic domain are thus concerned with "How to tailor an algorithm such as to cut the computational burden, to trim down memory requirements, and/or to speed up calculations without incurring unacceptable implementation losses."

What the trade-offs are and to what extent departures from the initial functionality are acceptable depends very much on the application. It is, therefore, crucial to have a good command of the theory and practise of the computational problems to be solved. Architectures of VLSI Circuits minimum memory results in an implementation loss. The effort for finding a good compromise between numerical accuracy and hardware efficiency is often underestimated. The necessity to validate trimmed-down implementations for all numerical conditions that may occur further adds to the effort. It is not uncommon to spend as much time on issues of numerical precision as on all subsequent VLSI design phases together.

In the architectural domain, the focus is on meeting given performance targets for a specific data processing algorithm with a minimum of hardware resources. The key concern is "How to organise datapaths, memories, controllers, and other hardware resources for implementing some given computation flow such as to optimise throughput, energy efficiency, circuit size, design effort, agility, overall costs, and similar figures of merit while leaving the original input-to-output relationship unchanged except, possibly, for latency[6], [7]."

Given some data or signal processing algorithm, there exists a profusion of alternative architectures although the number of fundamental options available for reformulating it is rather limited. This is because each such option can be applied at various levels of detail and can be combined with others in many different ways. Our approach is based on reformulating algorithms with the aid of equivalence transforms. The remainder of this chapter gives a systematic view on all such transforms and shows how they can be applied to optimise VLSI architectures for distinct size, throughput, and energy targets.

Systems theorists tend to think in purely mathematical terms, so a data or signal processing algorithm is not much more than a set of equations to them. To meet pressing deadlines or just for reasons of convenience, they tend to model signal processing algorithms in floating-point arithmetics, even when a fairly limited numeric range would amply suffice for the application. This is typically unacceptable in VLSI architecture design and establishing a lean bit-true software model is a first step towards a cost-effective circuit.

DISCUSSION

Generally speaking, it is always necessary to balance many contradicting requirements to arrive at a working and marketable embodiment of the mathematical or otherwise abstracted initial model of a system. A compromise will have to be found between the theoretically desirable and the economically feasible [8]. So, there is more to VLSI design than just accepting a given algorithm and turning that into gates with the aid of some HDL synthesis tool.

Algorithm design is typically carried out by systems engineers whereas VLSI architecture is more the domain of hardware designers.

The strong mutual interaction between algorithms and architectures mandates a close and early collaboration between the two groups, see fig.2.9. Observation 2.7. Finding a good tradeoff between the key characteristics of the final circuit and implementation losses requires an on-going collaboration between systems engineers and VLSI experts during the phases of specification, algorithm development, and architecture design.

Importance of VLSI in Modern Days:

1. **Advancements in Technology:** The advancements in technology have led to the development of sophisticated electronic devices that require small, high-performance ICs. VLSI has played a significant role in the development of these devices by enabling the integration of millions of transistors on a single chip.
2. **Miniaturization of Devices:** With the increasing demand for miniaturization of electronic devices, VLSI has become an essential technology. VLSI has enabled the development of small and lightweight electronic devices that can be easily carried around and used in various applications.
3. **Power Efficiency:** Power efficiency is an essential factor in electronic devices. VLSI has enabled the development of low-power ICs that consume less power and are ideal for battery-powered devices.
4. **Cost-Effective:** The VLSI technology has also made electronic devices more affordable. With the integration of multiple functions on a single chip, the cost of manufacturing electronic devices has reduced, making them more accessible to people.
5. **Performance:** The performance of electronic devices has significantly improved with VLSI technology. VLSI has enabled the development of high-performance ICs that are faster and more efficient than their predecessors.

Uses of VLSI in Various Industries:

1. **Consumer Electronics:** VLSI has revolutionized the consumer electronics industry by enabling the development of smaller, faster, and more efficient electronic devices. From smartphones to laptops, VLSI has played a crucial role in the development of these devices. The use of VLSI in consumer electronics has also led to the development of wearable devices and smart home appliances.

2. **Medical Electronics:** The use of VLSI in medical electronics has enabled the development of sophisticated medical devices that are used for diagnosis, monitoring, and treatment. These devices include ultrasound machines, CT scanners, and MRI machines.
3. **Automotive Industry:** VLSI has played a vital role in the development of automotive electronics. The use of VLSI in the automotive industry has enabled the development of sophisticated electronic systems that improve the performance, safety, and comfort of vehicles.
4. **Aerospace Industry:** The aerospace industry has also benefited from VLSI technology. The use of VLSI in aerospace electronics has enabled the development of sophisticated systems that are used in satellites, rockets, and other space vehicles.
5. **Defense Industry:** The defense industry has also been a significant beneficiary of VLSI technology. VLSI has enabled the development of sophisticated electronic systems that are used in defense applications such as radar systems, communication systems, and missile guidance systems.
6. **Industrial Automation:** The use of VLSI in industrial automation has enabled the development of sophisticated control systems that improve the efficiency and productivity of manufacturing processes. VLSI is used in programmable logic controllers (PLCs) and other automation systems.
7. **Telecommunications:** The telecommunications industry has also benefited from VLSI technology. VLSI has enabled the development of high-speed data communication systems that are used in fiber optic networks, wireless communication systems, and satellite communication systems.

Design architecture technologyspecific implementation algorithm design product idea IC fabrication data evaluation of functional needs and specification

a) design architecture technology specific implementation algorithm design IC fabrication data evaluation of functional needs and specification product idea
b) competence of systems engineers competence of systems engineers competence of VLSI designers competence of VLSI designers

Models of collaboration between systems engineers and hardware designers. Sequential thinking doomed to failure versus a networked team more likely to come up with satisfactory results. The fact that algorithm design is not covered in this text does not imply that it is of less importance to VLSI than architecture design[9], [10]. The opposite is probably true. A comprehensive textbook that covers the joint development of algorithms and architectures anecdotal observations can be found. Figure 1: illustrate the design of VLSI System.

We will often find it useful to capture a data processing algorithm in a data dependency graph (DDG) as this graphical formalism is suggestive of possible hardware structures. A DDG is a directed graph where vertices and edges have non-negative weights. A vertex stands for a memory less operation and its weight indicates the amount of time necessary to carry out that operation. The precedence of one operation over another one is represented as a directed edge. The weight of an edge indicates by how many computation cycles or sampling period's execution of the first operation must precede that of the second one.

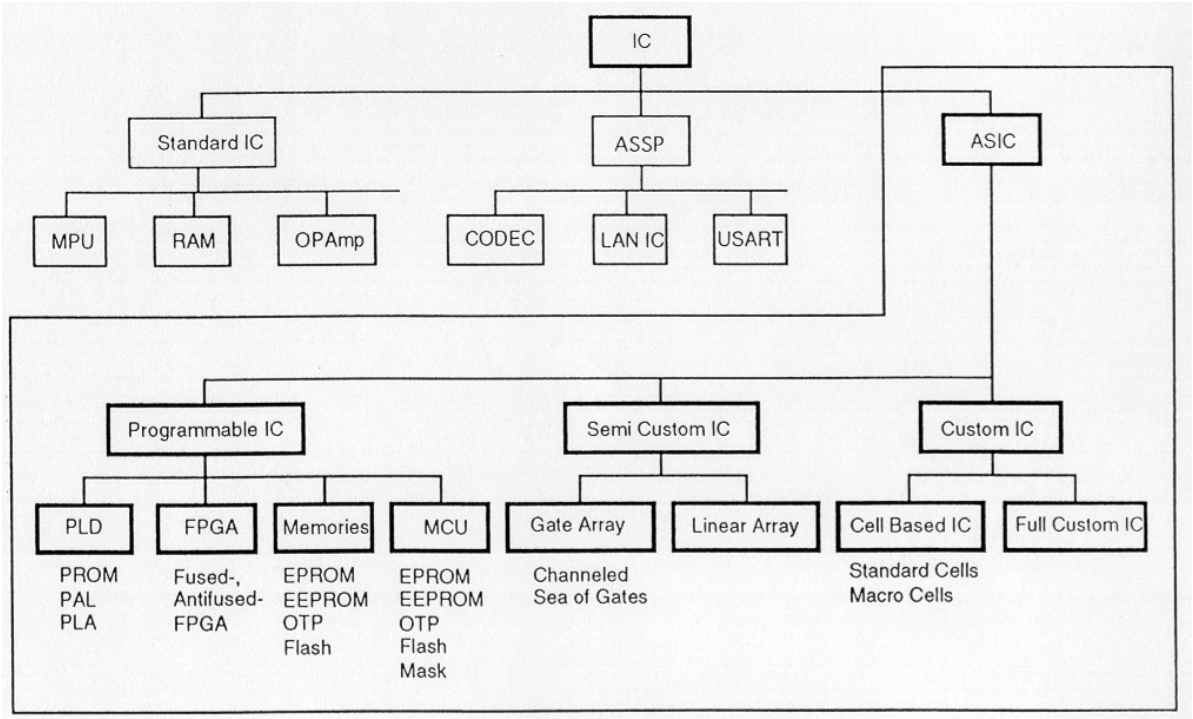


Figure 1: Illustrate the design of VLSI System.

Edge weight zero implies the two operations are scheduled to happen within the same computation or sampling period one after the other, though. An edge may also be viewed as expressing the transport of data from one operation to another and its weight as indicating the number of registers included in that transport path. To warrant consistent outcomes from computation, circular paths of total edge weight zero are disallowed in DDGs.

Put differently, any feedback loop shall include one or more latency registers. The term "computation cycle" is to be explained shortly. A circular path is a closed walk in which no vertex, except the initial and final one, appears more than once and that respects the orientation of all edges traversed. As the more customary terms "circuit" and "cycle" have

No matter how one has arrived at some initial proposal, it always makes sense to search for a better hardware arrangement. Inspired VLSI architects will let themselves be guided by intuition and experience to come up with one or more tentative designs before looking for beneficial reorganizations. Yet, for the subsequent discussion and evaluation of the various equivalence transforms available, we need something to compare with. A natural candidate is the isomorphic architecture.

Each combinational operation in the DDG is carried out by a hardware unit of its own, each hardware register stands for a latency of one in the DDG, There is no need for control because DDG and block diagram are isomorphic, and Clock rate and data input/output rate are the same. Other meanings in the context of hardware design, we prefer "circular path" in spite of its clum siness. For the same reason, let us use "vertex" when referring to graphs and "node" when referring to electrical networks.

A zero-weight circular path in a DDG implies immediate feedback and expresses a self-referencing combinational function. Such zero-latency feedback loops are known to expose the pertaining electronic circuits to unpredictable behavior and are, therefore, highly undesirable. Two directed graphs are said to be isomorphic if there exists a one-to-one correspondence between their vertices and between their edges such that all incidence relations and all edge orientations are preserved. More inform ally, two isomorphic graphs become indistinguishable when the labels and weights are removed from their vertices and edges. Remember that how a graph is drawn is of no importance for the theory of graphs.

An architecture design as naive as this obviously cannot be expected utilise hardware efficiently, but it will serve as a reference for discussing both the welcome and the unfavourable effects of various architectural reorganizations. You may also think of the isomorphic architecture as a hypothetical starting point from which any more sophisticated architecture can be obtained by applying a sequence of equivalence transforms.

Longest path delay tlp indicates the lapse of time required for data to propagate along the longest combinational path through a given digital network. Path lengths are typically indicated in ns. What makes the maximum path length so important is that it limits the operating speed of a given architecture. For a circuit to function correctly, it must always be allowed to settle to a typically new steady state within a single computation period Tcp.

Size–time product AT combines circuit size and computation time to indicate the hardware resources spent to obtain a given throughput. This is simply because $AT = A\,\Theta$. The lower the AT -product, the more hardware-efficient a circuit. Latency L indicates the number of computation cycles from a data item being entered into a circuit until the pertaining result becomes available at the output. Latency is zero when the result appears within the same clock cycle as that during which the input datum was fed in.

Energy per data item E is meant to quantify the amount of energy dissipated in carrying out some given computation on a data item. As examples consider indications in pJ/MAC, nJ/sample, µJ/datablock or mWs/videoframe. The same quantity can also be viewed as the quotient $E = P\,\Theta$ that relates power dissipation to throughput and is then be expressed in mW/ Mbit s , or W/GOPS (Giga operations per second), for instance. Using inverse term such as MOPS/mW and GOPS/W is more popular in the context of microprocessors.

Energy per data item is further related to the power–delay product (PDP) $pdp = P \cdot tlp$, a quantity often used for comparing standard cells and other transistor-level circuits. The phrase "computation period" hasn't been defined yet, but we've been using it. A calculation is divided into a series of shorter computation cycles in synchronous digital circuits, whose rhythm is enforced by a periodic clock signal. Fresh data are generated throughout each cycle of computation from the results are processed in different arithmetic, logic, and/or routing processes in combinational circuitry before being put in the subsequent analogue register (same clock, same active edge).

One example of an ignored factor is connection delays, which is an unduly hopeful assumption. Another issue is that the propagation delays of the arithmetic operations are just added together, which might be a pessimistic assumption, especially in cascades of multiple ripple carry adders when all operands arrive at once. The only reliable method for estimating total path delays is synthesis, followed by location and route. Consider dual-edge-triggering

as an exception, where each clock period consists of two consecutive calculation periods, causing Tc p = 1 2 Tclk. Circuit Architectures in VLSI Combinatorial computations are those that just rely on the current arguments. A DDG that is devoid of circular pathways and in which all edge weights are equal to zero is a necessary condition for combinational behaviour.

Take the combinational function y(k) = f(x(k)) which has a fixed but otherwise arbitrary value. Such a scenario is shown by the DDG. Both input x(k) and output y(k), as shown by the dashed edges, may have a number of subvectors. There are no presumptions made about the complexity of f, which might be anything from a two-bit addition to an algebraic division to a data block's Fast Fourier Transform (FFT) operation and beyond. In reality, designers would be particularly focused on those activities that have a significant impact on chip size, performance, power dissipation, etc.

To make the most of the same hardware, function f is broken down into a series of subfunctions that are run one after the other. Replicating the functional unit for f and having all units operate simultaneously. Pipelining the functional unit for f to increase computation rate by reducing combinational depth and working on several successive data items concurrently. Replication and pipelining both trade circuit size for performance, but iterative decomposition achieves the reverse, as is intuitively obvious. This raises issues like "Does it make sense to combine pipelining with iterative decomposition despite their antagonistic effects?" and "Are there circumstances when replication should be favoured over pipelining?" which we will attempt to address in the subsections that follow.

Of fact, there are also several gate-level circuit possibilities for implementing a certain arithmetic or logic function. Nevertheless, since this includes lower-level concerns that heavily rely on the precise operations and the target library, we do not discuss the issue of designing and assessing such possibilities within the overall framework of architectural design. The reader is advised to study up on computer algebra and logic design in specialist books. The isomorphic structure as the basis for the architectural arrangement that will be used to compare several alternative designs, with the addition of a register at the output to allow for the cascading of architectural chunks without their longest route delays stacking up. The reference architecture's features are as follows:

CONCLUSION

VLSI is a vital technology that has played a significant role in the development of modern electronic devices. The technology has enabled the integration of millions of transistors on a single chip, leading to the development of small, fast, and efficient electronic devices. VLSI has found its applications in almost every industry, and its continued advancements are opening up new possibilities for the future.

REFERENCES

[1] C. Fougstedt and P. Larsson-Edefors, “Energy-efficient high-throughput VLSI architectures for product-like codes,” *J. Light. Technol.*, 2019, doi: 10.1109/JLT.2019.2893039.

[2] I. A. M. Elfadel, D. S. Boning, and X. Li, *Machine Learning in VLSI Computer-Aided Design*. 2019. doi: 10.1007/978-3-030-04666-8.

[3] R. D. Clark, "Emerging applications for high K materials in VLSI technology," *Materials*. 2014. doi: 10.3390/ma7042913.

[4] M. Michael Vai, *VLSI design*. 2017. doi: 10.1201/9781315274201.

[5] J. Robertson *et al.*, "Use of carbon nanotubes for VLSI interconnects," *Diam. Relat. Mater.*, 2009, doi: 10.1016/j.diamond.2009.02.008.

[6] K. Vasanth *et al.*, "FSM based VLSI architecture for decision based neighborhood referred asymmetrical trimmed variant filter," in *Procedia Computer Science*, 2019. doi: 10.1016/j.procs.2019.05.035.

[7] B. S. Kandula, K. Padma Vasavi, and I. Santi Prabha, "Area and power potent VLSI architecture for modified CSLA with a logic optimization technique," *Int. J. Innov. Technol. Explor. Eng.*, 2019, doi: 10.35940/ijitee.L3051.1081219.

[8] E. H. Wold and A. M. Despain, "Pipeline and Parallel-Pipeline FFT Processors for VLSI Implementations," *IEEE Trans. Comput.*, 1984, doi: 10.1109/TC.1984.1676458.

[9] B. Pullarao and S. Srinivasan, "Energy storage with inclusion of power stability improvement with advanced controls based on VLSI," *Int. J. Recent Technol. Eng.*, 2019, doi: 10.35940/ijrte.B1146.0982S1019.

[10] T. Zhan, S. Z. Fatmi, S. Guraya, and H. Kassiri, "A Resource-Optimized VLSI Implementation of a Patient-Specific Seizure Detection Algorithm on a Custom-Made 2.2 cm 2 Wireless Device for Ambulatory Epilepsy Diagnostics," *IEEE Trans. Biomed. Circuits Syst.*, 2019, doi: 10.1109/TBCAS.2019.2948301.

CHAPTER 10

RECENT TRENDS IN MEDICAL IMAGING BY USING VLSI

Dr. Rahul Kumar, Assistant Professor
Department of Mechanical Engineering, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id- rahulk.soeit@sanskriti.edu.in

ABSTRACT:

Medical imaging is an essential part of modern healthcare, helping doctors to diagnose and treat a wide range of medical conditions. The use of VLSI technology in medical imaging has led to significant advancements in imaging quality, speed, and accuracy. With the emerging technologies of machine learning and artificial intelligence, the use of VLSI technology in medical imaging is set to become even more significant in the coming years, leading to further improvements in medical diagnosis and treatment. In short, VLSI technology has revolutionized medical imaging and will continue to play a crucial role in the healthcare industry in the future.

KEYWORDS:

Artificial Intelligence, Accuracy, Technology, Imaging Quality, VLSI.

INTRODUCTION

Numerous life-saving medical devices have been inserted into the human heart and brain body, too. Pacemaker is one of the hospital equipment. The primary disadvantage is that the battery of the Pacemaker batteries. In order to have these batteries updated, the patient needs regular surgery. The essential task is to make a Low powered, Low voltage circuit that can switch and manage battery charging. Also, analogue the charging current is switched and controlled by an ASIC or VLSI circuit. Using a lithium-ion battery turn on the pacemaker. The charging mechanism is bio-free and portable. The processing of ultrasounds is also will benefit more from the adoption of VLSI Technology than any other method of medical imaging since VLSI is utilised in ultrasound systems for temporary 3D displays. Thermocouple devices are being utilised to measure heat and it transforms this heat from the body into the dielectric breakdown and this P.D. is capable of charging the battery [1], [2] .

This method can provide enough energy to power pacemakers when implanted. It is intended to produce a thermoelectric power based on the human body. To change body temperature, thin thermoelectric films are used. Charge charges for low power devices, such pacemakers, by converting (heat) into electrical energy. Pacemaker and other very sensitive transistor devices use complementary metal-oxide silicon (CMOS). When pacemakers are used with very photonic integrated (VLSI) circuits, the functional unit they contain becomes more radiation-sensitive defects. Instead of its Bipolar/Unipolar Low Sensitivity Analog Properties Technology, specifically the use of Mosfets for analogue, expands its capabilities.

It has benefits like that. In the market today, CMOS IC architecture is readily accessible. Now, CMOS ASIC is produced instead of bipolar ICs. Currently, several well-known companies provide ICs that blend analogue and digital signals. The creator is to develop

novel CMOS technology analogue function implementation methods. Designer employs modern Technique has shown great success in developing fresh strategies for creating analogue CMOS circuits that can. Bipolar analogue design correctness. Located in the right heart atrium. The heart's Right Atrium is home to a cluster of nerves known as the somatoatrial node (SA). Additionally, these impulses travel along the atrium's conducting fibers to the atrioventricular the depolarization of the node, which is situated in the Lower section of the heart wall, as the AV node or node, respectively.

Electrical impulses are produced by the SA node known as the natural pacemaker, and they are primarily responsible for heart's contraction and expansion A pacemaker is a tiny device that is inserted under the skin to assist regulate a person's heartbeat. Patient requires a pacemaker because the (SA) node is malfunctioning [3]. (SA) Node when if the irregular cardiac rhythm causes the heart to not function correctly. The inadequate blood flow to the cardiac muscle is Chest pain (Angina pectoris), also known as ischemic heart disease, or myocardial infarction, is the result. Attack. Heart enlargement, which might be left ventricular, left aortic, or right, causes hypertrophy.

Right Artery or the Ventricular. The consequences of the metabolic changes might include incorrect medicine, improper electrolytes, or thyroid condition the pacemaker's batteries had a five-year guarantee, although their lifespan was just three to four years. To we built a VLSI circuit to get around the batteries' short lifespan. Additionally, this issue is resolved as the Body heat may be converted into energy and used to power low-power electronics the generation of heat in

With the use of a VLSI circuit, the body may generate power. Pacemaker is only one use for this application, although it is also applicable to other bio-medical devices. Medical imaging currently often uses VLSI circuits rather than general-purpose for applications such as computer processors in 3D image displays, bespoke VLSI ICs, and digital signal processing ultrasonic. It is possible to state that sound processing will be superior to all other types of medical imaging. Due to the fact that VLSI is used to construct entirely digital front ends for real-time ultrasound phased array signals processor. VLSI is around 100,000,000 transistors in size. Among them is the present generation of a 40–50 million transistor CPU The degree of chip design used nowadays is VLSI. After thatULSI, or "ultra large scale integration," is the highest level of VLSI and has an estimated one billion transistors [4], [5] .

Several of the architectural arrangements that will be presented need additional circuitry for datapath operation control and data item routing. Actl and Ectl, two additive words, are added to take this into consideration as appropriate. Actl is on the order of Areg or greater for the majority of architectural changes, but it is extremely impossible to estimate the additional hardware without comprehensive understanding of the particular case at hand. When sophisticated control schemes are used as a consequence of mixing numerous transformations, control overhead may really become large or even dominating.

In terms of energy, our attention will be on the dynamic contribution that is lost in the charging and discharging of electrical circuit nodes as a result of new data spreading across gate-level networks. When comparing low-leakage static CMOS circuits that are moderately active, any dissipation resulting from static currents or from idle switching is disregarded.

DISCUSSION

Iterative decomposition, or decomposition for short, is nothing more than the sharing of resources via sequential execution. A series of d separate jobs are performed one at a time as part of the calculation of function f.

A single hardware unit may be reused several times thanks to the recycling of intermediate outcomes from a dataflow perspective until the ultimate result is made accessible at the output d calculation cycles later schematically depicts a setup that time-multiplexes a multifunctional datapath to perform f in d = 3 successive phases. Take note of the inclusion of a control section that directs the datapath via a number of control lines on a cycle-by-cycle basis.

1. Digital Imaging and Communications in Medicine (DICOM)

DICOM is a widely used standard for medical imaging, which allows medical images to be exchanged and viewed across different medical systems. DICOM has made it easier for doctors and healthcare professionals to access medical images and diagnose patients quickly and accurately. VLSI technology has played a significant role in the development and implementation of DICOM standards, making it possible to store and retrieve medical images efficiently and securely.

2. Computed Tomography (CT) Imaging

CT imaging is a medical imaging technique that uses X-rays to produce detailed images of internal body structures. VLSI technology has made it possible to develop more advanced and faster CT scanners, which can produce high-quality images in a fraction of the time it used to take. With the help of VLSI technology, CT scanners have become more accurate, more efficient, and more accessible, making it easier for doctors to diagnose and treat medical conditions.

3. Magnetic Resonance Imaging (MRI)

MRI is a non-invasive medical imaging technique that uses a strong magnetic field and radio waves to produce detailed images of internal body structures. VLSI technology has made it possible to develop more powerful and advanced MRI machines, which can produce high-quality images in less time. With the help of VLSI technology, MRI machines have become more accurate, more efficient, and more accessible, making it easier for doctors to diagnose and treat medical conditions.

4. Positron Emission Tomography (PET) Imaging

PET imaging is a medical imaging technique that uses a radioactive substance to produce images of internal body structures. VLSI technology has made it possible to develop more advanced and faster PET scanners, which can produce high-quality images in a fraction of the time it used to take. With the help of VLSI technology, PET scanners have become more accurate, more efficient, and more accessible, making it easier for doctors to diagnose and treat medical conditions.

5. Ultrasonic Imaging

Ultrasonic imaging is a medical imaging technique that uses high-frequency sound waves to produce images of internal body structures. VLSI technology has made it possible to develop more advanced and faster ultrasonic scanners, which can produce high-quality images in less time. With the help of VLSI technology, ultrasonic scanners have become more accurate, more efficient, and more accessible, making it easier for doctors to diagnose and treat medical conditions.

6. Medical Image Processing

VLSI technology has made it possible to develop more advanced and efficient medical image processing algorithms, which can be used to enhance the quality and accuracy of medical images. With the help of VLSI technology, medical image processing has become more accurate, more efficient, and more accessible, making it easier for doctors to diagnose and treat medical conditions.

7. Machine Learning and Artificial Intelligence

Machine learning and artificial intelligence are emerging technologies that are being used in medical imaging to enhance the quality and accuracy of medical images. With the help of VLSI technology, machine learning algorithms can be implemented on imaging devices to improve the speed and accuracy of medical image processing. Artificial intelligence can be used to analyze medical images and provide more accurate and timely diagnoses, leading to better patient outcomes.

8. Wearable Medical Devices

Wearable medical devices are becoming increasingly popular, allowing patients to monitor their health and medical conditions in real-time. VLSI technology has played a significant role in the development of wearable medical devices, making it possible to develop small, fast,

So that Areg Actl Af and Ereg Ectl Ef, let's limit our study to circumstances when the control overhead can be maintained to a minimum.

Determine whether size A(d) tends closer towards its lower or higher limit in order to understand the facts presented. According to (2.16), iterative decomposition may greatly reduce the AT-product in the first scenario, while it is ineffective in the second.

When a single subfunction is repeatedly used by the chunk's function, the lower limitations still apply since the required datapath is effectively achieved by splitting the computation of into d identical chunks, only one of which is implemented in hardware. In this instance, a processing unit with a single function will do.

On the other end of the spectrum are cases when calculating f requires a wide range of subfunctions that cannot effectively share a large amount of hardware resources. In this situation, iterative decomposition is not a desirable alternative, particularly when register latency, control overhead, and the challenge of satisfying assumption 2 are taken into account. Figure 1 illustrate the medical image processing.

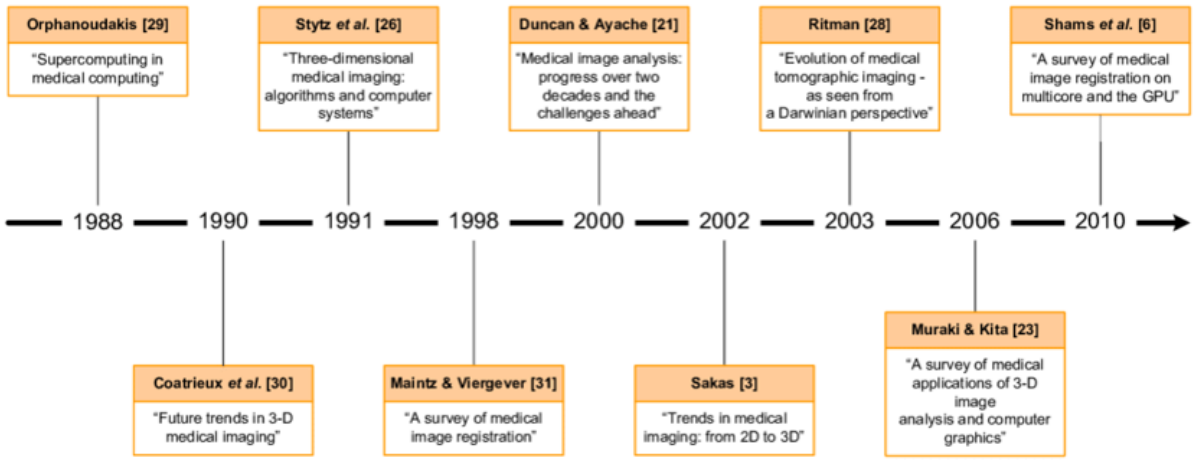


Figure 1: illustrate the medical image processing.

Addition and subtraction in either fixed-point or floating-point arithmetic, as well as different shift and rotate operations, are operations that lend themselves nicely to being integrated into a single computing unit [6], [7] . CORDIC units employ roughly the same hardware for trigonometric and hyperbolic functions, as well as for rotating angles. Regarding energy efficiency, there are two opposing processes. Iterative decomposition, on the other hand, involves register activity that wasn't there in the initial circuit. Dissipation is further increased by the additional control and data recycling logic required to achieve step-by-step execution.

We shall discover, however, that lengthy register-to-register signal propagation routes often encourage transitory node activity, or glitches. Reducing such propagation pathways often aids in reducing glitching activities and the energy losses they entail. Nevertheless, such second-order effects are not taken into account in the simple unit-wise additive model established, making it difficult to understand the effect of iterative decomposition on energy prior to the availability of precise circuit information.

The electronic code book (ECB) mode of a secret-key block cypher is a very costly combinational function. According to ECB, y(k) = c(x(k), u(k)) suggests a memoryless mapping where x(k) stands for the plaintext, y(k) for the ciphertext, u(k) for the key, and k is the block number or time index. A cascade of multiple rounds is a feature shared by the majority of block cyphers, including the Data Encryption Standard (DES), the International Data Encryption Algorithm (IDEA), and the Advanced Encryption Standard (AES) Rijndael [38]. The values of the subkeys utilised, which are obtained from u, are the sole variation between the otherwise identical rounds (k). The so-called output transform is nothing more than a breakdown of the earlier rounds.

If we choose iterative decomposition, a logical option is to build a datapath for one round and reuse the data while updating the subkeys until all rounds have been processed. The circuit's total size is expected to remain close to the lower constraint in after this first decomposition phase since control is so straightforward. There are several different architectures for VLSI circuits that multiply by modulo (216 + 1). Control would also have an effect on how big the total circuit is. A more radical method is to give a single ALU rather than breaking down arbitrary functions into sequences of arithmetic and/or logical operations from a tiny but very adaptable set.

The notion of step-by-step processing gave rise to the datapath of every microprocessor, which is basically a piece of universal hardware. The reduced instruction set computer (RISC) might be seen as another step in the same direction. Iterative decomposition, together with programmability and time sharing, provides for this paradigm's exceptional flexibility and hardware economy, but it also explains why it performs less well and uses more energy than more focused architectural approaches. Pipelining divides combinational depth into many distinct steps with roughly uniform computing delays and inserts registers between them to increase performance.

One particular subfunction is computed by the combinational logic between two consecutive pipeline registers, which is created and optimised for this purpose. The several phases work together as a unit much like specialised employees on an assembly line. A functional unit for f that is separated into p = 3 pipeline stages by p 1 additional registers. Be aware that there is no control hardware present.

The following performance and cost metrics are affected by pipelines: As pipeline depth, both performance and size increase monotonically. The same may be said about latency. What's more intriguing is that a small number of pipeline stages, each with a sizable depth, significantly reduces the AT-product as a result. This regime is referred to as coarse grain pipelining. Combinational delay is related to register delay. Expressed as a multiple of fanout-of-4 (FO4) inverter delays, the delay on the longest route is a typical approach to indicate the amount of pipelining. FO4 inverter delays of 30 year clock frequencies

One may wish to put extra pipeline registers if they continue down this path. The gain, however, diminishes when the combinational delay per stage tf p approaches the register delay treg. The register delay, not the payload function, dominates the area-delay product for high values of p. What is the greatest calculation rate for which a pipeline may be designed is a logical issue for this kind of deep or fine grain pipelining. Two-input and or nor gates are the quickest logic gates from which effective data processing may be derived.

We must allow space for at least one such gate between two consecutive registers, even if we are willing to completely rewrite a pipeline's logic circuitry in an effort to reduce the longest route (tlp). However, due to the disproportionately large number of registers needed to implement practical applications that even come close to this theoretical minimum, as well as the difficulty of using fine-grained pipelines to satisfy assumptions 1 and 2, these applications are restricted to tiny subcircuits. Economical considerations often prevent pipelining below FO4 inverter delays per stage, even in high-performance datapath logic.

More evidence that register delay is crucial in high speed architecture may be seen in equation (2.29). A normal relationship is really treg 3-5 min (tgate). As a result, flip-flop delays are not reduced to trivial levels until around twenty layers of logic are shared across consecutive registers. A \s3 0 Fanout-of-4 inverters are often used in buffer trees driving heavy loads, hence it is reasonable to compare circuit alternatives in terms of FO4 inverters because their delays have been shown to closely match those of other static CMOS gates [8]. This is so that binary nand and nor operations may be effectively performed using MOSFETs.

Equivalence transforms for combined computations, high-speed cell library must, therefore, not only include fast combinational gates but also provide bistables with minimum insertion

delays. Moreover, it is clear from that the more complicated a function is, the more pipelining it supports in terms of finding an economical solution. Since our original assumptions 1 and 2 cannot be fully realised, efficiency is likely to decline in reality before p0 is attained. According to [39], the ideal pipeline stage depth is between 6 and 8 FO4 inverter delays.

Every pipelined datapath may expend more energy than the reference design, according to the extra registers. This is unquestionably true for fine grain pipelines, where the energy lost by switching all of those additional subcircuits ends up being the main source of energy. The circumstance is more advantageous for coarse grain patterns. By using pipeline registers, it has been shown that the glitching-related switching activity in deep combinational networks tends to be reduced. This is a positive side effect that is missed by a basic additive model.

It's interesting that while throughput is significantly enhanced, coarse grain pipelining may be used to boost energy efficiency, although indirectly. Remember that reducing the longest route while maintaining a processing pace of one data item per calculation cycle is what leads to the increased throughput. Hence, a pipelined datapath built in a slower but more energy-efficient technology, such as running CMOS logic from a lower supply voltage or employing largely minimum-size transistors, is easily able to match the throughput of the isomorphic design. We chose to create energy statistics on the presumption of similar operating circumstances and cell libraries, therefore our model is unable to account for this opportunity. The lack of energy-dissipating control logic is another very desirable feature of pipelining.

Although though pipelining may be used for any feedforward calculation, there is an economic restriction when a DDG has several parallel routes. Any delay added into one of the signal propagation pathways must be balanced by adding an additional register into each of its parallel paths in order to maintain overall functioning. These shimming registers have significant size and energy costs, particularly for deep pipes where p is high, unless they assist reduce the combinational depth there.

CONCLUSION

VLSI technology has had a significant impact on medical imaging, making it easier for doctors and healthcare professionals to diagnose and treat medical conditions [9], [10] . With the help of VLSI technology, medical imaging techniques like CT, MRI, PET, and ultrasonic imaging have become more advanced, accurate, and efficient, leading to better patient outcomes. VLSI technology has also made it possible to develop more efficient medical image processing algorithms, which can be used to enhance the quality and accuracy of medical images.

REFERENCES

[1] S. G. Sreejeesh, R. Sakthivel, and J. U. Kidav, “Beamforming algorithm architectures for medical ultrasound,” *Int. J. Innov. Technol. Explor. Eng.*, 2019, doi: 10.35940/ijitee.L2556.1081219.

[2] A. A. Sujata and Y. S. Lalitha, “Performance analysis of analog data compression and decompression using ann in 32nm finfet technology,” *Int. J. Sci. Technol. Res.*, 2019.

[3] D. A. Bader and G. Cong, "Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs," *J. Parallel Distrib. Comput.*, 2006, doi: 10.1016/j.jpdc.2006.06.001.

[4] G. Sharma and K. Kumar, "Acceleration of Images via Software and Hardware using Proprietary Tools & Open Sources for Healthcare Industry," *Int. J. Image, Graph. Signal Process.*, 2017, doi: 10.5815/ijigsp.2017.07.02.

[5] S. P. V. Arasu and S. Baulkani, "An Efficient FPGA Architecture with High-Performance 2D DWT Processor for Medical Imaging," *J. Circuits, Syst. Comput.*, 2016, doi: 10.1142/S0218126616501103.

[6] P. Y. Chen, C. C. Huang, Y. H. Shiau, and Y. T. Chen, "A VLSI implementation of barrel distortion correction for wide-angle camera images," *IEEE Trans. Circuits Syst. II Express Briefs*, 2009, doi: 10.1109/TCSII.2008.2010165.

[7] M. Hegde and P. A. Sivakumar, "Development and Implementation of VLSI Reconfigurable Architecture for Gabor Filter in Medical Imaging Application," *Int. J. Eng. Manag. Res.*, 2018, doi: 10.31033/ijemr.8.3.10.

[8] Rupali Dhobale, "Implementation of 32 Bit Binary Floating Point Adder Using\nIEEE 754 Single Precision Format\n," *IOSR J. VLSI Signal Process.*, 2015.

[9] R. Ballabriga *et al.*, "Review of hybrid pixel detector readout ASICs for spectroscopic X-ray imaging," *Journal of Instrumentation*. 2016. doi: 10.1088/1748-0221/11/01/P01007.

[10] S. P. R. Borra, R. K. Panakala, and P. R. Kumar, "VLSI implementation of image fusion using DWT- PCA algorithm with maximum selection rule," *Int. J. Intell. Eng. Syst.*, 2019, doi: 10.22266/ijies2019.1031.01.

CHAPTER 11

VLSI: DEVELOPMENT AND BASIC PRINCIPLES OF IC FABRICATION

Sandeep Kumar Verma, Research Associate
Department of R&D Cell, IIMT University, Meerut Uttar Pradesh, India
Email id- sandeep91verma@gmail.com

ABSTRACT:

VLSI (Very Large Scale Integration) refers to the process of creating an integrated circuit (IC) with millions or billions of transistors on a single chip. The development of VLSI has revolutionized the electronics industry, enabling the creation of more powerful and efficient electronic devices.

KEYWORDS:

Basic Principles, Electronic Industry, Integrated Circuit, VSLI.

INTRODCUTION

The Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) is a fundamental building block of modern electronics, used in a wide range of applications from microprocessors to power electronics. Recently, a new type of MOSFET has been proposed that uses a vacuum channel instead of a traditional semiconductor channel.

In this new device, the channel region is replaced with a vacuum, which eliminates many of the performance limitations of conventional MOSFETs. Because the electrons in the vacuum are not scattered by impurities or defects in the channel material, the vacuum MOSFET (VMOSFET) is predicted to have faster switching times and higher power handling capabilities than traditional MOSFETs.

The VMOSFET is also expected to have improved high-temperature performance due to the absence of phonon scattering in the channel region. Additionally, the device's simple structure makes it compatible with existing semiconductor manufacturing processes.

While the VMOSFET is still in the early stages of development, initial results are promising, and further research is ongoing to optimize the device's performance and explore its potential applications in various fields [1]–[3].

Very Large Scale Integration (VLSI) is a field of electronics that deals with the fabrication of integrated circuits (ICs) containing millions or billions of transistors on a single chip. The development of VLSI technology has revolutionized the electronics industry and enabled the creation of powerful computers, mobile devices, and a wide range of other advanced electronics.

The development of VLSI technology has been driven by the need for higher performance and more functionality in smaller, more compact devices. In the early days of electronic

circuits, each component was individually wired together, which was a labor-intensive process and led to bulky, unreliable devices. The advent of ICs revolutionized this process, enabling the integration of multiple components on a single chip. VLSI technology takes this integration to the next level, allowing the integration of entire systems on a single chip.

The basic principle of IC fabrication is to create a series of layers of different materials, each layer forming a specific component of the circuit. The process of IC fabrication involves a series of steps, each step building on the previous one to create the final IC. The process begins with the creation of a substrate, which is usually made of silicon, onto which the various layers of the circuit will be deposited.

The first step in IC fabrication is to create a layer of oxide on the substrate, which will act as an insulator between the different layers of the circuit. This oxide layer is created using a process called thermal oxidation, in which the substrate is heated in the presence of oxygen to create a layer of silicon dioxide[4], [5].

The next step is to deposit a layer of material on top of the oxide layer, which will form the gates of the transistors in the circuit. This layer is usually made of polysilicon, which is deposited onto the oxide layer using a process called chemical vapor deposition (CVD).

The next step is to create the source and drain regions of the transistors in the circuit. This is done by selectively doping the polysilicon gate with impurities to create the desired conductivity type. This process is called ion implantation, and it involves bombarding the polysilicon with high-energy ions of the desired impurity.

Once the source and drain regions have been created, the next step is to deposit a layer of insulating material on top of the gates and source/drain regions. This layer is usually made of silicon dioxide and is created using a process called chemical vapor deposition.

The final step in IC fabrication is to create the metal interconnects that connect the different components of the circuit together. This is done by depositing a layer of metal on top of the insulating layer and then selectively etching away the metal to create the desired interconnect pattern.

While the basic process of IC fabrication has remained the same since the early days of IC technology, advances in VLSI technology have enabled the creation of ever-more complex and powerful circuits. One of the key breakthroughs in VLSI technology was the development of the complementary metal-oxide-semiconductor (CMOS) process, which enabled the creation of low-power, high-performance circuits.

The CMOS process involves the use of both p-type and n-type transistors, which are used to create complementary circuits that consume very little power. In a CMOS circuit, one set of transistors is used to turn on and off the circuit, while the other set is used to maintain the state of the circuit when it is not being actively used.

In addition to the CMOS process, other advances in VLSI technology have enabled the creation of specialized circuits such as memory chips, microprocessors, and application-specific integrated circuits (ASICs). These specialized circuits are designed to perform specific functions, such as storing data, processing information, or interfacing with external devices.

DISCUSSION

The technique of fitting a high number of transistors onto a single chip is known as very-large-scale integration (VLSI). The development of sophisticated semiconductor and digital methods marked the beginning of it in the 1970s.A popular VLSI device is the microcontroller. Most ICs had constrained functionality prior to VLSI. On a single board, an electrical circuit often includes a CPU, ROM, RAM, and other peripherals[6], [7].

All of these may be included onto a single chip thanks to VLSI. Before getting into the technicalities, let's take a look at the history of VLSI development. Integrated circuit development throughout time there were not many transistors in the first integrated circuits.

A few logic gates were supplied by early digital circuits with transistor counts in the tens, while early integrated circuits (ICs) only had transistors. Since then, there have been significant increases in the number of devices in ICs.

Constructing a VLSI IC

A VLSI IC's design is essentially divided into two components. Digital design for the front end uses HDLs like Verilog, VHDL, SystemVerilog, and others. It covers different methods of verification, such as design verification using simulation. Starting with gates and designing for testability are all parts of the process. The CMOS library's characterisation and design make up the backend design. Physical design and failure simulation are also covered. The complete design process is carried out in a step-by-step manner. The stages in front end design would be,

Issue Description

It is an abstract illustration of the system. Performance, functionality, physical dimensions, manufacturing technology, and design methods are the main factors. Market demands, accessible technology, and the design's economic feasibility all need to be balanced. The size, speed, power, and functionality of the VLSI system are all included in the final specs.

Definition of Architecture

Basic requirements include things like floating point units, the operating system to employ (such as RISC or CISC), the amount of ALUs and their cache sizes, etc.

Effective Design

Defines the main functional components of the system, making it easier to identify the physical and electrical requirements for interconnecting the various units.

Concept Design

The register transfer level (RTL) description is produced as a result of the development of boolean expressions, control flow, word width, register allocation, etc. The RTL description is implemented on a system via HDLs.

Network Design

While the logic design provides a more straightforward representation of the logic, this phase involves actualizing the circuit as a netlist. Gates, transistors, and different interconnects

make up the netlist. Once again, this is a software phase, and the result is verified by simulation.

Dimensional Design

In this stage, the netlist is transformed into its geometric representation; the end product is referred to as a layout. The lambda rules, which specify the precise specifications of the size, ratio, and spacing between components, are followed in this stage.

Hardware development on the back end

The hardware implementation is the following (or what we study in college). We switch to the real hardware after resolving the hardware-related concerns in a virtual setting. Generally speaking, hardware manufacturing includes the following steps:

Processing of wafers

At 1400o C, pure magnesium is dissolved in a saucepan. The desired crystal direction is contained in a tiny seed, which is placed into molten silicon and progressively (1mm/minute) removed. A cylindrical ingot is used to create silicon crystals. It is sawed into wafers or discs from this cylinder. Later, crystal alignment and polishing occur.

Lithography

Photo etching and photographic masking are both steps in the photolithography process. On the wafer, a photoresist coating is placed. The wafer is aligned to a mask using a photo aligner. By exposing the wafer to infrared rays via a mask, tracks are illuminatedVLSI provides a lot of benefits. Noting that function c is not at all necessary for this involution property, it is possible to design functions to maximise cryptographic security. A natural way to balance the computational complexity of C with aspirational performance goals seems to be extensive pipelining. However, due to the two paths avoiding c, every pipeline is affected. If g(g(x)) x, x, then a function g is said to be involutory. Consider the complement function in Boolean algebra, where x x, the multiplication by one operation in traditional algebra where we have (x) x, or a mirroring operation from geometry as trivial examples. Since it allows for the use of the exact same tools for both encoding and decoding, involution is a welcome property in cryptography. Figure 1 illustrate the steps for IC manufacturing.

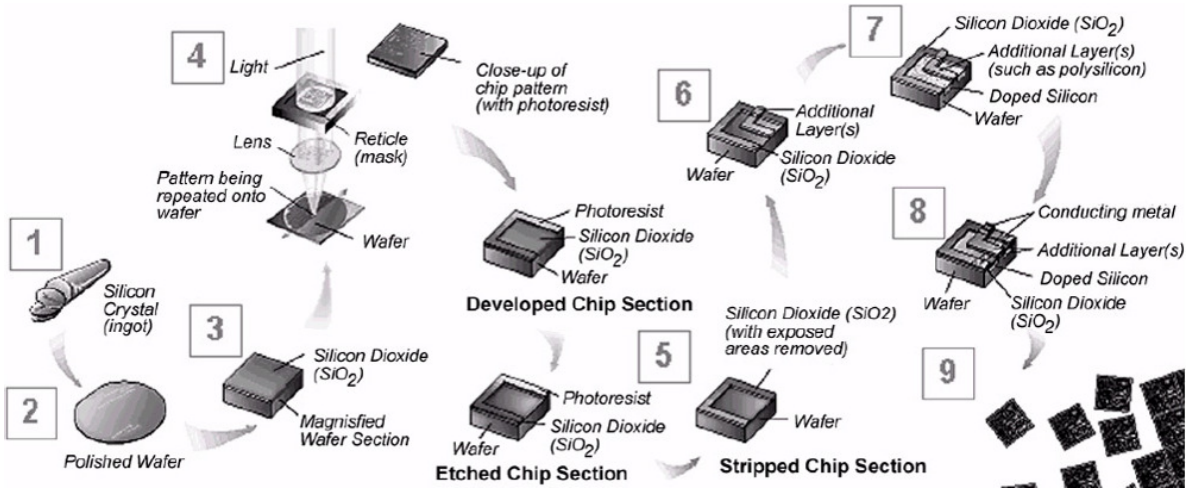


Figure 1: illustrate the steps for IC manufacturing.

Equivalence transforms for combined computations, register entails two shimming registers, effectively tripling the costs of pipelining. This is the reason why pipeline depth had to be limited to eight stages per round in a VLSI implementation of the IDEA cypher in spite of stringent throughput requirements.

Replication

Replication is a brute-force approach to performance: If one functional unit does not suffice, allow for several of them. Concurrency is obtained from providing q instances of identical functional units for f and from having each of them process one out of q data items in a cyclic manner. To that end, two synchronous q-way switches distribute and recollect data at the chunk's input and output respectively.

Any size penalties associated with distributing data to replicated functional units and with recollecting them are neglected. Multiple processing units that work in parallel are also found in situations where the application naturally provides data in parallel streams, each of which is to undergo essentially the same processing. In spite of the apparent similarity, this must not be considered as the result of replication, however, because DDG and architecture are isomorphic. This is reflected by the fact that no data distribution and recollection mechanism is required in this case. As everyone would expect, replication essentially trades area for speed. Except for the control overhead, the AT -product remains the same. Pipelining, therefore, is clearly more attractive than replication as long as circuit size and performance do not become dominated by the pipeline registers

A more accurate evaluation of replication versus pipelining would certainly require revision of some of the assumptions made here and does depend to a large extent on the actual DDG and on implementation details. Nevertheless, it is safe to conclude that neither fine grain pipelining nor replication is as cost-effective as coarse grain pipelining. Its penalising impact on circuit size confines replication to rather exceptional situations in ASIC design. A megacell available in layout form exclusively represents such a need because adding pipeline registers to a finished layout would ask for a disproportionate effort. Replication is limited to high-performance circuits and always combined with generous pipelining.

Several factors have pushed the computer industry towards replication: CMOS technology offered more room for increasing circuit complexity than for pushing clock frequencies higher. The faster the clock, the smaller the region on a semiconductor die that can be reached within a single clock period. Fine grain pipelines dissipate a lot of energy for relatively little computation. Reusing a well-tried subsystem benefits design productivity and lowers risks. A multicore processor can still be of commercial value even if one of its CPUs is found to be defective.

Now consider a situation where a number of parallel data streams undergo processing, for instance. Note that the processing functions f, g, and h may, but need not, be the same. The isomorphic architecture calls for a separate functional unit for each of the three operations \sin this case. This may be an option in applications such as image processing where a great number of dedicated but comparatively simple processing units are repeated along one or two dimensions, where data exchange is mainly local, and where performance requirements are very high[8], [9].

More often, however, the costs of fully parallel processing are unaffordable and one seeks to cut overall circuit size. A natural idea is to pool hardware by having a single functional unit process the parallel data streams one after the other in a cyclic manner. Analogously to replication, a synchronous s-way switch at the input of that unit collects the data streams while a second one redistributes the processed data at the output. While the approach is known as time-sharing in computing, it is more often referred to as multiplexing or as resource sharing in the context of circuit design. What it requires is that the circuitries for computing the various functions involved all be combined into a single datapath of possibly multifunctional nature. A student sharing his time between various subjects might serve as an analogy from everyday life.

The contrary condition occurs when f, g, and h are very dissimilar so that no substantial savings can be obtained from concentrating their processing into one multifunctional datapath.Time-sharing will then just lower throughput by a factor of s, thereby rendering it an unattractive option. Provided speed requirements are sufficiently low, a radical solution is to combine timesharing with iterative composition and to adopt a processor style. The energy situation is very similar. If the processing functions are all alike and if the computation rate is kept the same, then the energy spent for processing actual data also remains much the same. Extra energy is then spent only for controlling the datapath and for collecting and redistributing data items. More energy is going to get dissipated in a combined datapath when the functions markedly differ from each other. As time-sharing has no beneficial impact on glitching activity either, we conclude that such an architecture necessarily dissipates more energy than a comparable non-time-shared one.

By processing s data streams with a single computational unit, time-sharing deliberately refrains from taking advantage of the parallelism inherent in the original problem. This is of little importance as long as performance goals are met with a given technology. When in search of more performance, however, a time-shared datapath will have to run at a much higher speed to rival the s concurrent units of the isomorphic architecture, which implies that data propagation along the longest path must be substantially accelerated. Most measures suitable to do so, such as higher supply voltage, generous transistor sizing, usage of high-speed cells and devices, adoption of faster but also more complex adder and multiplier schemes, etc., tend to augment the amount of energy spent for the processing of one data item even further.

The Fast Fourier Transform (FFT) is a rather expensive combinational function. Luckily, due to its regularity, the FFT lends itself extremely well to various reorganisations that help reduce hardware size. A first iterative decomposition step cuts the FFT into log2 (n) rounds where n stands for the number of points. When an in-place computation scheme is adopted, those rounds become identical except for their coefficient values. The partial sum sEf + Ereg then becomes almost identical to s(Ef + Ereg) of the reference architecture. The apparent saving of Ereg obtained from making do with a single register does not materialise in practice because of the necessity to store data items from all streams. For a number of computational problem s, it is a logical choice to have two mem ories that work in a pingpong fashion. At any moment of time, one memory provides the datapath with input data [10], [11].

CONCLUSION

IC fabrication is a complex process that involves several steps, materials, and technologies. The development of VLSI and IC fabrication has revolutionized the electronics industry, enabling the creation of more powerful and efficient electronic devices. As technology continues to advance, the demand for smaller, faster, and more efficient electronic devices will only increase, driving further innovation in IC fabrication.

REFERENCES

[1] J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining trust in VLSI design: Design-for-trust techniques," *Proc. IEEE*, 2014, doi: 10.1109/JPROC.2014.2332154.

[2] A. Grill, S. M. Gates, T. E. Ryan, S. V. Nguyen, and D. Priyadarshini, "Progress in the development and understanding of advanced low k and ultralow k dielectrics for very large-scale integrated interconnects - State of the art," *Applied Physics Reviews*. 2014. doi: 10.1063/1.4861876.

[3] A. Schiltz, "A Review of Planar Techniques for Multichip Modules," *IEEE Trans. Components, Hybrids, Manuf. Technol.*, 1992, doi: 10.1109/33.142900.

[4] K. Mohamed, "Nanoimprint lithography for nanomanufacturing," in *Comprehensive Nanoscience and Nanotechnology*, 2019. doi: 10.1016/B978-0-12-803581-8.10508-9.

[5] J. Wang, S. R. Sama, P. C. Lynch, and G. Manogharan, "Design and topology optimization of 3D-printed wax patterns for rapid investment casting," 2019. doi: 10.1016/j.promfg.2019.06.224.

[6] S. H. Teen, "IC Layout Design of Decoder Using Electric VLSI Design System," *Int. J. Electron. Electr. Eng.*, 2014, doi: 10.12720/ijeee.3.1.54-60.

[7] T. Chen, "Integrated circuits," in *Microelectronics, Second Edition*, 2005. doi: 10.4324/9780429343056-18.

[8] I.-K. Cheng, C.-Y. Lin, and F.-M. Pan, "Selectivity Enhancement of ZnO Toward Reducing Gases Using Characteristic Response Features Based on Surface Kinetics below 250°C," *ECS Meet. Abstr.*, 2019, doi: 10.1149/ma2019-01/42/2025.

[9] T. Waryono, "RINGKASAN Puding merah (Gruptophyllum pictum L Griff)," *J. Farm. Galen. (Galenika J. Pharmacy)*, 2019.

[10] C.-H. Chien *et al.*, "Study Photo Imagable dielectric (PID) and non-PID on process, fabrication and reliability by using panel glass substrate for next generation interconnection," *Int. Symp. Microelectron.*, 2019, doi: 10.4071/2380-4505-2019.1.000216.

[11] Z. Huang, Q. Wang, and P. F. Yang, "Hardware Trojan: Research Progress and New Trends on Key Problems," *Jisuanji Xuebao/Chinese Journal of Computers*. 2019. doi: 10.11897/SP.J.1016.2019.00993.

CHAPTER 12

THE FUTURE OF VERY LARGE-SCALE INTEGRATION (VLSI) TECHNOLOGY

Dr. Vikram Singh, Associate Professor
Department of Computer Science and Engineering, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id- vikrams.soeit@sanskriti.edu.in

ABSTRACT:

Very Large-Scale Integration (VLSI) technology has revolutionized the electronics industry by enabling the creation of more powerful and efficient electronic devices. As technology continues to advance, the demand for smaller, faster, and more efficient electronic devices will only increase, driving further innovation in VLSI technology. In this article, we will discuss the future of VLSI technology and the potential advancements that we can expect to see in the coming years.

KEYWORDS:

Electronics, Efficient Devices, VLSI Technology, Internet of Things, 3D Integration.

INTRODUCTION

Very Large-Scale Integration (VLSI) technology has been the driving force behind the development of modern electronics, enabling the integration of millions or billions of transistors on a single chip. The future of VLSI technology is expected to bring about even more dramatic changes to the electronics industry, with the continued miniaturization of devices, the development of new materials and architectures, and the emergence of new applications and markets.

One of the key trends in the future of VLSI technology is the continued miniaturization of devices. This trend has been driven by the need for smaller, more compact devices that can be integrated into a wide range of applications, from wearable devices to smart home appliances to industrial automation systems. The miniaturization of devices is enabled by advances in lithography, which is the process used to pattern the circuits on a chip.

One of the key challenges in the continued miniaturization of devices is the development of new materials that can meet the stringent requirements of VLSI technology. For example, the use of new materials such as graphene, carbon nanotubes, and 2D materials such as molybdenum disulfide (MoS2) and tungsten diselenide (WSe2) is being explored as a way to improve the performance and reduce the power consumption of VLSI devices.

Another important trend in the future of VLSI technology is the development of new architectures for ICs. One such architecture is the 3D IC, which involves stacking multiple layers of circuits on top of each other. This architecture enables the creation of more complex circuits and the integration of heterogeneous components, such as memory and logic, on a single chip.

In addition to the development of new architectures, the future of VLSI technology is expected to bring about the emergence of new applications and markets. One such application

is the Internet of Things (IoT), which involves the integration of sensors, actuators, and other devices into a network that can be controlled and monitored remotely. VLSI technology is expected to play a critical role in the development of the IoT, enabling the creation of low-power, high-performance devices that can be integrated into a wide range of applications.

Another emerging application of VLSI technology is in the field of artificial intelligence (AI). The development of AI requires the creation of specialized hardware that can perform the complex calculations required for tasks such as image recognition and natural language processing. VLSI technology is expected to play a key role in the development of AI hardware, enabling the creation of specialized chips that can perform these tasks more efficiently and with lower power consumption than traditional processors.

In addition to these applications, VLSI technology is also expected to continue to drive innovation in the fields of medical devices, automotive electronics, and renewable energy. For example, VLSI technology is being used to create implantable devices that can monitor and regulate the body's functions, as well as sensors and control systems for electric vehicles and renewable energy systems.

One of the key challenges facing the future of VLSI technology is the increasing complexity of the design and manufacturing processes. As devices become more complex, the number of transistors and interconnects on a chip increases, which makes it more difficult to design and manufacture the chip. To address this challenge, new design and simulation tools are being developed that can help engineers optimize the performance and power consumption of their designs.

Another challenge facing the future of VLSI technology is the increasing demand for energy-efficient devices. As devices become more ubiquitous, the amount of energy they consume becomes a critical factor in their design and use. To address this challenge, new power management techniques are being developed that can optimize the energy consumption of VLSI devices, while maintaining their performance.

The historical increase in IC processing power has had a significant impact on how we produce, process, share, and store information. The capacity to reduce transistor size every few years is the driving force behind this amazing increase. The Moore's law phenomenon has persisted for the last 50 years. Thanks to technical advancements, Moore's law has frequently been shown to be alive and well. The CMOS scaling plan, however, is expected to come to an end in one or eight years when it will no longer be feasible to further reduce transistor size due to cost concerns. The device community is now pursuing a number of post-CMOS contenders, and this article examines their promise and constraints [1] .

DISCUSSION

The minimal voltage necessary to switch a transistor between an on-state and an off-state determines the transistor's capacity to scale its supply voltage. This attribute is measured using the sub-threshold slope (SS). For example, a lower SS allows the transistor to be powered on with a lower supply voltage so the same off existing [2], [3] . Assuming k is the Proportionality constant, T is the heat capacity, and q is the electron charge, the SS for MOSFETs must be more than ln(10) kT/q. The basic power/performance trade-off caused by this limitation, which results from the vacuum tube nature of the MOSFET ionic conduction, may be resolved if SS values much below the estimated 60-mV/decade cutoff could be

attained. There are other device types that have been suggested as having the potential to generate steep SS values, including ferroelectric-gate FETs, impact ionisation MOSFETs, tunnelling field-effect transistors (TFETs), and nanoelectromechanical systems (NEMS) devices. The quantitative verification of SS values in Transistors as low as 40 mV/decade at ambient temperature has been documented in a number of recent articles. Their mobility issues, unbalanced drive potential, bias independent SS, and higher statistical fluctuations in contrast to conventional MOSFETs are these so-called "steep" devices' primary drawbacks [4].

Spintronics is a technique that uses the spin direction of nanomagnets as the state variable to create spin devices. Over CMOS, spintronics provides advantages such as nonvolatility, fewer devices, and the possibility for ou alors computer networks. The nonvolatility of Spintronics devices allows for instantaneous processor wake-up and shut-down, which might significantly lower static power usage. In addition, it may make it possible to create brand-new processor-in-memory or logic-in-memory designs that are not feasible with silicon technology. Although still in its infancy, spintronics research has been gathering steam over the last 10 years due to the possibility that spinning devices might circumvent the CMOS scaling power limitation by providing a whole new computing paradigm. A variety of post-CMOS spintronic devices, and including logic, spin signal devices, applied field magnets for logic applications, stator flux torque magnetoresistive RAM (STT-MRAM), and spin-Hall torque (SHT) MRAM, have made progress in recent years toward demonstration. But in order for spintronics to be a practical post-CMOS device platform, scientists need to figure out how to do away with the transistors needed to control the clock and energy storage signals. Otherwise, CMOS technology would always have a limit on the performance. The relatively high active power, close connector distance, and complicated manufacturing process of spintronic devices are still issues [5], [6] .

Distributed large area (cm2-to-m2) electronic systems based on flexible thin-film transistor (TFT) technology are attracting a lot of interest because of their special characteristics, including mechanical conformability, low temperature processability, large area coverage, and low fabrication costs. Different kinds of flexible TFTs have the potential to either allow applications that conventional silicon-based technology could not support or outperform them in terms of cost per area. Due to the poor carrier mobility, flexible electronics cannot match the performance of silicon-based ICs. Instead, by providing dispersed sensor systems across a vast region with middling performance, this technology is aimed to complement them (less than 1 MHz). Flexible TFTs are now being developed using inkjet or roll-to-roll printing methods for low-cost manufacture, making product-level implementations possible. Despite these hopeful new discoveries, fabricating flexible electronic devices is very difficult because to the poor mobility and great sensitivity to processing conditions.

while the other accommodates the partial results at present being computed. A fter the evaluation of one round is completed, their roles are swapped. Simple as it is, this approach unfortunately requires twice as much memory as needed to store one set of intermediate data. A more efficient technique is in-place computation, whereby some of the input data are immediately overwritten by the freshly computed values. In-place computation may cause data item s to get scrambled in mem ory, though, which necessitates corrective action. Problem s amenable to in-place computation combined with memory unscrambling include the FFT and the Viterbi algorithm.

DDGs as regular as this offer ample room for devising a range of architectures that represent diverse compromises between a single-ALU microprocessor and a hardwired data pipeline maximised for throughput [7], [8] . Providing a limited degree of scalability to accommodate FFTs of various sizes is not overly difficult either. Favorable conditions similar to these are found in many more applications including, among others, transversal filters (repetitive multiply–accumulate operations), correlators (idem), lattice filters (identical stages), and block cyphers (cascaded rounds). Figure 1: illustrate the Large Scale Integrated Circuit.

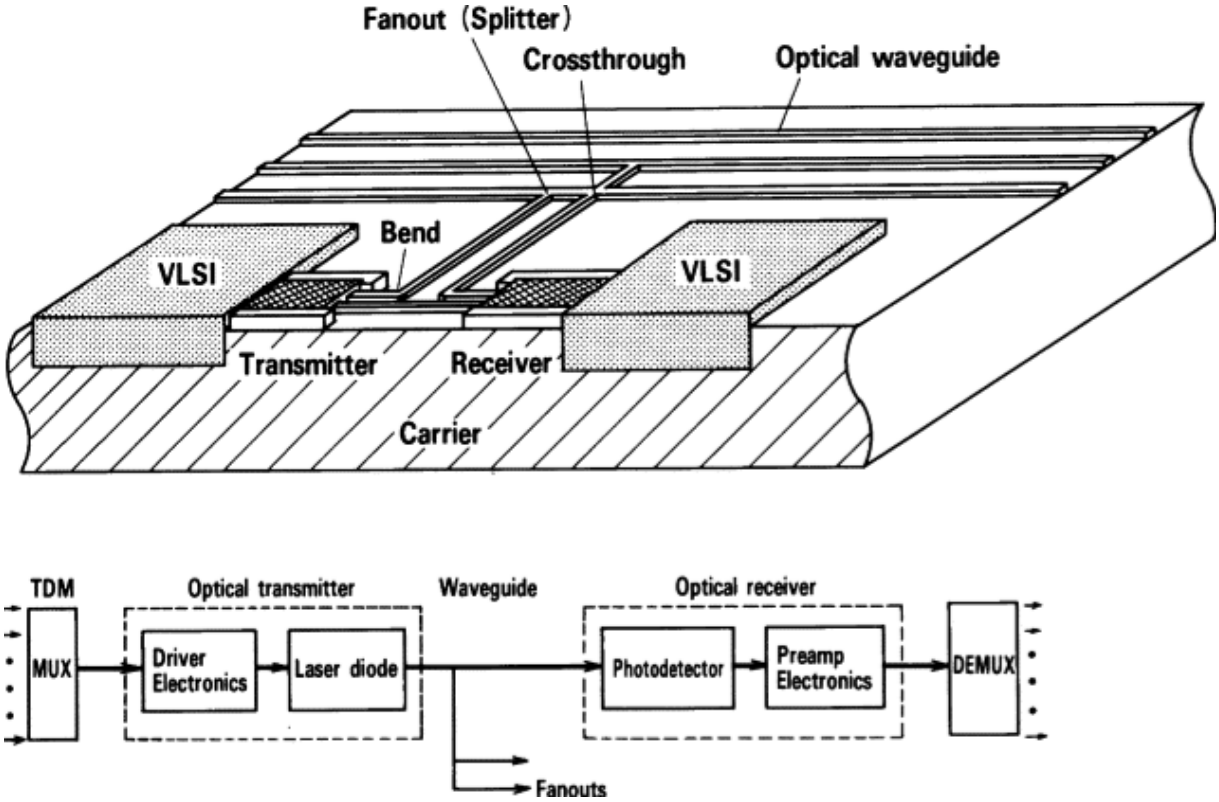


Figure 1: illustrate the Large Scale Integrated Circuit.

So far, we have come up with four equivalence transforms, namely iterative decomposition, Pipelining, Replication, and Time-sharing. A roadmap illustrating the four universal transforms for tailoring combinational hardware. Only a subset of all possible architectural configurations Greatly simplified by abstracting from register overhead (Areg = 0, treg = 0), which also implies not making any difference between RAMs and flip-flops (ARAM = Aff · #bits, tRAM = tff), assuming ideal iterative decomposition and ideal time-sharing, and ignoring any overhead associated with control and/or data distribution and collection.

All four architectural transforms discussed so far have one thing in common. Whether and how to apply them for maximum benefit can be decided from a DDG's connectivity and weights alone, no matter what operations the vertices stand for. In what follows, we will call any architectural reorganization that exhibits this property a universal transform. The practical consequence is that any computational flow qualifies for reorganisation by way of universal transforms. This also implies that any two computations the DDGs of which are isomorphic can be solved by the same architecture just with the vertices interpreted

differently. More on the negative side, universal transforms have a limited impact on the flow of computation because the number and precedence of operations are left unchanged.

As many computational problems ask for more specific and more profound forms of reorganisation in order to take full advantage of the situation at hand, one cannot expect to get optimum results from universal transforms alone. Rather, one needs to bring in knowledge on the particular functions involved and on their algebraic properties. Architectural reorganisations that do so are referred to as operation-specific or algebraic transforms.

Assuming the availability of 2-way minimum operators, this immediately suggests a chain structure such as the one depicted in fig.2.22a for I = 8. The delay along the longest path is tmin and increases linearly with the number of terms. As the 2-way minimum function is associative, the DDG lends itself to being rearranged into a balanced tree. The longest path is thereby shortened from I − 1 to log2 I operations, which makes the tree a much better choice, especially for large values of I. The number of operations and the circuit's size remain the same. The conversion of a chain of operations into a tree, as in the above example, is specifically referred to as tree-height minimization. As a side effect, this architectural transform often has a welcome impact on energy efficiency. This is because glitches die out more rapidly and are more likely to 4 2 While it is true that the number of DDG vertices may change, this is merely a consequence of viewing the original operations at a different level of detail. It goes without saying that many more algebraic laws can be put to use for improving dedicated architectures. Distributivity helps to replace the computation of (a2 − 2ax + x2) by the more economic form of (a − x) 2 , for instance, and is instrumental in exploiting known symmetries in coefficient sets. Together with commutativity, distributivity is also at the heart of distributed arithmetic. Horner's scheme serves to evaluate polynomials with a minimum number of multiplications, the principle of superposition holds in linear systems, the De Morgan theorem helps in optimising Boolean networks, and so on. As a rule, always ask yourself what situation-specific properties might be capitalised on. The transforms discussed in this text just represent the more common ones and are by no means exhaustive.

Iterative decomposition, pipelining, replication, and time-sharing are based on the DDG as a graph and make no assumptions on the nature of computations carried out in its vertices, which is why they are qualified as universal. The associativity transform, in contrast, is said to be an algebraic transform because it depends on the operations involved being identical exposes another difficulty of system-level design that has its roots in the highly heterogeneous nature of electronic systems. At various points, some fairly abstract design description must be propagated from one software tool to the next. Yet, there are no mathematical formalisms and agreed-on computer languages of sufficient scope to capture a sufficient portion of a system, let alone a system as a whole. The practical consequences are that some specifications need to be manually restated several times, that simulations do not extend over the entire system, and that certain aspects are being lost in the process.

Algorithm design

The central theme is to meet the data and/or signal processing requirements defined before with a series of computations that are streamlined in view of their implementation in hardware.

The subsequent assignments are part of algorithm design:

- Coming up with a collection of suitable algorithms or computational paradigms.
- Cut down computational burden and memory requirements.
- Find acceptable compromises between computational complexity and accuracy.
- Analyze and contain effects of finite word-length computation.
- Decide on number representation schemes.
- Evaluate alternatives and select the one best suited for the situation at hand.
- Quantify the minimum required computational resources in terms of memory, word widths, arithmetic and logic operations, and their frequencies of occurrence.

Algorithm design culminates in a bit-true software model which is indispensable for checking figures of merit relevant for the application at hand, e.g. signal-to-noise ratio, coding gain, data compression factor, error rate, and the like against specifications. Architecture design. VLSI architects essentially decide on the necessary hardware resources and organize their interplay in such a way as to implement a known computational algorithm under the performance, cost, power, and other constraints imposed by the target application.

The hardware arrangement they have to come up with must capture the essential structural characteristics of the future circuit but, at the same time, abstracts from implementation details. Still, architecture design also implies selecting a target technology and taking into account its possibilities and limitations. Architecture design starts from fairly abstract notions of a circuit's functionality and gradually proceeds to more detailed representations. The process is understood to happen in two substages, namely high-level architecture design and register transfer-level design. The former involves the following.

The term "computational paradigm" has been chosen to include finite state machines, cellular automata, neural networks, fuzzy logic, and other computational schemes that are not necessarily covered by the word "algorithm" as it is normally understood in the context of software engineering. Take this as an analogy from everyday life. Assume you were given the recipe for a fantastic cake by your grandmother and you were now to make a business out of it by setting up a bakery to mass-produce the cake.

The recipe corresponds to the algorithm or software model that specifies how the various ingredients must be processed in order to obtain the final product. Architecture design can then be likened to deciding on the mixers, kneaders, ovens, and other machines for processing the ingredients, and to planning the material flow in an industrial bakery. Observe that you will arrive at different factory layouts depending on the quantity of cakes that you intend to produce and depending on the availability and costs of labour and equipment.

The result is captured in a high-level block diagram that includes datapaths, controllers, memories, interfaces, and key signals. A preliminary floorplan is also being established. Verification of an architecture typically occurs by way of simulations, where each major building block is represented by a behavioural model of its own. The work is then carried down to the more detailed register transfer level (RTL) where the circuit gets modelled as a collection of storage elements interconnected by purely combinational subcircuits. Relevant issues at this stage include • How to implement arithmetic and logic units (e.g. ripple-carry, carry-lookahead, carry-select) [9], [10] .

CONCLUSION

VLSI technology has already revolutionized the electronics industry and will continue to do so in the future. As feature sizes shrink and designs become more complex, new challenges and limitations will arise, but researchers are already working on solutions to address these issues. The future of VLSI technology is bright, with potential advancements in areas such as 3D integration, new materials, photonics, and xeromorphic computing. These advancements will enable the creation of a wide range of new applications and technologies, from artificial intelligence to environmental monitoring.

REFERENCES

[1] W. Li and J. C. S. Woo, "Vertical P-TFET with a P-Type SiGe Pocket," *IEEE Trans. Electron Devices*, 2020, doi: 10.1109/TED.2020.2971475.

[2] B. K. Kaushik and M. K. Majumder, "Interconnects," *SpringerBriefs Appl. Sci. Technol.*, 2015, doi: 10.1007/978-81-322-2047-3_1.

[3] H. R. Mahdiani, S. M. Fakhraie, and C. Lucas, "Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors," *IEEE Trans. Neural Networks Learn. Syst.*, 2012, doi: 10.1109/TNNLS.2012.2199517.

[4] X. Chen, G. Liu, N. Xiong, Y. Su, and G. Chen, "A Survey of Swarm Intelligence Techniques in VLSI Routing Problems," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.2971574.

[5] H. Tang, G. Liu, X. Chen, and N. Xiong, "A survey on steiner tree construction and global routing for VLSI design," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.2986138.

[6] D. Z. Pan, B. Yu, and J. R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2013, doi: 10.1109/TCAD.2013.2276751.

[7] F. D. Broccard, S. Joshi, J. Wang, and G. Cauwenberghs, "Neuromorphic neural interfaces: From neurophysiological inspiration to biohybrid coupling with nervous systems," *Journal of Neural Engineering*. 2017. doi: 10.1088/1741-2552/aa67a9.

[8] P. Rössler and R. Höller, "Programmable logic devices – key components for today's and tomorrow's electronic-based systems," *Elektrotechnik und Informationstechnik*, 2020, doi: 10.1007/s00502-019-00781-w.

[9] F. B. N. Al Amin, N. Ahmad, and S. H. Ruslan, "Low power design of ultra wideband PLL using 90 nm CMOS technology," *Indones. J. Electr. Eng. Comput. Sci.*, 2020, doi: 10.11591/ijeecs.v20.i2.pp727-735.

[10] J. Zhang *et al.*, "Robust digital VLSI using carbon nanotubes," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2012, doi: 10.1109/TCAD.2012.2187527.

CHAPTER 13

EQUIVALENCE TRANSFORMS FOR RECURSIVE COMPUTATIONS

Dr. Vikas Sharma, Assistant Professor
Department of Computer Science and Engineering, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id- vikass.soeit@sanskriti.edu.in

ABSTRACT:

Recursive computations are a common type of computation used in many different areas of computer science, such as algorithms, programming, and artificial intelligence. Recursive computations are often used to perform computations that would be too complex or too difficult to perform using traditional sequential algorithms. Recursive computations can also be computationally expensive and can be difficult to optimize for performance. One approach to optimizing recursive computations is to use equivalence transforms, which can help to simplify the computations and reduce the computational complexity.

KEYWORDS:

Algorithms, Artificial Intelligence, Designs, Recursive Computation, Memory.

INTRODUCTION

Recursive computations are a common type of computation used in many different areas of computer science, such as algorithms, programming, and artificial intelligence. Recursive computations are often used to perform computations that would be too complex or too difficult to perform using traditional sequential algorithms. However, recursive computations can also be computationally expensive and can be difficult to optimize for performance. One approach to optimizing recursive computations is to use equivalence transforms, which can help to simplify the computations and reduce the computational complexity [1].

Equivalence transforms are mathematical transformations that can be used to simplify or optimize a computation without changing its result. Equivalence transforms are based on the principle of equivalence, which states that two expressions are equivalent if they produce the same result for all possible inputs. Equivalence transforms can be used to simplify or optimize computations by replacing one expression with an equivalent expression that is simpler or more efficient to compute. In the context of recursive computations, equivalence transforms can be used to simplify or optimize the recursion, making it more efficient or easier to understand. There are several different types of equivalence transforms that can be used for recursive computations, including substitution, iteration, and memoization.

Substitution

Substitution is a type of equivalence transform that involves replacing one expression with another equivalent expression. In the context of recursive computations, substitution can be

used to replace a recursive expression with an equivalent non-recursive expression, or to replace a complex recursive expression with a simpler equivalent expression.

This function is recursive because it calls itself with a smaller input. However, the recursion can be replaced with an equivalent non-recursive expression using substitution. We can define a new function that computes the factorial using a loop instead of recursion. This function is similar to the original recursive function, but it uses a dictionary to cache the results of previous computations. If the result for a particular input has already been computed and cached, the function returns the cached result instead of recomputing it. This can significantly reduce the computational complexity of the function, especially for large inputs.

Equivalence transforms can be used to optimize a wide range of recursive computations, from simple mathematical functions to complex machine learning algorithms. However, it is important to note that not all recursive computations can be optimized using equivalence transforms, and the effectiveness of a particular transform may depend on the specific properties of the computation. In addition, some transforms may introduce new computational complexities or trade-offs that must be carefully considered.

The outcome is a set of more detailed diagrams that include every single register, memory, and major block of combinational logic. As opposed to gate-level schematics, however, combinational functions are specified in behavioural rather than structural terms. Simulations are instrumental in debugging the RTL code. The floorplan is refined on the basis of the more detailed data that are now available and compared against the die size and cost targets for the final product. This is also the point to decide on the most appropriate design level synthesis, schematic entry, hand layout for each circuit block.

The delays and energy-dissipation figures associated with the various computational and storage operations are being calculated. Subcircuits that are found to limit performance during pre-layout analysis are identified and redesigned or reoptimized where possible. The result is a complete set of gate-level schematics and/or netlists validated by electrical rule check (ERC), logic simulation, timing verification, and power estimation [2].

Improvement of testability. A malfunctioning IC is the result of design flaws, fabrication defects, or both. Special provisions are necessary to ascertain the correct operation of millions of transistors enclosed in a package with a couple of hundred pins at most. Design for test (DFT) implies improving the controllability and observability of inner circuit nodes by adding auxiliary circuitry on top of the payload logic.

DISCUSSION

Equivalence transforms are an important tool for analyzing recursive computations. In this discussion, we will explore what equivalence transforms are, how they can be used to analyze recursive computations, and some specific examples of how they can be applied.

To start, let's define what we mean by a recursive computation. A recursive computation is a computation that involves repeatedly applying a function to its own output. For example, the factorial function is defined recursively as follows:

scssCopy code

factorial(n) = n * factorial(n-1)

This definition states that the factorial of a number n is equal to n times the factorial of n-1. This definition is recursive because it defines the function in terms of itself. To compute the factorial of a number, we apply the factorial function to a smaller number, until we reach the base case of factorial(0) = 1.

Recursive computations can be challenging to analyze because they involve an infinite number of steps. In the case of the factorial function, we need to apply the function n times to compute factorial(n). This can be difficult to reason about and can lead to inefficiencies in computation.

Equivalence transforms provide a way to simplify recursive computations by transforming the function into an equivalent form that is easier to analyze. The idea behind an equivalence transform is to replace the original function with a new function that computes the same result, but in a different way. This new function may be simpler, more efficient, or more amenable to analysis than the original function.

There are several different types of equivalence transforms that can be applied to recursive computations. In this discussion, we will focus on three: unfolding, folding, and fusion.

Unfolding is an equivalence transform that expands a recursive function into a non-recursive form by repeatedly applying the definition of the function. The idea behind unfolding is to replace the recursive call with an equivalent expression that does not involve the function itself. This can make it easier to reason about the computation and can often lead to more efficient code.

Let's consider an example of unfolding. Suppose we have the following recursive function that computes the nth Fibonacci number:

scssCopy code

fib(n) = if n < 2 then n else fib(n-1) + fib(n-2)

To compute fib(n), we need to compute fib(n-1) and fib(n-2). We can use this fact to unfold the function as follows:

scssCopy code

fib(n) = if n < 2 then n else fib(n-1) + fib(n-2) = if n < 2 then n else (if n-1 < 2 then n-1 else fib(n-2) + fib(n-3)) + (if n-2 < 2 then n-2 else fib(n-3) + fib(n-4)) = if n < 2 then n else if n-1 < 2 then n-1 else if n-2 < 2 then n-2 else fib(n-2) + fib(n-3) + fib(n-3) + fib(n-4) = ...

In this way, we can keep expanding the function until we reach the base cases of fib(0) = 0 and fib(1) = 1. This gives us a non-recursive form of the function that can be used to compute Fibonacci numbers more efficiently.

Folding is the inverse of unfolding. It is an equivalence transform that collapses a non-recursive form of a function into a recursive form by replacing repeated computations with function calls. The idea behind folding is to replace the repeated computations with function calls to a helper function that computes the repeated computation only once.

In a procedure referred to as fault grading, testability is rated by relating the number of fabrication defects that can in fact be detected with a test vector set under consideration to the total number of conceivable faults. Both the test circuitry and the test patterns are iteratively refined until a satisfactory fault coverage is obtained.

Physical design. Physical design addresses all issues of arranging the multitude of subcircuits and devices along with their interconnections on a piece of semiconductor material. Floorplanning is concerned with organising the major circuit blocks into a rectangular area as small as possible while, at the same time, limiting the effects of interconnect delays on the chip's performance. 19 Chip-level power and clock distribution are also to be dealt with. A padframe must be generated to hold the bond pads and the top-level layout blocks [3]. During the subsequent place and route (P&R) steps, each cell gets assigned a specific location on the die before the courses of myriads of metal wires that are to carry electrical signals between those cells get defined. It is often necessary to reoptimize the circuit logic as a function of the estimated interconnect delays that become available during the process. The final phaseStandard techniques include block isolation, scan testing, and BIST. Block isolation makes ma jor circuit blocks accessible from outside a chip with the aid of extra multiplexers so that stimuli can be applied and responses evaluated via package pins while in test mode. Scan testing is to be outlined. The idea behind built-in self-test (BIST) is to move stimuli generation and response checking onto the chip itself, and to essentially output a "go/no go" result [4]. BIST and block isolation are popular for testing on-chip memories. As DFT, test vector preparation, and autom ated test equipment (ATE) are not part of this text, the reader is referred to the specialised literature such as, for instance. Floorplanning makes part of physical design much as layout design does. What is the difference then? As an analogy, floorplanning is concerned with the partitioning of a flat into ro om s and hallways whereas layout design deals with tiny geometric patterns on a carpet.

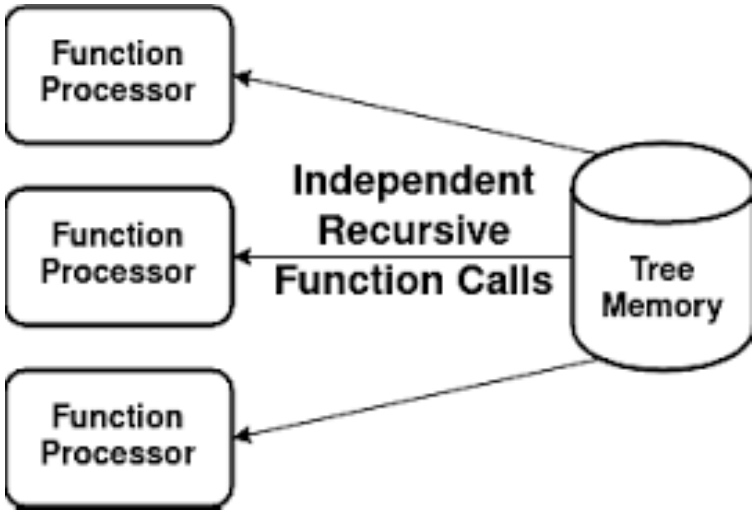


Figure 1: illustrate the Tree-Based Hardware Recursion for Divide-and-Conquer Algorithms.

As no customer is willing to pay for fabricated parts that do not conform with this requirement, the vendor wants to make sure the design is consistent with good engineering practice and with company-specific guidelines before doing so. DRC, manufacturability,

ERC, LVS, post-layout simulation, and fault coverage are routinely examined. Inspection often extends to timing verification, clocking discipline, power and clock distribution, circuit design style, test structures, and more [4].

In reality, the separation into individual subtasks is not as nice and clear. Various side effects of deep submicron technologies and the quest for optimum results make it necessary for most software tools to work across several levels of abstraction. As an example, it is no longer possible to place and route a gate-level netlist without adapting the circuit logic as a function of the resulting layout parasitics and interconnect delays. In the drawing, this gets reflected by the joint refinement of layout data and netlists.

Only ideally does design occur as a linear sequence of steps. Some back and forth between the various subtasks is inevitable to obtain a truly satisfactory result. Also, not all design stages are explicitly covered in every IC development project. Depending on the circuit's nature, fabrication depth, and design level, some of the design stages are skipped or outsourced, i.e. delegated to specialists at third-party companies. The design of a simple glue logic chip, for instance, begins at the logic level as there are no algorithmic or architectural questions to deal with. Models of industrial collaboration are to be discussed floorplan overall behavioural simulation inp. to outp. mapping block-level behavioural simul. transaction-based register transfer level simulation cycle-true simulation gate-level event-driven substitution of detailed layout for cell abstracts chip detailed layout extraction of devices and interconnect layout versus schematic (LVS) substitution of detailed circuits for cell icons automatic layout merge to IC manufacturing behavioral modelling (algorithm with software model data formats) gate-level netlist placement and gate netlist RTL design incl.

Macrocell preparation logic design and optimization estimation of die size and major cost factors preliminary power estimationtransistor-level netlist back-annotated extraction of cell abstracts and interconnect layout versus schematic (LVS) layout/design rule check (DRC)post-layout timing verification logic simulation post-layout event-driven calculation delay analysis signal integrity gate-level netlist back-annotated cell and wire delays analysis power grid high-level synthesis manufacturability analysis DRC and/or floorplanning, and pinout package selection padframe constr., power distribution, initial placement drawing of bonding diagram to IC packaging bonding diagram reoptimization and rebuffering of logic clock tree insertion placement and gate netlist rebuffering, hold time fixing, and rerouting gate-level netlist final preliminary abstract layout [5], [6].

DESIGN FLOW IN DIGITAL VLSI

Note the presence of angular and rounded boxes. While angular boxes refer to construction activities, the rounded ones stand for analysis and verification steps. A backward arrow implies that any problem uncovered during such an analysis triggers corrective action by the designer. The results from construction steps are subject to immediate verification, which is typical for VLSI. The reason is that correcting a mistake becomes more and more onerous the further the design process has progressed. Correcting a minor functional bug after layout design, for instance, would require redoing several design stages and would waste many hours of labour and computer time. Also, a functional bug can be uncovered more effectively from a behavioural or RTL model than from a post-layout transistor-level netlist because simulation speed is orders of magnitude higher and because automatic response checking is much easier to implement for logic and numeric data types than for analogue waveforms.

A critical point is reached when first silicon is going to be produced. While it is possible to cut and add wires using advanced and expensive equipment such as focused ion-beam (FIB) technology to patch a malfunctioning prototype, there is virtually no way to fix bugs in volume production. Depending on the circuit's size, fabrication depth, process, and manufacturer, expenses somewhere between 12 kUSD and 1 MUSD are involved with preparation of photomasks, tooling, wafer processing, preparation of probe cards and evaluation of preproduction samples. Any design flaw found after prototype fabrication thus implies the waste of important sums of money.

To make things worse, with turnaround times ranging between two weeks and three months, a product's arrival on the market is delayed so much that the chip is likely to miss its window of opportunity. Observation 1.2. Redesigns are so devastating for the business that the entire semiconductor industry has committed itself to "first-time-right" design as a guiding principle. To avoid them, VLSI engineers typically spend much more time verifying a circuit than actually designing it.

It also includes a number of forward arrows that bypass one or two construction steps. They suggest how electronic design automation, cell libraries, and purchased know-how help speed up the design process. Keeping pace with the breathtaking progress of fabrication technology is in fact one of the major challenges for today's VLSI designers.

While there is not too much of a difference in the front-end flow, back-end design for field programmable logic (FPL) differs. The preliminary gate-level netlist obtained from HDL synthesis is mapped onto configurable blocks available in the target FPGA or CPLD device. After the EDA software has decided how to run all necessary interconnects using the wires, switches, and drivers available, the result is converted into a configuration bit stream for download into the FPL device. As FPGAs and CPLDs come with many diverse architectures, product-specific back-end tools made available by the FPL vendor are used for this procedure. Whoever has learned to design full-custom ICs is in an excellent position for designing semi-custom ICs and to design with field-programmable logic, but not necessarily the other way round[7], [8].

Library development occurs quite separately from actual IC design as cell-based circuits largely dominate VLSI. Cell libraries are typically licenced to IC developers by specialised library vendors since silicon vendors have largely withdrawn from this business. Once the set of prospective library cells has been defined functionally, library development proceeds in three major phases. Electrical design deals with implementing logic functions as transistor-level networks and with sizing the individual devices such as to find an optimum trade-off between performance, circuit complexity, and energy efficiency. During the subsequent layout design, the locations and geometric shapes of individual devices are defined along with the shapes of the wires running in between. The goal is to obtain leaf cells that are compact, fast, energy-efficient, suitable for automatic place and route (P&R), and that can be manufactured with maximum yield. Verification includes the customary ERC, DRC, manufacturability analysis, extraction, and LVS procedures. Next the electrical and timing parameters that are to be included in data sheets and simulation models of the cells are determined.

In order to protect their investments, most library vendors consider their library cells to be proprietary and are not willing to disclose how they are constructed internally. They supply

datasheets, icons, simulation models, and abstracts, but no transistor-level schematics and no layouts. Under this scheme, detailed layouts are to be substituted for all cell abstracts by the vendor before mask preparation can begin.

The VLSI industry long ago became entirely dependent on electronic design automation (EDA) software. There is not one single step that could possibly be brought to an end without the assistance of sophisticated computer programs. The sheer quantity of data necessary to describe a multi-million transistor chip makes this impossible. The design flow outlined in the previous section gives a rough idea of the variety of CAE/CAD programmes that are required to pave the way for VLSI and FPL design. While a few vendors can take pride in offering a range of products that covers all stages from systemlevel decision making down to physical layout, much of their effort tends to focus on relatively small portions of the overall flow for reasons of market penetration and profitability. Frequent mergers and acquisitions are another characteristic trait of the EDA industry. Truly integrated design environments and seamless design flows are hardly available off the shelf.

Also, the idea of integrating numerous EDA tools over a common design database and with a consistent user interface, once promoted as front-to-back environments, aka frameworks, has lost momentum in the marketplace in favour of point tools and the “best in class” approach. Design flows are typically pieced together from software components of various origins. 23 The presence of software tools, design kits, and cell libraries from multiple sources in conjunction with the absence of agreed-on standards adds a lot of complexity to the maintainance of a coherent design environment. Many of the practical difficulties with setting up efficient design flows are left to EDA customers.

30 Architectures of VLSI Circuits and can sometimes become a real nightmare. It is to be hoped that this trend will be reversed one day when customers are willing to pay more attention to design productivity than to layout density and circuit performance.

Field-programmable logic

The general idea behind programmable logic has been introduced. The goal of this section is to explain the major differences that separate distinct product families from each other. Key properties of any FPL device are fixed by decisions along two dimensions taken at development time. A first choice refers to how the device is being configured and how its configuration is stored electrically while a second choice is concerned with the overall organisation of the hardware resources available to customers. Customers, in this case, are design engineers who want to implement their own circuits in an FPL device[9], [10].

Configuration technologies

Static memory. The key element here is an electronic switch such as a transmission gate, a pass transistor, or a three-state buffer that gets turned “on” or “off” under control of a configuration bit. Unlimited reprogram ability is obtained from storing the configuration data in SRAM cells or in similar on-chip sub circuits built from two cross-coupled inverters. As a major drawback, the circuit must (re)obtain its entire configuration from outside whenever it is being powered up. The problem is solved in one of three possible ways, namely (a) by reading from a dedicated bit-serial or bit-parallel off-chip ROM, (b) by downloading a bit stream from a host computer, or (c) by long-term battery backup.

CONCLUSION

Equivalence transforms are a powerful tool for optimizing recursive computations. By using substitution, iteration, or memorization, it is possible to simplify or optimize a computation without changing its result. Equivalence transforms can be particularly useful for recursive computations, where they can help to reduce computational complexity and improve performance. However, the effectiveness of a particular transform may depend on the specific properties of the computation, and careful consideration is required to ensure that the transform produces the desired results.

REFERENCES

[1] G. Wang and S. Wang, "Recursive computation of Tchebichef moment and its inverse transform," *Pattern Recognit.*, 2006, doi: 10.1016/j.patcog.2005.05.015.

[2] Z. Cvetkovic and M. V. Popovic, "New fast recursive algorithms for the computation of discrete cosine and sine transforms," *IEEE Trans. Signal Process.*, 1992, doi: 10.1109/78.150010.

[3] B. Honarvar Shakibaei Asli, R. Paramesran, and C. L. Lim, "The fast recursive computation of Tchebichef moment and its inverse transform based on Z-transform," *Digit. Signal Process. A Rev. J.*, 2013, doi: 10.1016/j.dsp.2013.05.004.

[4] Y. Liu, L. Tong, W. X. Zhu, Y. Tian, and B. Gao, "Impedance measurements of nonuniform transmission lines in time domain using an improved recursive multiple reflection computation method," *Prog. Electromagn. Res.*, 2011, doi: 10.2528/PIER11042406.

[5] P. Han, X. He, Y. Wang, H. Ren, X. Peng, and Z. Shu, "Harmonic analysis of single-phase neutral-point-clamped cascaded inverter in advanced traction power supply system based on the big triangular carrier equivalence method," *Energies*, 2018, doi: 10.3390/en11020431.

[6] G. Bacchiani, M. Patander, A. Cionini, and D. Giaquinto, "Parking slots detection on the equivalence sphere with a progressive probabilistic hough transform," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018. doi: 10.1109/ITSC.2017.8317764.

[7] Y. H. Jiang, G. X. Li, H. M. Li, L. Li, and G. P. Zhang, "Effect of flame inherent instabilities on the flame geometric structure characteristics based on wavelet transform," *Int. J. Hydrogen Energy*, 2018, doi: 10.1016/j.ijhydene.2018.03.141.

[8] J. Ko, J. Son, C. L. Myung, and S. Park, "Comparative study on low ambient temperature regulated/unregulated emissions characteristics of idling light-duty diesel vehicles at cold start and hot restart," *Fuel*, 2018, doi: 10.1016/j.fuel.2018.05.144.

[9] M. Fukuda, N. Kobayashi, and T. Nishioka, "Operator product expansion for conformal defects," *J. High Energy Phys.*, 2018, doi: 10.1007/JHEP01(2018)013.

[10] G. Calcagni, L. Modesto, and G. Nardelli, "Initial conditions and degrees of freedom of non-local gravity," *J. High Energy Phys.*, 2018, doi: 10.1007/JHEP05(2018)087.

CHAPTER 14

DEVELOPING AN ADEQUATE SIMULATION STRATEGY

Dr. Rajbhadur Singh, Assistant Professor
Department of Computer Science and Engineering, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id- rajbhadurs.soeit@sanskriti.edu.in

ABSTRACT:

Simulation is the process of creating an artificial environment to study the behavior of a system or process over time. It is used in a variety of fields, including engineering, economics, and social sciences, to study complex systems that are difficult or impossible to analyze using analytical or experimental methods. Developing an adequate simulation strategy involves a number of steps, including defining the problem, selecting the appropriate simulation tool, designing the simulation model, validating the model, and analyzing the results.

KEYWORDS:

Adequate, Simulation Strategy, Models, Social Sciences, Validating Model.

INTRODUCTION

The first step in developing a simulation strategy is to define the problem to be studied. This involves identifying the system or process of interest, defining the objectives of the simulation, and specifying the inputs and outputs of the system. The problem should be defined in sufficient detail to ensure that the simulation accurately represents the system or process being studied.

Selecting the Appropriate Simulation Tool The second step is to select the appropriate simulation tool for the problem at hand. There are a variety of simulation tools available, each with their own strengths and weaknesses [1], [2] . Some of the factors that should be considered when selecting a simulation tool include the complexity of the system, the desired level of detail in the simulation, the availability of data on the system, and the computational resources available.

Designing the Simulation Model Once the simulation tool has been selected, the next step is to design the simulation model. This involves creating a mathematical representation of the system or process being studied, specifying the input and output variables, and defining the relationships between them. The model should be designed to capture the key features of the system or process, while also being computationally tractable.

Validating the Model The next step is to validate the simulation model. This involves comparing the results of the simulation to real-world data or to analytical solutions, if available. The model should be validated under a range of conditions to ensure that it accurately represents the system or process being studied. If the model is found to be inaccurate or incomplete, it may need to be revised or refined.

The final step in developing a simulation strategy is to analyze the results of the simulation. This involves interpreting the output variables and identifying patterns or trends in the data. The results should be used to draw conclusions about the behavior of the system or process being studied and to inform decision-making.

In addition to these steps, there are a number of best practices that can help to ensure the development of an adequate simulation strategy.

Overall, developing an adequate simulation strategy requires a careful and systematic approach that involves defining the problem, selecting the appropriate simulation tool, designing the simulation model, validating the model, and analyzing the results. By following best practices and being mindful of the limitations and assumptions of the simulation, it is possible to develop a simulation strategy that provides valuable insights into the behavior of complex systems and processes [3], [4]. In addition to the steps and best practices outlined above, there are a number of additional considerations that should be taken into account when developing a simulation strategy.

DISCUSSION

Configurability is very helpful for debugging. It permits one to probe inner nodes, to alternate between normal operation and various diagnostic modes, and to patch a design once a flaw has been located. Many RAM-based FPL devices further allow reconfiguring of their inner logic during operation, a capability known as in-system configuration (ISC) that opens a door towards configurable computing.

Electrically programmable read-only memories (EPROM) rely on special MOSFETs where a second gate electrode is sandwiched between the transistor's bulk material underneath and a control gate. The name floating gate captures the fact that this gate is entirely surrounded by insulating silicon dioxide material. An electrical charge trapped there determines whether the MOSFET, and hence the programmable link too, is "on" or "off". More precisely, the presence or absence of an electrical charge modifies the MOSFET's threshold voltage and so determines whether the transistor will conduct or not when a voltage is applied to its control gate during memory readout operations.

FPL configuration technologies (simplified, programming circuitry not shown). Switch steered by static memory cell MOSFET controlled by a charge trapped on a floating gate fuse and antifuse. Charging occurs by way of hot electron injection from the channel. That is, a strong lateral field applied between source and drain accelerates electrons to the point where they get injected through the thin dielectric layer into the floating gate. The necessary programming voltage in the order of 5 to 20 V is typically generated internally by an on-chip charge pump. Erasure of the charge is obtained by shining ultraviolet (UV) radiation on the chip, thereby causing the charges to leak away from the floating gate. The necessary quartz window in the plastic or ceramic package gives UV-erasable devices their unmistakable appearance but also renders the package rather expensive.

UV-erasable devices are non-volatile and immediately live at power-up, thereby doing away with the need for any kind of configuration-backup apparatus. Reprogramming necessitates removing the component from the circuit board and placing it into a special UV eraser,

however, which is undesirable and often altogether impossible. This explains why EPROMbased FPL devices much like the memories themselves have been superseded by parts that are more convenient to reconfigure[5], [6]. Figure 1: shows the A simulation model shows how individual differences affect major life decisions.

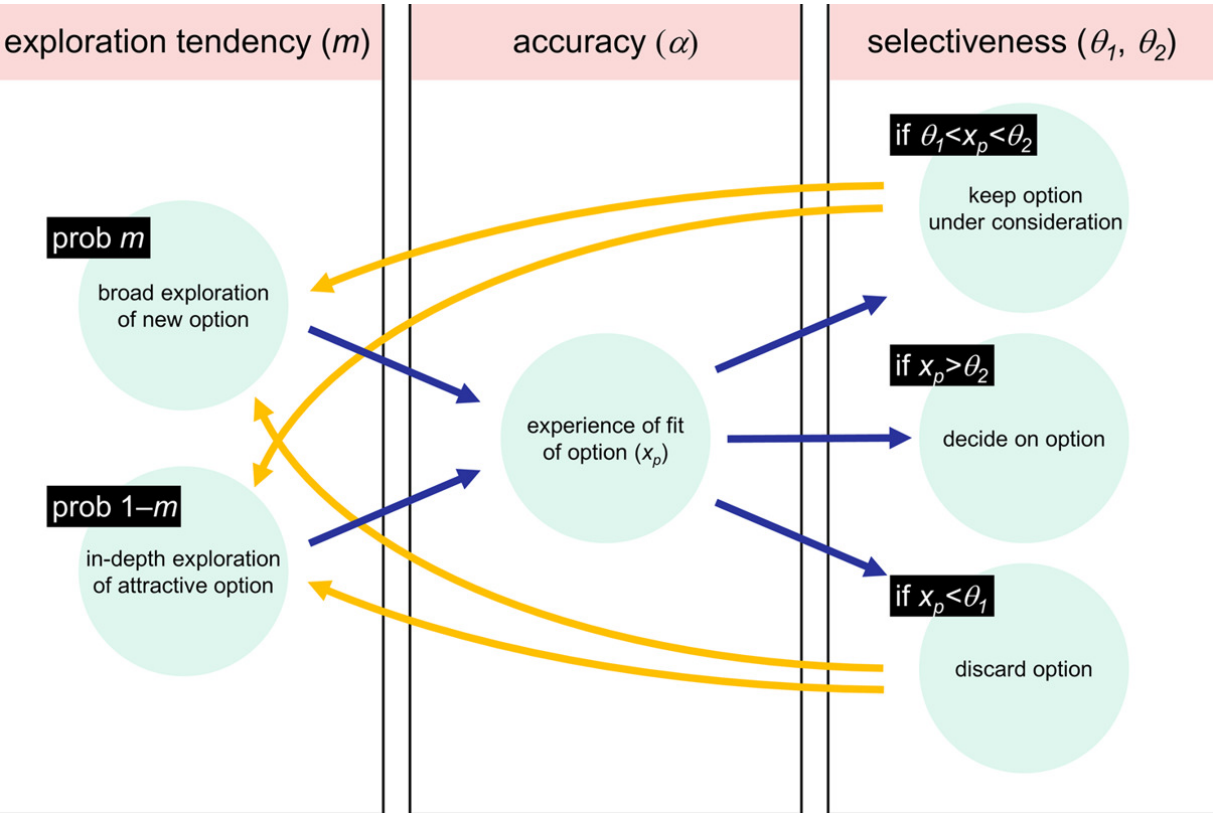


Figure 1: Shows the A simulation model shows how individual differences affect major life decisions.

Electrically erasable memory. EEPROM technology borrows from UV-erasable memories. The difference is that the electrons trapped on the floating gate are removed electrically by having them tunnel through the oxide layer underneath the floating gate without exposure to ultraviolet light, thereby making it possible to manufacture FPL devices that are non-volatile but nevertheless reconfigurable through their package pins [7]. The secret is a quantum-mechanical effect known as Fowler–Nordheim tunnelling that comes into play when a strong vertical field (8–10 MV/cm or so) is applied across the gate oxide.

Early electrically erasable devices were penalised by the fact that an EEPROM cell occupies about twice as much area as its UV-erasable counterpart because each bit cell includes a select transistor connected in series with the storage transistor. The flash memory technology prevalent today manages with a single floating-gate transistor per bit. The fact that erasure must occur in chunks, that is to say many bits at a time, is perfectly adequate in the context of FPL. Data retention times vary between 10 and 40 years [8]. Endurance of flash FPL is typically specified with 100 to 1000 configure–erase cycles, which is much less than for flash memory chips.

Fuse or antifuse. Fuses, which were used in earlier bipolar PROMs and SPLDs, are narrow bridges of conducting material that blow in a controlled fashion when a programming current

is forced through. Antifuses, such as those employed in today's FPGAs, are thin dielectrics separating two conducting layers that are made to rupture upon applying a programming voltage, thereby establishing a conductive path of low impedance.

In either case, programming is permanent. Whether this is desirable or not depends on the application. Full factory testing prior to programming of one-time programmable links is impossible for obvious reasons. Special circuitry is incorporated to test the logic devices and routing tracks at the manufacturer before the unprogrammed devices are being shipped. On the other hand, antifuses are only about the size of a contact or via and, therefore, allow for higher densities than reprogrammable links, see fig.1.15c and d. Antifuse-based FPL is also less sensitive to radiation effects, offers superior protection against unauthorised cloning, and does not need to be configured following power-up.

Organization of hardware resources

Simple programmable logic devices (SPLDs). Historically, FPL has evolved from purely combinational devices with just one or two programmable levels of logic such as ROMs, PALs, and PLAs. Flip-flops and local feedback paths were added later to allow for the construction of finite state machines, see fig.1.16a and b. Products of this kind continue to be commercially available for glue logic applications. Classic SPLD examples include the 18P8 (combinational) and the 22V10 (sequential).

The rigid two-level-logic-plus-register architecture in conjunction with the limited numbers of inputs, outputs, product terms, and flip-flops always restricted SPLDs to small applications. More scalable and flexible architectures had thus to be sought, and the spectacular progress of VLSI technology has made their implementation economically feasible from the late 1980s onwards. Two broad classes of hardware organisation prevail today.

Complex programmable logic devices (CPLDs) expand the general idea behind SPLDs by providing many of them on a single chip. Up to hundreds of identical subcircuits, each of which conforms to a classic SPLD, are combined with a large programmable interconnect matrix or network, see fig.1.16c. A difficulty with this type of organisation is that a partitioning into a bunch of cooperating SPLDs has to be imposed artificially on any given computational task, which benefits neither hardware nor design efficiency.

Depending on the manufacturer, products are known as complex programmable logic device (CPLD), programmable large-scale integration (PLSI), erasable programmable logic device (EPLD), and the like in the commercial world. Field-programmable gate arrays (FPGAs) have their overall organisation patterned after that of gate arrays. Many configurable logic cells are arranged in a two-dimensional array with bundles of parallel wires in between. A switchbox is present wherever two wiring channels intersect. Depending on the product, each logic cell can be configured so as to carry out some not-too-complex combinational operation, to store a bit or two, or both[9], [10].

While it is correct to think of alternating cells and wiring channels from a conceptual point of view, you will hardly be able to discern them under a microscope. The reason is that logic and wiring resources are superimposed for the sake of layout density in modern FPGA chips.

As opposed to traditional gate arrays, it is the state of programmable links rather than fabrication masks that decides on logic functions and signal routing. Parts with this organisation are being promoted under names such as field-programmable gate array (FPGA), logic cell array (LCA), and programmable multilevel device (PMD).The number of configurable logic cells greatly varies between products, with typical figures ranging between a few dozens and hundreds of thousands.

FPGA architectures are differentiated further depending on the granularity and capabilities of the configurable logic cells employed. One speaks of a fine-grained architecture when those cells are so simple that they are capable of implementing no more than a few logic gates and/or one bistable. For instance, each logic cell can be configured into a latch, or a flip-flop, or into almost any 3-input gate. As opposed to this, cells that are designed to implement combinational functions of four to six input variables and that are capable of storing two or more bits at a time are referred to as coarse-grained. The logic cell has 16 inputs and 11 outputs, and includes two programmable look-up tables (LUTs), two generic bistables that can be configured either into a latch or a flip-flop, a bunch of configurable multiplexers, a fast carry chain, plus other gates. Of course, the superior functional capabilities offered by a coarse-grained cell are accompanied by a larger area occupation.

The gate-level netlists produced by automatic synthesis map more naturally onto fine-grained architectures. The fact that fine-grained FPGAs and semi-custom ICs provide similar primitives further supports extensive reuse of design flows, HDL code, building blocks, and design. Incidentally note that FPL vendors refer to configurable logic cells by proprietary names. “Logic tile” is Actel’s term for their fine-grained cells whereas Xilinx uses the name “configurable logic block” (CLB) for their coarsegrained counterparts. Depending on the product family, one CLB consists of two or three LUTs plus two flip-flops or of several “slices”, each of which includes one LUT and one bistable. “Module” and “eCell”are commercial names used by other vendors.

It thus becomes practical to move back and forth between field- and maskprogrammed circuits with little overhead and to postpone any final commitment until fairly late in the design cycle. Conversely, fine-grained FPGAs tend to be more wasteful in terms of configuration bits and routing resources. Another reason that contributed to the popularity of coarse-grained FPGAs is that on-chip RAMs come at little extra cost when that architectural concept is combined with configuration from static memory. In fact, a reprogrammable LUT is nothing else than a tiny storage array. It is thus possible to bind together multiple logic cells in such a way as to make them act collectively like a larger RAM. As opposed to many other types of FPGAs, there is no compelling need to set aside special die areas for embedded SRAMs. In the occurrence of each of the two larger LUTs in each logic tile contributes another 16 bits of storage capacity.

In addition to FPL, field-programmable analogue arrays (FPAAs) began to appear on the market in the late 1990s. The next logical step was the extension to mixed-signal applications. Advanced products that combine configurable analogue building blocks with a micro- or digital signal processor and with analog-to-digital and digital-to-analog converters come quite close to the vision of fieldprogrammable systems on a chip. Vendors of field-programmable analogue and mixed-signal arrays include Anadigm, Actel, Cypress, Lattice, and Zetex FAS. Technical details on commercial FPL devices are distributed over thousands

of datasheets, help to keep track of products and manufacturers. More condensed background information is available from references such as. Capacity figures of semi-custom ICs and FPL may be confusing. As opposed to full-custom ICs, manufactured gates, usable gates, and actual gates are not the same. Manufactured gates indicate the total number of GEs that are physically present on a silicon die. A substantial fraction thereof is not usable in practise because the combinational functions in a given design do not fit into the available look-up tables exactly, because an FPL device only rarely includes combinational and storage resources with the desired proportions, and because of limited interconnect resources. The percentage of usable gates thus depends on the application. The actual gate count, finally, tells how many GEs are indeed put to service by a given design. The three figures frequently get muddled up, all too often in a deliberate attempt to make one product look better than its competitors in advertisements, product charts, and datasheets. Some FPL vendors prefer to specify the available resources using their own proprietary capacity units rather than in gate equivalents. It often pays to conduct benchmarks with a few representative designs before undertaking serious cost calculations and making a misguided choice. This also helps to obtain realistic timing figures that take into account interconnect delays. Exposes another difficulty of system-level design that has its roots in the highly heterogeneous nature of electronic systems. At various points, some fairly abstract design description must be propagated from one software tool to the next. Yet, there are no mathematical formalisms and agreed-on computer languages of sufficient scope to capture a sufficient portion of a system, let alone a system as a whole. The practical consequences are that some specifications need to be manually restated several times, that simulations do not extend over the entire system, and that certain aspects are being lost in the process. The central theme is to meet the data and/or signal processing requirements defined before with a series of computations that are streamlined in view of their implementation in hardware.

CONCLUSION

Developing an adequate simulation strategy is a complex and iterative process that requires careful attention to the problem being studied, the simulation tool being used, the model being designed, and the results being analyzed. By following best practices and considering the additional considerations outlined above, it is possible to develop a simulation strategy that provides valuable insights into the behavior of complex systems and processes.

REFERENCES

[1] A. C. Urquidi-Martín, C. Tamarit-Aznar, and J. Sánchez-García, “Determinants of the effectiveness of using renewable resource management-based simulations in the development of critical thinking: An application of the experiential learning theory,” *Sustain.*, 2019, doi: 10.3390/su11195469.

[2] F. Yassin, S. Razavi, M. Elshamy, B. Davison, G. Sapriza-Azuri, and H. Wheater, “Representation and improved parameterization of reservoir operation in hydrological and land-surface models,” *Hydrol. Earth Syst. Sci.*, 2019, doi: 10.5194/hess-23-3735-2019.

[3] Z. Luo, C. Sun, and Q. Dong, “A daylight-linked shading strategy for automated blinds based on model-based control and Radial Basis Function (RBF) optimization,” *Build. Environ.*, 2020, doi: 10.1016/j.buildenv.2020.106854.

[4] K. Hansson, M. Danielson, and L. Ekenberg, "A framework for evaluation of flood management strategies," *J. Environ. Manage.*, 2008, doi: 10.1016/j.jenvman.2006.12.037.

[5] P. Adhikari, H. G. Rao, and D.-I. M. Buderath, "Machine Learning based Data Driven Diagnostics & Prognostics Framework for Aircraft Predictive Maintenance," *10th Int. Symp. NDT Aerospace, Oct. 24-26, 2018, Dresden, Ger.*, 2018.

[6] S. M. H. Rostami, A. K. Sangaiah, J. Wang, and H. jin Kim, "Real-time obstacle avoidance of mobile robots using state-dependent Riccati equation approach," *Eurasip J. Image Video Process.*, 2018, doi: 10.1186/s13640-018-0319-1.

[7] O. Liberg, G. Chapron, P. Wabakken, H. C. Pedersen, N. Thompson Hobbs, and H. Sand, "Shoot, shovel and shut up: Cryptic poaching slows restoration of a large carnivore in Europe," *Proc. R. Soc. B Biol. Sci.*, 2012, doi: 10.1098/rspb.2011.1275.

[8] K. Vermeiren, M. Vanmaercke, J. Beckers, and A. Van Rompaey, "ASSURE: a model for the simulation of urban expansion and intra-urban social segregation," *Int. J. Geogr. Inf. Sci.*, 2016, doi: 10.1080/13658816.2016.1177641.

[9] S. M. Glasgow, Z. B. Perkins, N. R. M. Tai, K. Brohi, and C. Vasilakis, "Development of a discrete event simulation model for evaluating strategies of red blood cell provision following mass casualty events," *Eur. J. Oper. Res.*, 2018, doi: 10.1016/j.ejor.2018.03.008.

[10] J. F. Carias, L. Labaka, J. M. Sarriegi, and J. Hernantes, "An approach to the modeling of cyber resilience management," in *2018 Global Internet of Things Summit, GIoTS 2018*, 2018. doi: 10.1109/GIOTS.2018.8534579.

CHAPTER 15

CLOCKING OF SYNCHRONOUS CIRCUITS

Dr. Devendra Singh, Assistant Professor
Department of Computer Science and Engineering, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id- devendras.soeit@sanskri.edu.in

ABSTRACT:

Synchronous circuits are digital circuits that are designed to work with a clock signal. The clock signal is used to synchronize the operations of the various components of the circuit. In a synchronous circuit, all state changes occur at the rising or falling edge of the clock signal. This approach to designing digital circuits has become the dominant method due to its ability to simplify the design and analysis of circuits.

KEYWORDS:

Circuits, Clocks, Digital Circuit, Design, Operations.

INTRODUCTION

Clocking in synchronous circuits involves the use of a clock signal to control the timing of events in the circuit. The clock signal is generated by an oscillator or clock generator and is typically a square wave with a fixed frequency. The clock signal is then distributed to all the components of the circuit, and each component performs its operations in synchrony with the clock signal. The clock signal is an essential component of synchronous circuits, and the clocking methodology used can have a significant impact on the performance and reliability of the circuit. In this article, we will discuss the different clocking methodologies used in synchronous circuits, their advantages and disadvantages, and their impact on the performance and reliability of the circuit [1]–[3] .

1. Single-Phase Clocking

Single-phase clocking is the most common clocking methodology used in synchronous circuits. In this methodology, a single clock signal is used to control the timing of all the components of the circuit. All state changes occur at either the rising or falling edge of the clock signal.

Single-phase clocking is simple to implement, requires fewer components, and is easy to analyze. However, single-phase clocking has some disadvantages. The clock signal needs to have a high-frequency, which can be a challenge in low-power circuits. The clock signal also needs to be distributed across the entire circuit, which can result in timing skew and clock jitter.

2. Dual-Phase Clocking

Dual-phase clocking is a clocking methodology that uses two non-overlapping clock signals to control the timing of the circuit. In dual-phase clocking, each component of the circuit is

controlled by a separate clock signal. The two clock signals are generated from the same clock source and are shifted by 180 degrees.

Dual-phase clocking has several advantages over single-phase clocking. It eliminates the need for high-frequency clock signals, as each component of the circuit operates at half the frequency of the clock signal. Dual-phase clocking also eliminates clock skew and jitter, which can result in improved performance and reliability.

However, dual-phase clocking requires more components and is more complex to analyze than single-phase clocking. It can also be more challenging to implement in large circuits.

3. Multi-Phase Clocking

Multi-phase clocking is a clocking methodology that uses multiple non-overlapping clock signals to control the timing of the circuit. In multi-phase clocking, each component of the circuit is controlled by a separate clock signal. The number of clock signals used can vary depending on the complexity of the circuit. Algorithm design culminates in a bit-true software model which is indispensable for checking figures of merit relevant for the application at hand, e.g. signal-to-noise ratio, coding gain, data compression factor, error rate, and the like against specifications. Architecture design. VLSI architects essentially decide on the necessary hardware resources and organize their interplay in such a way as to implement a known computational algorithm under the performance, cost, power, and other constraints imposed by the target application [4].

The hardware arrangement they have to come up with must capture the essential structural characteristics of the future circuit but, at the same time, abstracts from implementation details. Still, architecture design also implies selecting a target technology and taking into account its possibilities and limitations. Architecture design starts from fairly abstract notions of a circuit's functionality and gradually proceeds to more detailed representations. The process is understood to happen in two substages, namely high-level architecture design and register transfer-level design. The former involves the following.

The term "computational paradigm" has been chosen to include finite state machines, cellular automata, neural networks, fuzzy logic, and other computational schemes that are not necessarily covered by the word "algorithm" as it is normally understood in the context of software engineering. Take this as an analogy from everyday life. Assume you were given the recipe for a fantastic cake by your grandmother and you were now to make a business out of it by setting up a bakery to mass-produce the cake.

The recipe corresponds to the algorithm or software model that specifies how the various ingredients must be processed in order to obtain the final product. Architecture design can then be likened to deciding on the mixers, kneaders, ovens, and other machines for processing the ingredients, and to planning the material flow in an industrial bakery. Observe that you will arrive at different factory layouts depending on the quantity of cakes that you intend to produce and depending on the availability and costs of labour and equipment.

DISCUSSION

Clocking is a critical aspect of synchronous digital circuits that is essential for ensuring reliable and correct operation. In this discussion, we will explore what clocking is, why it is important, and how it is implemented in synchronous circuits [5], [6].

To start, let's define what we mean by a synchronous circuit. A synchronous circuit is a digital circuit that operates based on a clock signal. The clock signal is a periodic waveform that provides a timing reference for the circuit. The clock signal is used to synchronize the operation of the circuit's internal logic elements, such as flip-flops and registers, to ensure that they change their state at the appropriate times.

The clock signal is typically generated by an external oscillator circuit and is provided to the synchronous circuit as a separate input signal. The frequency of the clock signal determines the rate at which the circuit operates. The clock frequency is often specified as a clock period, which is the time between successive rising edges (or falling edges) of the clock signal.

The clock signal is used to control the operation of the circuit's internal logic elements. Each logic element in the circuit is connected to the clock signal and changes its state only on the rising (or falling) edge of the clock signal. This ensures that all the logic elements in the circuit change their state at the same time and that their output signals are stable and valid for the next clock cycle.

Clocking is important for several reasons. First, it provides a timing reference that ensures that the circuit's internal logic elements operate in a synchronous manner. This simplifies the design of the circuit and ensures that the circuit operates correctly.

Second, clocking enables the circuit to operate reliably in the presence of noise and other disturbances. By synchronizing the operation of the logic elements to the clock signal, the circuit can reject transient signals that occur between clock cycles and ensure that the logic elements operate only when the input signals are stable and valid.

Finally, clocking enables the circuit to operate at a higher frequency than would be possible with asynchronous circuits. By synchronizing the operation of the logic elements to the clock signal, the circuit can operate at the same frequency as the clock signal, which is typically much higher than the frequency of the input signals.

Implementing clocking in a synchronous circuit involves several key aspects, including clock generation, clock distribution, clock skew, and clock gating.

Clock generation is the process of generating the clock signal that is used to synchronize the operation of the circuit's internal logic elements. The clock signal is typically generated by an external oscillator circuit, which generates a periodic waveform with a frequency that is equal to or greater than the desired clock frequency.

The clock signal is then provided to the synchronous circuit as a separate input signal. The circuit's internal logic elements are connected to the clock signal and change their state only on the rising (or falling) edge of the clock signal.

Clock distribution is the process of distributing the clock signal to all the internal logic elements in the circuit. The clock signal is typically distributed using a network of clock distribution lines that are designed to minimize clock skew.

Clock skew is the difference in arrival times of the clock signal at different points in the circuit. Clock skew can occur due to variations in the length of the clock distribution lines, differences in the delay of the logic elements, and other factors.

Clock skew can cause timing violations in the circuit and can lead to errors in the circuit's operation. To minimize clock skew, clock distribution networks are designed to ensure that the clock signal arrives at all the logic elements in the circuit at the same time.

Clock gating is a technique that is used to reduce the power consumption of synchronous circuits by disabling the clock signal to selected parts of the circuit when they are not needed. Clock gating is typically implemented using logic gates that are controlled by a separate signal, called the clock enable signal.Figure 1 illustrate the Synchronous Sequential Circuit.

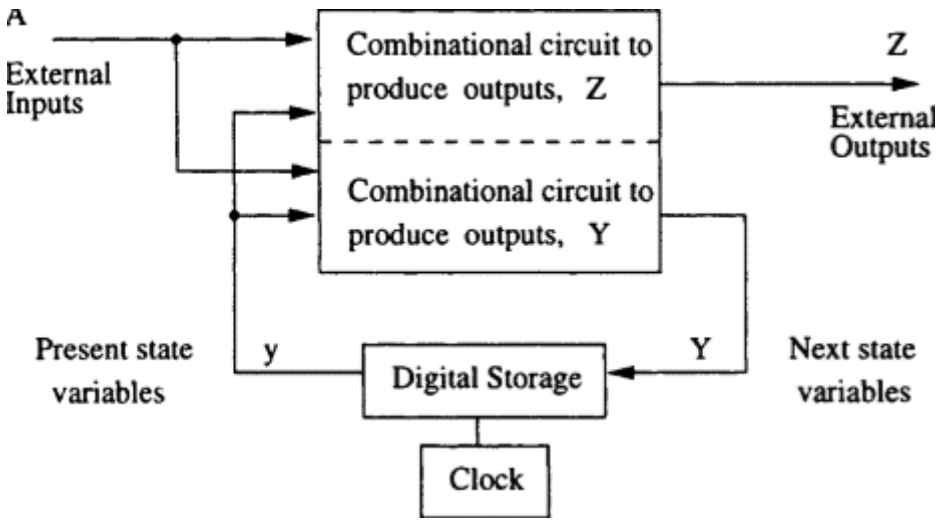


Figure 1: illustrate the Synchronous Sequential Circuit.

Clock gating is typically implemented using logic gates that are controlled by a separate signal, called the clock enable signal. When the clock enable signal is high, the clock signal is allowed to pass through the clock gating logic and reach the logic elements in the circuit. When the clock enable signal is low, the clock signal is blocked, and the logic elements in the circuit are prevented from changing their state [7], [8] .

Clock gating is an important technique for reducing power consumption in synchronous circuits. By disabling the clock signal to selected parts of the circuit when they are not needed, clock gating can reduce the power consumption of the circuit and increase its energy efficiency.

Clock gating is often used in large synchronous circuits, such as microprocessors, where a significant portion of the circuit may be idle for long periods of time. By selectively disabling the clock signal to these idle parts of the circuit, clock gating can reduce the power consumption of the circuit and extend the battery life of portable devices.

However, clock gating can also introduce additional timing constraints on the circuit. When a logic element is clock-gated, it must maintain its state during the period when the clock signal is disabled. This can result in additional delays and may require the use of additional logic elements to ensure that the circuit operates correctly.

Another important aspect of clocking in synchronous circuits is clock recovery. Clock recovery is the process of extracting the clock signal from a data signal that has been transmitted over a noisy channel. In many digital communication systems, the clock signal is

not transmitted along with the data signal, but instead, the receiver must extract the clock signal from the data signal itself.

Clock recovery is typically achieved using a phase-locked loop (PLL) circuit. A PLL is a feedback loop that compares the phase and frequency of an input signal with those of a locally generated reference signal and adjusts the phase and frequency of the reference signal to match those of the input signal.

In clock recovery applications, the input signal is the data signal that contains the embedded clock signal, and the reference signal is a locally generated signal that is initially synchronized with the embedded clock signal. The PLL circuit then adjusts the phase and frequency of the reference signal to match those of the embedded clock signal, thereby recovering the clock signal from the data signal.

Clock recovery is important for ensuring the reliable operation of digital communication systems, especially those that operate at high speeds and over long distances. By recovering the clock signal from the data signal, clock recovery ensures that the receiver can synchronize its internal clock with that of the transmitter, thereby enabling reliable communication.

In conclusion, clocking is a critical aspect of synchronous digital circuits that is essential for ensuring reliable and correct operation. Clocking provides a timing reference that synchronizes the operation of the circuit's internal logic elements, enables the circuit to operate reliably in the presence of noise and other disturbances, and enables the circuit to operate at a higher frequency than would be possible with asynchronous circuits.

Implementing clocking in a synchronous circuit involves several key aspects, including clock generation, clock distribution, clock skew, and clock gating. Clock gating is an important technique for reducing power consumption in synchronous circuits, while clock recovery is critical for extracting the clock signal from a data signal in digital communication systems.

Clocking is a fundamental aspect of synchronous digital circuits. A synchronous circuit is a digital circuit that operates with a clock signal. The clock signal is a periodic signal that is used to synchronize the operation of the circuit elements. Clocking is used to ensure that the various parts of a synchronous circuit operate at the same rate, allowing for predictable behavior and easy timing analysis. In this discussion, we will explore clocking in synchronous circuits in detail.

Synchronous Circuits and Clocking: Synchronous circuits are digital circuits that use a clock signal to coordinate the operation of their elements. The clock signal is used to trigger the flip-flops, which are the basic building blocks of synchronous circuits. The flip-flops store data and the clock signal is used to transfer data from one flip-flop to another.

The clock signal is a periodic signal that has a fixed period and duty cycle. The period of the clock signal determines the frequency of the clock, while the duty cycle determines the amount of time that the clock signal is high versus low. The duty cycle is typically 50% in most synchronous circuits.

Clock Skew: One of the most critical aspects of clocking in synchronous circuits is clock skew. Clock skew refers to the difference in arrival times of the clock signal at different parts

of the circuit. Clock skew can occur due to differences in the routing of the clock signal or differences in the propagation delays of the gates in the circuit.

Clock skew can cause a variety of problems in synchronous circuits. One of the most significant problems is setup and hold violations. Setup and hold violations occur when the input data to a flip-flop changes too close to the rising edge of the clock signal. In this case, the flip-flop may not have enough time to store the new data before the rising edge of the clock signal.

Clock skew can also cause problems with data stability. Data stability refers to the amount of time that the data remains stable after it has been clocked into a flip-flop. Clock skew can cause the data to become unstable and lead to erroneous behavior in the circuit.

Clock Distribution: Clock distribution is another critical aspect of clocking in synchronous circuits. Clock distribution refers to the process of distributing the clock signal to all the parts of the circuit. In large synchronous circuits, the clock signal may need to be distributed over long distances, which can lead to clock skew.

There are several techniques that can be used to minimize clock skew in large synchronous circuits. One common technique is to use clock buffers to amplify the clock signal and reduce the propagation delay. Another technique is to use clock trees to distribute the clock signal in a balanced manner. Clock trees use a series of buffers to ensure that the clock signal arrives at all parts of the circuit at the same time.

Clock Jitter: Clock jitter is another important aspect of clocking in synchronous circuits. Clock jitter refers to the variation in the period of the clock signal. Clock jitter can be caused by a variety of factors, including noise in the power supply or variations in the temperature of the circuit.

Clock jitter can cause several problems in synchronous circuits. One of the most significant problems is increased setup and hold times. As the clock period varies, the setup and hold times for the flip-flops also vary, which can lead to timing violations and incorrect behavior in the circuit.

Clock Skew vs. Clock Jitter: Clock skew and clock jitter are both important aspects of clocking in synchronous circuits, but they are different phenomena. Clock skew refers to the difference in arrival times of the clock signal at different parts of the circuit, while clock jitter refers to the variation in the period of the clock signal.Clock skew and clock jitter can both cause problems in synchronous circuits, but they require different solutions. Clock skew can be minimized by careful clock distribution techniques [9], [10].

Clock Domain Crossing: Clock domain crossing is another important aspect of clocking in synchronous circuits. Clock domain crossing occurs when signals that are synchronized with different clocks need to be transferred between different parts of the circuit.

Clock domain crossing can cause several problems in synchronous circuits. One of the most significant problems is metastability. Metastability occurs when a flip-flop receives an input that changes close to the rising or falling edge of the clock signal. In this case, the flip-flop may not have enough time to settle into a stable state, and the output may oscillate between high and low for an extended period of time.

To avoid metastability, several techniques can be used in synchronous circuits. One common technique is to use two flip-flops in series, with the output of the first flip-flop feeding into the input of the second flip-flop. This technique is called double synchronization, and it ensures that the output of the first flip-flop is stable before it is fed into the second flip-flop.

Another technique that can be used to avoid metastability is to use synchronization logic that guarantees a minimum setup and hold time for the input to the flip-flop. This technique is called synchronizer or 2-flop synchronizer.

Clocking Challenges in High-Speed Designs: High-speed synchronous circuits present several challenges for clocking. In high-speed designs, clock skew and clock jitter become more significant due to the high frequency of the clock signal. Clock skew can cause problems with setup and hold violations, while clock jitter can lead to increased setup and hold times. In addition to clock skew and clock jitter, high-speed designs also face challenges with signal integrity. Signal integrity refers to the ability of a signal to maintain its shape and amplitude as it travels through the circuit. In high-speed designs, signal integrity can be compromised due to the parasitic capacitance and inductance of the interconnects, as well as the noise generated by the switching of the gates.

To address these challenges, several techniques can be used in high-speed designs. One common technique is to use differential signaling, where the signal is transmitted on two wires that are driven in opposite directions. Differential signaling can help to reduce the effects of noise and parasitic capacitance and inductance.

Another technique that can be used in high-speed designs is to use clock gating. Clock gating is a technique where the clock signal is selectively turned off to parts of the circuit that are not currently in use. This technique can help to reduce the power consumption of the circuit and reduce the effects of clock skew and clock jitter.

CONCLUSION

Clocking is an essential aspect of synchronous circuit design. It provides a means of coordinating the operation of various parts of a circuit and ensures that they all operate in synchronization with each other. In clocked circuits, the clock signal acts as a reference for timing the operation of the circuit elements [11]. Proper clocking is critical in synchronous circuit design, and designers must carefully consider the clocking methodology and techniques to ensure the reliable and efficient operation of the circuit.

REFERENCES

[1] "Clocking of Synchronous Circuits," in *Top-Down Digital VLSI Design*, 2015. doi: 10.1016/b978-0-12-800730-3.00007-1.

[2] N. Van Toan, D. M. Tung, and J. G. Lee, "Design of a multi-frequency clocking circuit on an FPGA and analysis of its EMI emission," in *2016 Asia-Pacific International Symposium on Electromagnetic Compatibility, APEMC 2016*, 2016. doi: 10.1109/APEMC.2016.7522808.

[3] K. Gaj, E. G. Friedman, and M. J. Feldman, "Timing of Multi-Gigahertz Rapid Single Flux Quantum Digital Circuits," *J. VLSI Signal Process. Syst. Signal Image. Video Technol.*, 1997, doi: 10.1007/978-1-4684-8440-3_11.

[4] R. N. Tadros and P. A. Beerel, "A theoretical foundation for timing synchronous systems using asynchronous structures," *ACM Trans. Des. Autom. Electron. Syst.*, 2020, doi: 10.1145/3373355.

[5] H. Liu *et al.*, "Synchronous circuit design with beyond-CMOS magnetoelectric spin-orbit devices toward 100-mV logic," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, 2019, doi: 10.1109/JXCDC.2019.2897598.

[6] L. C. Müller, H. R. Gerber, and C. J. Fourie, "Review and comparison of RSFQ asynchronous methodologies," *J. Phys. Conf. Ser.*, 2008, doi: 10.1088/1742-6596/97/1/012109.

[7] A. Ragab, Y. Liu, K. Hu, P. Chiang, and S. Palermo, "Receiver jitter tracking characteristics in high-speed source synchronous links," *J. Electr. Comput. Eng.*, 2011, doi: 10.1155/2011/982314.

[8] M. R. Dagenais and N. C. Rumin, "On the Calculation of Optimal Clocking Parameters in Synchronous Circuits with Level-Sensitive Latches," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 1989, doi: 10.1109/43.21846.

[9] J. P. Fishburn, "Clock Skew Optimization," *IEEE Trans. Comput.*, 1990, doi: 10.1109/12.55696.

[10] A. Kempitiya, D. A. Borca-Tasciuc, and M. M. Hella, "Analysis and optimization of asynchronously controlled electrostatic energy harvesters," *IEEE Trans. Ind. Electron.*, 2012, doi: 10.1109/TIE.2011.2141097.

[11] T. O. Dickson *et al.*, "A 1.8 pJ/bit 16 × 16Gb/s source-synchronous parallel interface in 32nm SOI CMOS with receiver redundancy for link recalibration," *IEEE J. Solid-State Circuits*, 2016, doi: 10.1109/JSSC.2016.2550499.

CHAPTER 16

THE DATA CONSISTENCY PROBLEM OF SCALAR ACQUISITION

Dr. Sovit Kumar, Assistant Professor
Department of Computer Science and Engineering, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id- sovit.soeit@sanskriti.edu.in

ABSTRACT:

The data consistency problem of scalar acquisition is a challenge that arises in data acquisition systems, particularly in systems that use analog-to-digital converters (ADCs) to convert analog signals into digital data. Scalar acquisition refers to the process of acquiring and converting a single value or sample from an analog signal, such as a voltage or current.

KEYWORDS:

ADCs, Digital Data, Data Consistency, Scalar Acquisition, Voltage.

INTRODUCTION

The data consistency problem arises when the acquired data is not consistent with the actual value of the analog signal at the time of acquisition. This can occur due to various factors, including noise, signal interference, and the limitations of the acquisition system itself.

To understand the data consistency problem in scalar acquisition, it is important to first understand the basic principles of analog-to-digital conversion. ADCs are electronic devices that convert analog signals into digital data that can be processed by digital systems, such as computers or microcontrollers. The process of ADC involves several stages, including sampling, quantization, and encoding [1]–[3] .

In the first stage, the analog signal is sampled at a specific rate, known as the sampling rate. The sampling rate determines how frequently the signal is measured, and it must be high enough to accurately capture the analog signal's variations. The sampling rate is typically expressed in samples per second, or hertz (Hz).

After sampling, the analog signal is quantized, which means it is divided into discrete levels or steps. The number of quantization levels determines the resolution of the ADC, which is the smallest change in the analog signal that can be detected by the ADC. The resolution is typically expressed in bits, with higher resolution ADCs having more bits. Finally, the quantized signal is encoded into a digital signal using a binary code, such as the two's complement or binary offset binary (BOB) code. The digital signal can then be processed by digital systems, such as computers or microcontrollers.

The data consistency problem in scalar acquisition arises from several factors, including noise, signal interference, and the limitations of the ADC and acquisition system. Noise refers to any unwanted variation or distortion in the analog signal that is not part of the signal being measured. Noise can be caused by various sources, including electromagnetic interference (EMI), thermal noise, and shot noise [4], [5].

Signal interference refers to any external signal that interferes with the analog signal being measured. Interference can be caused by various sources, including other electrical devices, electromagnetic radiation, and environmental factors such as temperature and humidity.

The limitations of the ADC and acquisition system can also contribute to the data consistency problem. For example, the resolution of the ADC may not be high enough to accurately capture the variations in the analog signal, leading to quantization errors. Similarly, the sampling rate may not be high enough to capture high-frequency variations in the analog signal, leading to aliasing.

To address the data consistency problem in scalar acquisition, several techniques can be used. One approach is to use a higher resolution ADC, which can detect smaller changes in the analog signal and reduce quantization errors. Another approach is to use oversampling, which involves sampling the analog signal at a higher rate than the Nyquist rate, the minimum sampling rate required to avoid aliasing. Oversampling can increase the resolution of the ADC and reduce the effects of noise. Another technique is to use signal conditioning, which involves filtering and amplifying the analog signal before it is sampled by the ADC. Filtering can remove unwanted noise and interference, while amplification can increase the signal-to-noise ratio (SNR) and improve the accuracy of the ADC [6], [7].

Calibration is another important technique for addressing the data consistency problem in scalar acquisition. Calibration involves comparing the acquired data to a known reference value and adjusting the ADC's parameters to improve its accuracy. Calibration can be done at the factory or in the field, and it is important to periodically recalibrate the ADC to ensure its accuracy over.

In addition to these techniques, it is also important to consider the effects of signal aliasing and to ensure that the sampling rate is high enough to accurately capture the analog signal's variations. Aliasing occurs when the sampling rate is not high enough to capture high-frequency components of the analog signal, leading to distortion and errors in the acquired data. To avoid aliasing, it is important to ensure that the sampling rate is at least twice the highest frequency component of the analog signal, as dictated by the Nyquist-Shannon sampling theorem.

Furthermore, it is important to consider the effects of noise and interference and to use appropriate filtering techniques to remove unwanted noise and interference from the analog signal before it is sampled by the ADC. Common filtering techniques include low-pass, high-pass, and band-pass filters, which can be implemented using various types of analog and digital circuits.

Another important consideration is the impact of environmental factors, such as temperature and humidity, on the accuracy and reliability of the ADC and acquisition system. Temperature and humidity can affect the performance of electronic components and circuits, leading to changes in the signal levels and characteristics of the analog signal. To address these effects, it is important to use temperature and humidity sensors and to calibrate the ADC and acquisition system under different environmental conditions.

Finally, it is important to consider the limitations of the ADC and acquisition system and to choose the appropriate components and configurations for the specific application. For example, if high accuracy is required, a higher resolution ADC may be necessary, even if it is

more expensive. Similarly, if low power consumption is required, a lower power ADC may be necessary, even if it has lower accuracy.

DISCUSSION

Electrically erasable memory. EEPROM technology borrows from UV-erasable memories. The difference is that the electrons trapped on the floating gate are removed electrically by having them tunnel through the oxide layer underneath the floating gate without exposure to ultraviolet light, thereby making it possible to manufacture FPL devices that are non-volatile but nevertheless reconfigurable through their package pins [8], [9]. The secret is a quantum-mechanical effect known as Fowler–Nordheim tunnelling that comes into play when a strong vertical field (8–10 MV/cm or so) is applied across the gate oxide.

Early electrically erasable devices were penalised by the fact that an EEPROM cell occupies about twice as much area as its UV-erasable counterpart because each bit cell includes a select transistor connected in series with the storage transistor. The flash memory technology prevalent today manages with a single floating-gate transistor per bit. The fact that erasure must occur in chunks, that is to say many bits at a time, is perfectly adequate in the context of FPL. Data retention times vary between 10 and 40 years. Endurance of flash FPL is typically specified with 100 to 1000 configure–erase cycles, which is much less than for flash memory chips.

Fuse or antifuse. Fuses, which were used in earlier bipolar PROMs and SPLDs, are narrow bridges of conducting material that blow in a controlled fashion when a programming current is forced through. Antifuses, such as those employed in today's FPGAs, are thin dielectrics separating two conducting layers that are made to rupture upon applying a programming voltage, thereby establishing a conductive path of low impedance.

In either case, programming is permanent. Whether this is desirable or not depends on the application. Full factory testing prior to programming of one-time programmable links is impossible for obvious reasons. Special circuitry is incorporated to test the logic devices and routing tracks at the manufacturer before the unprogrammed devices are being shipped. On the other hand, antifuses are only about the size of a contact or via and, therefore, allow for higher densities than reprogrammable links. Antifuse-based FPL is also less sensitive to radiation effects, offers superior protection against unauthorised cloning, and does not need to be configured following power-up Figure 1 illustrate the Axiom dark matter.

Logic programmable AND plane OR plane SPLD flip-flops & feedback inputs outputs programmable feedback logic programmable evolution technological evolution technological flip-flops & feedback AND plane OR plane configurable I/O cell.

The rigid two-level-logic-plus-register architecture in conjunction with the limited numbers of inputs, outputs, product terms, and flip-flops always restricted SPLDs to small applications. More scalable and flexible architectures had thus to be sought, and the spectacular progress of VLSI technology has made their implementation economically feasible from the late 1980s onwards. Two broad classes of hardware organisation prevail today.

Complex programmable logic devices (CPLDs) expand the general idea behind SPLDs by providing many of them on a single chip. Up to hundreds of identical subcircuits, each of

which conforms to a classic SPLD, are combined with a large programmable interconnect matrix or network. A difficulty with this type of organisation is that a partitioning into a bunch of cooperating SPLDs has to be imposed artificially on any given computational task, which benefits neither hardware nor design efficiency.

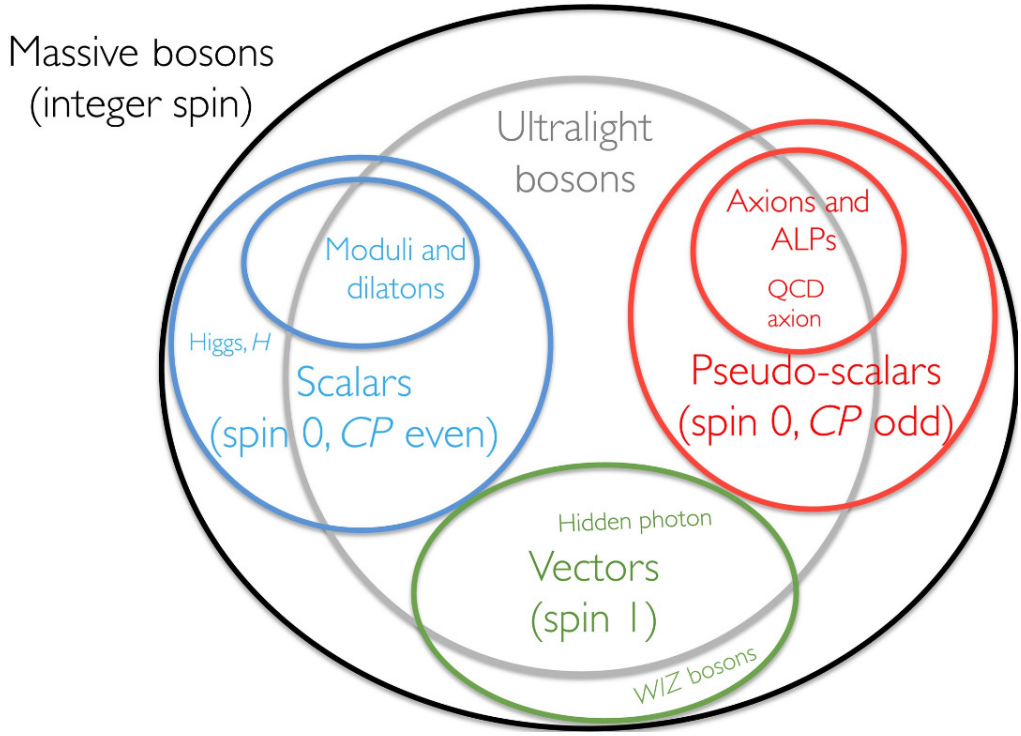


Figure 1: illustrate the Axiom dark matter.

Depending on the manufacturer, products are known as complex programmable logic device (CPLD), programmable large-scale integration (PLSI), erasable programmable logic device (EPLD), and the like in the commercial world. Field-programmable gate arrays (FPGAs) have their overall organisation patterned after that of gate arrays. Many configurable logic cells are arranged in a two-dimensional array with bundles of parallel wires in between. A switchbox is present wherever two wiring channels intersect. Depending on the product, each logic cell can be configured so as to carry out some not-too-complex combinational operation, to store a bit or two, or both.

FPGA architectures are differentiated further depending on the granularity and capabilities of the configurable logic cells employed. One speaks of a fine-grained architecture when those cells are so simple that they are capable of implementing no more than a few logic gates and/or one bistable. In the example depicted for instance, each logic cell can be configured into a latch, or a flip-flop, or into almost any 3-input gate.

As opposed to this, cells that are designed to implement combinational functions of four to six input variables and that are capable of storing two or more bits at a time are referred to as coarse-grained. The logic cell has 16 inputs and 11 outputs, and includes two programmable look-up tables (LUTs), two generic bistables that can be configured either into a latch or a flip-flop, a bunch of configurable multiplexers, a fast carry chain, plus other gates. Of course, the superior functional capabilities offered by a coarse-grained cell are accompanied by a larger area occupation.

The gate-level netlists produced by automatic synthesis map more naturally onto fine-grained architectures. The fact that fine-grained FPGAs and semi-custom ICs provide similar primitives further supports extensive reuse of design flows, HDL code, building blocks, and design Incidentally note that FPL vendors refer to configurable logic cells by proprietary names. “Logic tile” is Actel’s term for their fine-grained cells whereas Xilinx uses the name “configurable logic block” (CLB) for their coarsegrained counterparts. Depending on the product family, one CLB consists of two or three LUTs plus two flip-flops or of several “slices”, each of which includes one LUT and one bistable. “Module” and “eCell”are commercial names used by other vendors.

Another reason that contributed to the popularity of coarse-grained FPGAs is that on-chip RAMs come at little extra cost when that architectural concept is combined with configuration from static memory. In fact, a reprogrammable LUT is nothing else than a tiny storage array. It is thus possible to bind together multiple logic cells in such a way as to make them act collectively like a larger RAM. As opposed to many other types of FPGAs, there is no compelling need to set aside special die areas for embedded SRAMs. In the occurrence ofeach of the two larger LUTs in each logic tile contributes another 16 bits of storage capacity.

In addition to FPL, field-programmable analogue arrays (FPAAs) began to appear on the market in the late 1990s. The next logical step was the extension to mixed-signal applications. Advanced products that combine configurable analogue building blocks with a micro- or digital signal processor and with analog-to-digital and digital-to-analog converters come quite close to the vision of fieldprogrammable systems on a chip. Vendors of field-programmable analogue and mixed-signal arrays include Anadigm, Actel, Cypress, Lattice, and Zetex FAS. Technical details on commercial FPL devices are distributed over thousands of datasheets, [8] [9] help to keep track of products and manufacturers. More condensed background information is available from references such as [10] [11] [12].

Capacity figures of semi-custom ICs and FPL may be confusing. As opposed to full-custom ICs, manufactured gates, usable gates, and actual gates are not the same. Manufactured gates indicate the total number of GEs that are physically present on a silicon die. A substantial fraction thereof is not usable in practise because the combinational functions in a given design do not fit into the available look-up tables exactly, because an FPL device only rarely includes combinational and storage resources with the desired proportions, and because of limited interconnect resources. The percentage of usable gates thus depends on the application. The actual gate count, finally, tells how many GEs are indeed put to service by a given design. The three figures frequently get muddled up, all too often in a deliberate attempt to make one product look better than its competitors in advertisements, product charts, and datasheets. Some FPL vendors prefer to specify the available resources using their own proprietary capacity units rather than in gate equivalents.

Hint: It often pays to conduct benchmarks with a few representative designs before undertaking serious cost calculations and making a misguided choice. This also helps to obtain realistic timing figures that take into account interconnect delays.

The IEEE 1076 standard insists that a resolution function be available for any signal that is being driven from multiple sources but places the details under the designer’s control. Function resolved is fine for fully complementary static CMOS logic. By programming his

own resolution functions, the designer can indicate how to solve driver conflicts in other situations.19 Make sure you understand that there can be no such thing as a resolution function for variables. The same applies to bits, bit vectors, integers, reals, and similar data types. Collapsing of logic values for the purpose of synthesis Not all nine values of the IEEE 1164 logic system make sense from a synthesis point of view. The semantics of 0 and 1 are obvious. A don't care symbol - on the right-hand side of an assignment implies logic value is of no importance, in which case logic synthesis is allowed to select either a 0 or a 1 so as to minimize gate count.

Z also has a well-defined meaning because it calls for a driver with built-in three-state capability as discussed before. Values U, X, and W, in contrast, capture specific situations that occur during simulation, but have no sensible interpretation for synthesis. As far as L and H are concerned, one might imagine an EDA tool that would insert pull-down/-up devices or otherwise play with weak drivers. However, as this would entail static currents unpopular with CMOS circuit designers, L and H are not normally honored by today's synthesis software. Most synthesis tools collapse meaningless (to them) values to more sensible ones, e.g. L to 0, H to 1, and X or W to -. Hint: For the sake of clarity and portability, do not use logic values other than 0, 1, Z, and - in VHDL source code that is intended for synthesis.

Data types for modelling of scalar (single-bit) signals types std ulogic and std logic have been introduced to emulate the electrical behavior of circuit nodes in a more realistic way than IEEE 1076 type bit does. Using them for simulation purposes is not without cost, however. After all, multiple values occupy more storage capacity than a two-valued data type does, and their processing asks for a higher computational effort. The latter is particularly true when a resolution function is, before opting for a type for a VHDL signal or variable, find out what features you need to model, and what effects you can afford to neglect.

To handle open-collector outputs and open-drain when constructing wired-and operations. 198 Architectures of VLSI Circuits data type bit std ulogic std logic defined in VHDL ieee.std logic 1164 for simulation purposes modelling of power-up phase no yes yes modelling of weakly driven nodes no yes yes modelling of multi-driver nodes no yes yes handling of drive conflicts n.a. reported resolved storage requirements minimal substantial computational effort minimal substantial for synthesis purposes three-state drivers no yes yes don't care conditions no yes yes As an example, assume you want to synthesize a circuit node with a single driver. If the code is intended for synthesis exclusively, type bit will do.

If you further want to simulate your code, you will want to learn whether the node has ever been initialized or not. Also, you will want to make sure you get a message from the simulator, should a short circuit between this and some other node inadvertently creep into your design. Type std ulogic would then be the safest choice, although most designers tend to use std logic throughout. Data types for modelling vectored (multi-bit) signals Both std ulogic and std logic represent a single bit, whereas most digital circuits operate on several bits at a time. One data type that gets coded using multiple bits in computers is integer.

There are limitations for describing circuit hardware at lower levels of detail using integers, though. Word width is fixed — to 32 bit in the case of VHDL — and there is no way to access just a portion of a data word. Integers also suffer from a lack of expressiveness for describing electrical phenomena much as type bit does. VHDL further supports the collection of multiple bits into a vector such as in bit vector, std ulogic vector, and std logic vector, all

of which imply a one-dimensional array built from their scalar counterparts. The problem here is the absence of arithmetic operations for those data types. As the existing standards offered no solution, two new packages were developed and accepted. Both packages define two extra data types called signed and unsigned that are overloaded for standard VHDL arithmetic operators as much as possible. Objects of type unsigned are interpreted as unsigned integer binary numbers, and objects of type signed as signed integer binary numbers coded in 2's complement (2'C) format.

No provisions are made to support other number representation schemes such as 1's complement (1'C), sign-and-magnitude (S&M), or any floating-point format. The programmer is free to specify how many bits shall be set aside for coding an unsigned or a signed when declaring a constant, variable or signal. 2 0 A floating-point standard is currently in preparation. Please check section A.1.1 if you are not familiar with binary number representation schemes. The difference between the two packages is that ieee.numeric bit is composed of bit type elements, whereas ieee.numeric std operates on std logic elements. As they otherwise define identical data types and functions, only one of the two packages can be used at a time. Clearly, what has been said about the costs of simulating with multi-valued data types in the context of single-bit nodes also applies to multi-bit nodes. data type(s) integer, bit std logic signed, signed, natural, vector vector unsigned unsigned positive defined in VHDL VHDL ieee.std ieee.nu- ieee.nulogic 1164 meric bit meric std word width 32 bit

CONCLUSION

The data consistency problem of scalar acquisition is a significant challenge in data acquisition systems, particularly in systems that use ADCs to convert analog signals into digital data. The problem arises from various factors, including noise, signal interference, and the limitations of the ADC and acquisition system [10], [11]. To address the problem, various techniques can be used, including oversampling, signal conditioning, calibration, and filtering. It is also important to consider the effects of aliasing, environmental factors, and the limitations of the ADC and acquisition system when designing and implementing data acquisition systems.

REFERENCES

[1] Y. Gowda, E. Newman, L. Rosenstein, and M. Hackl, "Scalar Inferences in the Acquisition of even," *Front. Commun.*, 2020, doi: 10.3389/fcomm.2020.593634.

[2] R. Slabakova, "Scalar implicatures in second language acquisition," *Lingua*, 2010, doi: 10.1016/j.lingua.2009.06.005.

[3] N. Snape and H. Hosoi, "Acquisition of scalar implicatures," *Linguist. Approaches to Biling.*, 2018, doi: 10.1075/lab.18010.sna.

[4] G. Barrio-Arranz, R. De Luis-García, A. Tristán-Vega, M. Martín-Fernández, and S. Aja-Fernández, "Impact of MR acquisition parameters on DTI scalar indexes: A tractography based approach," *PLoS One*, 2015, doi: 10.1371/journal.pone.0137905.

[5] C. Bill, J. Romoli, F. Schwarz, and S. Crain, "Scalar Implicatures Versus Presuppositions: The View from Acquisition," *Topoi*, 2016, doi: 10.1007/s11245-014-9276-1.

[6] B. Geurts, N. Katsos, C. Cummins, J. Moons, and L. Noordman, "Scalar quantifiers: Logic, acquisition, and processing," *Lang. Cogn. Process.*, 2010, doi: 10.1080/01690960902955010.

[7] L. Tieu, C. Bill, J. Romoli, and S. Crain, "Plurality inferences are scalar implicatures: Evidence from acquisition," *Semant. Linguist. Theory*, 2015, doi: 10.3765/salt.v24i0.2421.

[8] S. Aja-Fernández, R. de Luis-García, M. Afzali, M. Molendowska, T. Pieciak, and A. Tristán-Vega, "Micro-structure diffusion scalar measures from reduced MRI acquisitions," *PLoS One*, 2020, doi: 10.1371/journal.pone.0229526.

[9] S. Feng and J. Cho, "Asymmetries between direct and indirect scalar implicatures in second language acquisition," *Front. Psychol.*, 2019, doi: 10.3389/fpsyg.2019.00877.

[10] N. Snape and H. Hosoi, "Acquisition of scalar implicatures evidence from adult Japanese L2 learners of English," *Linguist. Approaches to Biling.*, 2018.

[11] J. Sullivan, K. Davidson, S. Wade, and D. Barner, "Differentiating scalar implicature from exclusion inferences in language acquisition," *J. Child Lang.*, 2019, doi: 10.1017/S0305000919000096.

CHAPTER 17

THE METASTABLE SYNCHRONIZER BEHAVIOR

Mr. Lokesh Lodha, Associate Professor,
Department of Electronics & Communication Engineering, Jaipur National University, Jaipur, India,
Email Id-lokesh.lodha@jnujaipur.ac.in

ABSTRACT:

Met stability is a phenomenon that occurs in digital circuits when a signal's value is uncertain and remains in an intermediate state for a period of time. In synchronizer circuits, met stability can cause data loss or errors and is a significant challenge in digital system design. In this article, we will discuss the behavior of metastable synchronizers, the causes of met stability, and techniques to reduce its effects.

KEYWORDS:

Digital Circuit, Data Loss, Signal, Synchronizer Behavior, Metastable.

INTRODUCTION

A metastable synchronizer is a digital circuit used to transfer data between two clock domains with different clock frequencies. The synchronizer ensures that data transferred from one clock domain to another is sampled at the correct time and is free from any metastable state. When data is transferred from one clock domain to another, it is first captured by a flip-flop or latch that is synchronized to the source clock. The output of this flip-flop is then sampled by a second flip-flop or latch that is synchronized to the destination clock. If the output of the first flip-flop changes close to the edge of the destination clock, it may enter a metastable state, causing errors in the data transfer [1]–[3] .

Causes of Metastability in Synchronizers:

Metastability can occur in synchronizers due to various reasons. One of the main reasons is the clock skew, which is the difference in arrival times of the clock signals at different parts of the circuit. Clock skew can cause the data input to be sampled at different times, leading to metastability. Another cause of metastability is clock jitter, which is the variation in the arrival times of the clock signal edges due to noise and other factors. Clock jitter can cause the input signal to be sampled at different times, leading to metastability.

The size and delay of the flip-flop or latch used in the synchronizer circuit can also affect the occurrence of metastability. If the flip-flop or latch is too small, it may not be able to hold the input signal long enough to stabilize, causing metastability. If the flip-flop or latch is too slow, it may not be able to capture the input signal before the next clock edge, leading to metastability.

The duration of the metastable state depends on the input signal's amplitude, the size and delay of the flip-flop or latch, and the amount of noise in the circuit. The duration of the metastable state can range from nanoseconds to microseconds and can cause data loss or errors [4], [5].

Techniques to Reduce the Effects of Metastability:

Various techniques can be used to reduce the effects of metastability in synchronizer circuits. One such technique is to use a multi-stage synchronizer, which includes multiple flip-flops or latches to increase the delay and reduce the probability of metastability. In a multi-stage synchronizer, the output of one flip-flop is captured by another flip-flop, and so on, until the output is stable. The delay introduced by each flip-flop reduces the probability of metastability.

Another technique is to use a higher-speed flip-flop or latch that can capture the input signal faster, reducing the probability of metastability. A higher-speed flip-flop or latch can also reduce the duration of the metastable state, leading to faster recovery.

A third technique is to use a pulse synchronizer, which generates a pulse signal that is synchronized to the destination clock and is used to sample the input signal. A pulse synchronizer can reduce the probability of metastability by providing a short, precise pulse signal that ensures the input signal is sampled at the correct time.

A fourth technique is to use a self-correcting synchronizer, which includes a feedback loop that corrects any metastable state that occurs. A self-correcting synchronizer includes a latch that captures the output of the first flip-flop and compares it to the output of the second flip-flop.

If the outputs are different, the self-correcting synchronizer generates a correction signal that resets the second flip-flop and resamples the input signal. The feedback loop in the self-correcting synchronizer reduces the duration of the metastable state and ensures that the output is stable.

A fifth technique is to use a high-impedance state in the input of the synchronizer. A high-impedance state can reduce the amount of noise in the circuit, leading to a lower probability of metastability.

It is worth noting that metastability is not always avoidable, and even with the use of these techniques, there is still a small probability of metastability occurring. Therefore, it is important to have a system in place to detect and handle metastability when it occurs.

One way to handle metastability is to use a timeout mechanism that detects when the output of the synchronizer has remained in a metastable state for too long and forces the output to a known state. This approach ensures that the system does not remain stuck in a metastable state indefinitely.

Another way to handle metastability is to use error correction codes (ECC) that can detect and correct errors in the data transfer. ECCs can detect errors caused by metastability and correct them, ensuring the integrity of the data transfer.

Finally, it is important to note that while metastability is a significant challenge in digital system design, it is not always a critical issue. In some cases, the probability of metastability occurring may be low enough that the use of metastable synchronizers is not necessary. It is important to carefully consider the design requirements and determine whether the use of metastable synchronizers is necessary for a given application [6].

The behavior of metastable synchronizers is a complex and challenging aspect of digital system design. The causes of metastability, such as clock skew, clock jitter, and flip-flop or latch size and delay, must be carefully considered, and techniques such as multi-stage synchronizers, higher-speed flip-flops or latches, pulse synchronizers, self-correcting synchronizers, and high-impedance inputs can be used to reduce the effects of metastability. While metastability is not always avoidable, error correction codes and timeout mechanisms can be used to detect and handle metastability when it occurs. Ultimately, designers must carefully consider the design requirements and choose the appropriate technique to ensure reliable data transfer between different clock domains.

DISCUSSION

In VHDL, this applies to data types signed, unsigned, bit vector, std logic vector, and std ulogic vector. Any misinterpretation is likely to cause serious problems for a circuit's simulation and functioning. Hint: Any vector that contains a data item coded in some positional number system should consistently be declared as where 2i is the weight of the binary digit with index i. The MSB will thus have the highest index referring to it and will appear in the customary leftmost position.

Most designers go for resolved data types Simulating with unresolved std ulogic and std ulogic vector types is definitely more conservative than simulating with their resolved counterparts because an error message will tell you, should any of those accidentally get involved in a drive or naming conflict. A historical note is due here most vendors of VHDL software tools had introduced extensions of their own, thereby turning a "no" into a "yes" where indicated. Yet, as all such efforts were made on a proprietary basis, relying on them is detrimental to code p ortability. While the interpretation of arithmetic operators was unlikely to differ, the names and coding schemes of the extra data types were not always the same [7], [8] . Unoffi cial extensions, such as the former logic arith package, must be viewed as obsolete temporary fixes that must no longer be used, now that the numeric packages are available.

For vector ports and signals, the developer should use std logic vector type."22 In practice, the types std logic and std logic vector prevail. An event-based concept of time for governing simulation The need for a mechanism that schedules process execution that VHDL simulation is to yield the same result as if the many processes present in a circuit model were operating simultaneously, although no more than a few processors are normally available for running the simulation code. What is obviously required then is a mechanism that schedules processes for sequential execution and that combines their effects so as to perfectly mimic concurrency.

This mechanism that always sits in the background of VHDL models is the central theme of current events invoke sensitive processes schedule future transactions processes being executed signal concurrent assignment statement (process) variable drivers multiple electrical data types.

Circuit model augmented with an event queue mechanism that governs process activation. Two reasons are given for this surprising advice: Concerns expressed by EDA vendors that they might not be able to optimize simulator performance for both data types, and interoperability of circuit and test bench models from different sources. It is in fact a bizarre

quirk of VHDL that std logic is a subtype which allows for cross assignments without type conversion, whereas std logic vector are two distinct types and, hence, make type conversion compulsory when assigning one type to the other. A perfect comprehension of how a model's concurrent processes are being scheduled during simulation is essential for hardware modelling. Understanding and writing code for synthesis is no exception.

Simulation time versus execution time First of all, we must distinguish between simulation time and execution time. Simulation time is to a VHDL model what physical time is to the hardware described by that model. The simulator software maintains a counter that is set to zero when a new simulation run begins and that registers the progress of simulation time from then on.

This counter can be likened to a stopwatch and any event that occurs during simulation can be thought of as being stamped with the time currently displayed by that clock. Execution time, in contrast, refers to the time a computer takes to execute statements from the VHDL code during simulation. It is of little interest to circuit designers as long as their simulation runs do complete within an acceptable lapse of time. The benefits of a discretized model of time Assume you wanted to model a digital circuit using some conventional programming language. Capturing the logic behavior of its gates and registers poses no major problem, but how about taking into account their respective propagation delays? Figure 1: illustrate the Met stability and Synchronizers.

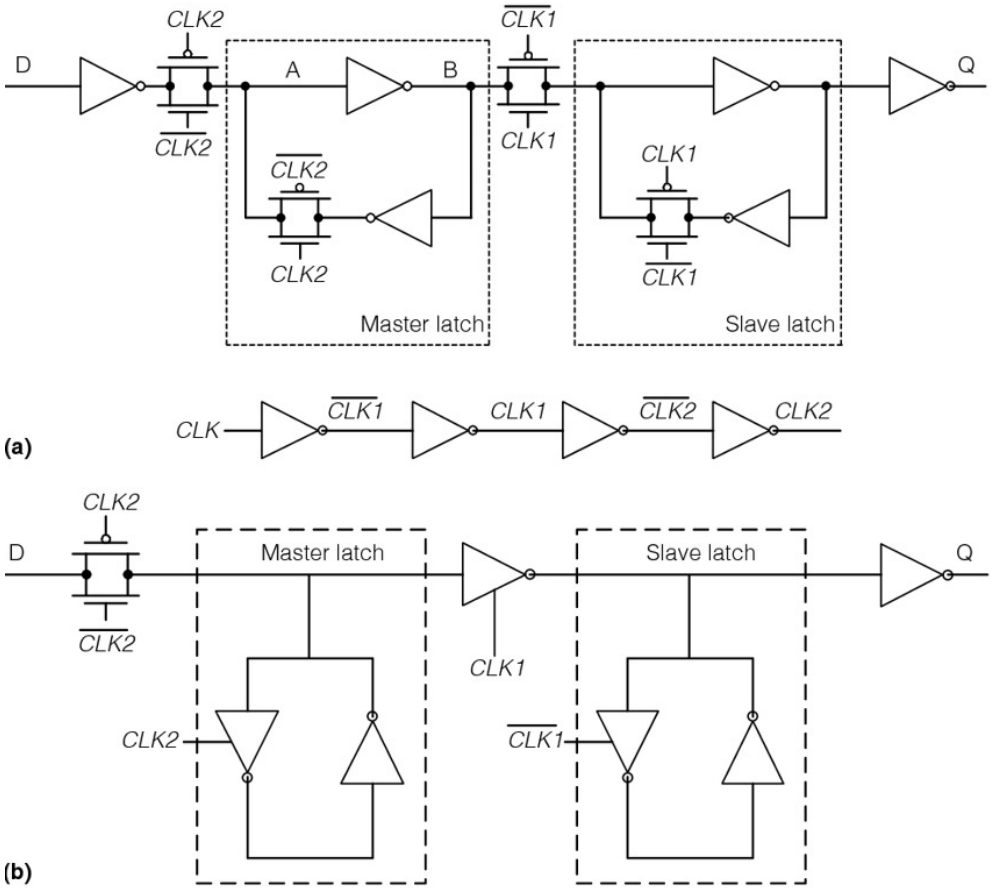


Figure 1: illustrate the Met stability and Synchronizers.

How would you organize a simulation run? You would find that no computations are required unless a node switches. For the sake of efficiency, you would thus decide to consider time as being discrete and would devise some data structure that activates the relevant circuit models when they have to (re)evaluate their inputs. These are precisely the ideas underlying event-driven simulation. Observation 4.9. In VHDL simulation, the continuum of time gets subdivided by events each of which occurs at a precise moment of simulation time. An event is said to happen whenever the value of a signal changes [9].

Event-driven simulation the key element that handles events and that invokes processes is called event queue and can be thought of as a list where entries are arranged according to their time of occurrence. An entry is referred to as a transaction. Event-driven simulation works in cycles where three stages alternate.

Set all signals that are to be updated at the present moment of time to the value associated with the current transaction. Invoke all processes that need to respond to the new situation and have them (re)evaluate their inputs. Every signal assignment supposed to modify a signal's value causes a transaction to be entered into the event queue at that point in the future when the signal is anticipated to take multiple entries may be present for the same moment of time, but the general procedure remains the same.

This stage comes to an end when all processes invoked suspend after having finished to schedule signal updates in response to their current input changes. After completing the third stage, a new simulation cycle is started. Simulation stops when the event queue becomes empty or when simulation time reaches some predefined final value. As nothing happens between transactions, an event-driven simulator essentially skips from one transaction to the next. No computational resources are wasted while models sit idle. Parallel processes and event queue together form a powerful mechanism for modelling the behavior of discrete-time systems.

Past events future transactions present moment of time event queue δ transaction can be inserted into the queue earliest moment of time at which a new (infinitesimal) cycles simulation simulation time b) event queue processes signals current events invoke sensitive processes schedule future transactions processes being executed process process gets invoked by events on signal B and schedules transactions on signals C and D a) sensitive input input signals output signals propagation delays. Event-driven simulation. Interactions between the event queue and a VHDL process (a), actions that repeat during every simulation cycle (b). Transaction versus event it is important to note that any signal update actually occurs in two steps.

Execution of a signal assignment causes a transaction to be entered into the event queue but has no immediate impact. The update is to become effective only later when simulation time has reached the scheduled time for that transaction. By the same token, not every signal assignment that is being carried out necessarily causes a signal to toggle. All too often, a process gets evaluated in response to some event on one of the wake-up signals just to find out that the result is the same as before. Consider a four-input gate or an E-type flip-flop, for instance. Also, the effect of a first transaction may get nullified by a second Incidentally, note that the event queue mechanism is by no means confined to electronic hardware but is also being used for simulating land, air, and data traffic, for evaluating communication protocols, for planning fabrication and logistic processes, and in many other discrete-time applications.

A transaction that does not alter the value of a signal is still a transaction but it does not give rise to an events are observable from the past evolution of a signal's value up to the present moment of simulation time whereas transactions merely reflect future plans that may, or might not, materialize. Delay modelling the lapse of time between an event at the input of a process and the ensuing transaction at its output reflects the delay of the piece of hardware being modelled. In VHDL, delay figures are typically conveyed by the after clause which forms an optional part of the signal assignment statement. The statement below, for instance, models the propagation delay of an adder by scheduling a transaction on its output tpd after an event at either input. Example OUP <= INA + INB after propdelay;

The δ delay For obvious reasons, a process cannot be allowed to schedule signal updates for past or present moments of time. It is, therefore, natural to ask "What is the earliest point in time at which a new transaction can be entered into the queue?" In the occurrence of VHDL, the answer is δ time later, where δ does not advance simulation time but requires going through another simulation cycle. Put differently, δ can be thought of as an infinitesimally small lapse of time greater than zero.

This refinement to the basic event queue mechanism serves to order transactions when the simulation involves models that are supposed to respond with delay zero. Without the δ time step, there would be no way to order zero-delay transactions and simulation could, therefore, not be guaranteed to yield consistent and reproducible results. Although simulation time does not progress in regular intervals, δ may, in some sense, be interpreted as the timewise resolution of the simulator. "How does a simulator handle signal assignments with no after clause?" The answer is that delay is assumed to be zero exactly as if the code read ... after 0 ns ... .

The transaction is then scheduled for the next simulation cycle or, which is the same, one δ delay. An event queue resembles very much an agenda in everyday life. Transactions are analogous to entries there. Signals reflect the evolution of the state of our affairs such as current location and occupation, health condition, social relations, material possessions, and much more. An entry in the agenda stands for some specific intention as anticipated today.

At any time, an event, such as a phone call, m ay force us to alter our plans, i.e. to add, cancel or modify intended activities to adapt to a new situation. Some of our activities remain in vain and do not advance the state of affairs, very much as some of the transactions do not turn into events. Finally, in retrospect, an agenda also serves as a record of past events and bygone states. 2 6 related language constructs that also express time intervals are wait for and reject. A wait for statement causes a process to suspend for the time indicated before being reactivated. The reject clause, a feature added in the 1993 standard update, helps to describe rejection phenomena on narrow pulses in a m ore concise way.

VLSI Circuits simulation time changes scheduled to occur at later moments of time earliest moment of time a signal is allowed to change in response to an assignment at present moment of time kept on record waveforms so far object's value signal object's value variable future waveforms at present moment of time as planned and foreseen no plans known discarded immediately past events future transactions changes that have occurred δ minimum time step present event in response to an assignment at present moment of time only moment of time a variable is allowed to change (infinitesimal).

The past, present, and future of VHDL variables and signals. later. Omitting the after clauses is typical in RTL synthesis models because physically meaningful delay data are unavailable at the time when such models are being established. Much the same applies to behavioral models at the algorithmic level. Hint: When simulating models with no delays other than infinitesimally small δ delays, it becomes difficult to distinguish between cause and effect from graphical simulation output because the pertaining events appear to coincide. It then helps to artificially postpone transactions by a tiny amount of time in otherwise delayless signal assignments.

To allow for quick adjustments, a constant of type time is best declared in a package and referenced throughout a model hierarchy. Note that the largest sum of fake delays must not exceed one clock period, though. Example OUP <= INA + INB after fakedelay; Signal versus variable We now are in a good position to understand those features that separate signals from variables. While the difference in terms of scope exposes those particularities that relate to time. A variable has no time dimension attached, which is to say that it merely holds a present value. Neither transactions nor events are involved [10].

The effect of a signal assignment is not felt before the delay specified in the after clause has expired. The minimum delay, and default value in the absence of an after clause, is δ. VHDL signals convey time-varying information between processes via the event queue. They are instrumental in process invocation, which is directed by the same mechanism. Variables, in contrast, are confined to within a process statement or subprogram and do not interact with the event queue in any way. Be sure to understand the observation below as ignoring it gives rise to frequent misconceptions.

A signal assignment (<=) does not become effective before the delay specified in the after clause has expired. In the absence of an explicit indication, there is a delay of one simulation cycle, so the effect can never be felt in the next statement. This sharply contrasts with a variable assignment (:=), the effect of which is felt immediately, that is, in the next statement exactly as in any programming language. Concurrent processes (order of execution)

A process is either active or suspended at any time. Simulation time is stopped while the code of the processes currently active is being carried out, which implies that (a) all active processes are executed concurrently with respect to simulation time, and (b) all sequential statements inside a process statement are executed in zero simulation time. The order of process invocation with respect to execution time is undetermined.

As opposed to conventional programming languages where the thread of execution is strictly defined by the order of statements in the source code, there is no fixed ordering for carrying out processes (including concurrent signal assignments and assertion statements) in VHDL. When to invoke a process gets determined solely by events on the signals that run back and forth between processes. Sensitivity list each process has its own set of signals that cause it to get (re)activated whenever an event occurs on one or more of them. The process is said to be sensitive to those signals.

The entirety of such signals are aptly qualified as its wake-up or trigger signals, although this is not official VHDL terminology. What are the wake-up signals of a given process? The answer depends on the type of process. Architectures of VLSI Circuits Concurrent, selected, and conditional signal assignments (activation) Specifying wake-up signals is neither

necessary nor legal as the process is simply sensitive to any signal that appears anywhere on the right-hand side of the assignment operator.

Another option for indicating where execution of a process statement is to suspend and when it is to resume, is to include a wait statement. Note that the two forms are mutually exclusive. That is, no process statement is allowed to include both a sensitivity list and waits. As the name suggests, process execution suspends when it reaches a wait statement. It resumes with the subsequent instruction as soon as a condition specified is met and continues until the next wait is encountered, and so on.

The wait statement comes in four flavors that differ in the nature of the condition for process reactivation: statement wake-up condition wait on ... an event (signal change) on any of the signals listed here wait until ... idem plus the logic conditions specified here wait for ... a predetermined lapse of time as specified here wait none, sleep forever as no wake-up condition is given process suspends here until reactivated -- by an event on any of these signals end process memless2; -- execution continues with first statement Note that process execution does not terminate with the end process statement but resumes at the top of the process body. In a process statement with a single wait statement, execution thus necessarily makes a full turn through the process code every time the process gets (re)activated.

As opposed to this, only a fragment of the code gets executed in a process with multiple waits, which also implies that there can be no equivalent process with a sensitivity list in this case.A process statement implies memory whenever one or more of the conditions below apply. Conversely, memoryless behavior is being modeled iff none of them holds.

VHDL knows of no specific language construct that could distinguish a sequential model from a combinational one. Similarly, there are no reserved words to indicate whether a piece of code is intended to model a synchronous or an asynchronous circuit, or whether a finite state machine is of Mealy, Moore or Medvedev type. The reason why the wait is placed at the end — rather than at the beginning — in the memless2 code is that all processes get activated once until they suspend as part of the initialization phase at simulation time zero.

CONCLUSION

Metastable synchronizers are a significant challenge in digital system design, as they can cause data loss or errors. Metastability can occur due to clock skew, clock jitter, flip-flop or latch size and delay, and noise in the circuit [11] . Various techniques can be used to reduce the effects of metastability, including multi-stage synchronizers, higher-speed flip-flops or latches, pulse synchronizers, self-correcting synchronizers, and high-impedance inputs. Designers must carefully consider the design requirements and choose the appropriate technique to reduce the effects of metastability and ensure reliable data transfer between different clock domains.

REFERENCES

[1] D. F. Wann, “Theoretical and Experimental Behavior of Synchronizers Operating in the Metastable Region,” *IEEE Trans. Comput.*, 1975, doi: 10.1109/T-C.1975.224273.

[2] J. U. Horstmann, H. W. Eichel, and R. L. Coates, “Metastability Behavior of CMOS ASIC FlipFlops in Theory and Test,” *IEEE J. Solid-State Circuits*, 1989, doi: 10.1109/4.16314.

[3] H. J. M. Veendrick, "The Behavior of Flip-Flops Used as Synchronizers and Prediction of Their Failure Rate," *IEEE J. Solid-State Circuits*, 1980, doi: 10.1109/JSSC.1980.1051359.

[4] T. J. Chaney and C. E. Molnar, "Anomalous Behavior of Synchronizer and Arbiter Circuits," *IEEE Trans. Comput.*, 1973, doi: 10.1109/T-C.1973.223730.

[5] R. Männer, "Metastable states in asynchronous digital systems: Avoidable or unavoidable?," *Microelectron. Reliab.*, 1988, doi: 10.1016/0026-2714(88)90363-0.

[6] L. Kleeman, "The Jitter Model for Metastability and Its Application to Redundant Synchronizers," *IEEE Trans. Comput.*, 1990, doi: 10.1109/12.55694.

[7] L. Kleeman and A. Cantoni, "METASTABLE BEHAVIOR IN DIGITAL SYSTEMS.," *IEEE Des. Test Comput.*, 1987, doi: 10.1109/MDT.1987.295189.

[8] A. El-Amawy, "Comments on 'Can Redundancy and Masking Improve the Performance of Synchronizers?,'" *IEEE Transactions on Computers*. 1989. doi: 10.1109/12.24278.

[9] "Acquisition of Asynchronous Data," in *Top-Down Digital VLSI Design*, 2015. doi: 10.1016/b978-0-12-800730-3.00008-3.

[10] M. PĚCHOUČEK, "Anomalous Response Times of Input Synchronizers," *IEEE Trans. Comput.*, 1976, doi: 10.1109/TC.1976.5009227.

[11] W. Fleischhammer and O. Dortok, "The Anomalous Behavior of Flip-Flops in Synchronizer Circuits," *IEEE Trans. Comput.*, 1979, doi: 10.1109/TC.1979.1675337.

CHAPTER 18

A CONCEPTUAL STUDY ON ELECTRICAL CMOS CONTRAPTIONS

Mr. Lokesh Lodha, Associate Professor,
Department of Electronics & Communication Engineering, Jaipur National University, Jaipur, India,
Email Id-lokesh.lodha@jnujaipur.ac.in

ABSTRACT:

Electrical CMOS (complementary metal-oxide-semiconductor) contraptions are electronic devices that use CMOS technology to perform various functions in digital circuits. CMOS technology is a type of semiconductor fabrication process that uses both n-type and p-type transistors to create logic gates and other digital components.

KEYWORDS:

Contraptions, CMOS, Digital Components, Electrical, Semiconductor.

INTRODUCTION

The use of complementary transistors in a CMOS circuit means that the power dissipated by the circuit is minimal, making it well-suited for use in battery-powered devices and other low-power applications.

Some of the most common electrical CMOS contraptions include logic gates, multiplexers, flip-flops, and shift registers [1], [2].

Logic gates are fundamental building blocks of digital circuits and are used to perform Boolean logic operations such as AND, OR, and NOT. CMOS logic gates typically use two complementary transistors to create a high and low voltage state, depending on the input values.

Multiplexers, also known as data selectors, are used to select one of several input signals and output it to a single output line. CMOS multiplexers typically use a combination of logic gates and transmission gates to perform this function.

Flip-flops are bistable circuits that can store a single bit of information. CMOS flip-flops typically use four transistors to store a single bit of information and can be used to implement various types of digital circuits such as counters and registers.

Shift registers are digital circuits that can shift a series of bits through a chain of flip-flops. CMOS shift registers typically use a combination of flip-flops and multiplexers to perform this function and are used in various digital applications such as serial communication and data storage.

In addition to these basic electrical CMOS contraptions, there are many other more complex devices that can be created using CMOS technology, such as microprocessors and memory circuits.

It is worth noting that while CMOS technology offers many advantages, it also has some limitations. One limitation is the relatively slow speed of CMOS circuits compared to other types of digital circuits. Another limitation is the susceptibility of CMOS circuits to noise and interference, which can result in errors in digital circuits [3].

In summary, electrical CMOS contraptions are essential building blocks of digital circuits that use CMOS technology to perform various functions such as logic gates, multiplexers, flip-flops, and shift registers. CMOS technology offers low power consumption and is well-suited for use in battery-powered devices and other low-power applications. While CMOS technology has some limitations, it remains one of the most widely used fabrication processes in digital circuit design.

In addition to the basic electrical CMOS contraptions mentioned earlier, there are also other types of electrical CMOS contraptions that can be used in digital circuits. Some examples include:

1. Adders and Subtractors: These are circuits used for performing arithmetic operations such as addition and subtraction. CMOS adders and subtractors typically use a combination of logic gates and flip-flops to perform these operations.
2. Comparators: These are circuits used to compare two input values and output a high or low signal depending on which input is greater. CMOS comparators typically use a combination of logic gates and transistors to perform this function.
3. Multiplication and Division Circuits: These are circuits used for performing multiplication and division operations on digital data. CMOS multiplication and division circuits typically use a combination of logic gates, shift registers, and other digital components to perform these operations.
4. Decoders and Encoders: These are circuits used for decoding and encoding digital signals. CMOS decoders and encoders typically use a combination of logic gates and multiplexers to perform this function.
5. Oscillators: These are circuits used to generate a periodic waveform. CMOS oscillators typically use a combination of resistors, capacitors, and transistors to create the oscillation.
6. Analog-to-Digital and Digital-to-Analog Converters: These are circuits used to convert analog signals to digital signals and vice versa. CMOS analog-to-digital and digital-to-analog converters typically use a combination of operational amplifiers, resistors, capacitors, and other analog components to perform this function.

DISCUSSION

In the world of electronics, the Complementary Metal-Oxide-Semiconductor (CMOS) technology is one of the most widely used and popular technologies for designing digital integrated circuits. CMOS technology has been an integral part of the electronic industry for over four decades now and has undergone significant improvements and advancements over the years. This article aims to provide a conceptual study of electrical CMOS contraptions [4]–[6] .

Complementary Metal-Oxide-Semiconductor (CMOS) is a digital technology used for designing digital integrated circuits. The technology uses two types of complementary transistors, namely NMOS (n-channel Metal-Oxide-Semiconductor) and PMOS (p-channel Metal-Oxide-Semiconductor), to create logic gates and other digital circuits. The main advantage of CMOS technology over other digital technologies is that it consumes very low power and produces less heat. This makes it an ideal technology for battery-operated devices and other low-power applications.

CMOS Transistors:

The basic building block of CMOS technology is the MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor). A MOSFET is a type of transistor that uses a voltage signal to control the flow of current through a channel. In CMOS technology, there are two types of MOSFETs, namely the NMOS and PMOS transistors.

NMOS Transistors:

The NMOS transistor is a type of MOSFET that uses a negative voltage signal to control the flow of current through a channel. The NMOS transistor consists of a source, drain, and gate. When a positive voltage signal is applied to the gate of the transistor, it creates an electric field that attracts electrons from the source to the drain, allowing current to flow through the channel. When a negative voltage signal is applied to the gate, the electric field is reversed, and the channel is closed, preventing current from flowing.

PMOS Transistors:

The PMOS transistor is a type of MOSFET that uses a positive voltage signal to control the flow of current through a channel. The PMOS transistor consists of a source, drain, and gate. When a negative voltage signal is applied to the gate of the transistor, it creates an electric field that attracts holes from the source to the drain, allowing current to flow through the channel. When a positive voltage signal is applied to the gate, the electric field is reversed, and the channel is closed, preventing current from flowing.

CMOS Logic Gates:

CMOS technology uses NMOS and PMOS transistors to create logic gates and other digital circuits. A logic gate is a device that performs a Boolean operation on one or more input signals and produces an output signal. There are several types of logic gates, including AND, OR, NOT, NAND, NOR, and XOR gates.

In CMOS technology, a logic gate is created by combining an NMOS transistor and a PMOS transistor in a specific configuration. The NMOS transistor is used as a switch that allows current to flow when the input signal is high, while the PMOS transistor is used as a switch that allows current to flow when the input signal is low. By combining these two transistors in different configurations, various logic gates can be created.

CMOS Integrated Circuits:

CMOS technology is widely used for designing digital integrated circuits, such as microprocessors, memory chips, and other digital circuits. An integrated circuit is a device that contains multiple electronic components, such as transistors, resistors, and capacitors, on a single chip of semiconductor material. CMOS technology is ideal for designing integrated

circuits because it consumes very low power and produces lessheat compared to other digital technologies. This allows for the integration of a large number of electronic components on a single chip without the risk of overheating or power consumption issues.

CMOS integrated circuits are designed using a process called photolithography. This process involves the use of light-sensitive materials, called photoresists, to create patterns on a silicon wafer. The patterns are then etched onto the silicon wafer using a series of chemical processes, creating the electronic components of the integrated circuit.

One of the key advantages of CMOS technology is its scalability. The technology can be scaled up or down to create integrated circuits of different sizes and complexities. This scalability allows for the creation of high-performance microprocessors, memory chips, and other digital circuits that can be used in a wide range of electronic devices, from smartphones and laptops to industrial control systems and automotive electronics.

Applications of CMOS Technology:

CMOS technology has a wide range of applications in the field of electronics. Some of the most common applications include:

1. Microprocessors: CMOS technology is widely used for designing high-performance microprocessors used in desktops, laptops, servers, and other computing devices.
2. Memory chips: CMOS technology is also used for designing memory chips, such as RAM (Random Access Memory) and ROM (Read-Only Memory) chips used in computers and other electronic devices.
3. Digital signal processors: CMOS technology is used for designing digital signal processors used in audio and video processing, speech recognition, and other signal processing applications.
4. Imaging sensors: CMOS technology is used for designing imaging sensors used in digital cameras, smartphones, and other imaging devices.
5. Power management: CMOS technology is also used for designing power management circuits used in battery-operated devices, such as smartphones, laptops, and portable gaming consoles.

Complementary Metal-Oxide-Semiconductor (CMOS) is a technology used to construct integrated circuits. The basic building block of a CMOS circuit is the CMOS inverter. It consists of two transistors - one p-channel and one n-channel - that are connected in a particular configuration. The output of the inverter is the voltage at the junction of the two transistors, which can be either high or low.
The p-channel transistor is controlled by a voltage applied to its gate, and the n-channel transistor is controlled by a voltage applied to its gate. When the voltage on the p-channel transistor's gate is high, it is turned off, and when the voltage on the n-channel transistor's gate is low, it is turned off. The output of the inverter is the voltage at the junction of the two transistors, which is high when the p-channel transistor is turned off and the n-channel transistor is turned on, and low when the n-channel transistor is turned off and the p-channel transistor is turned on.

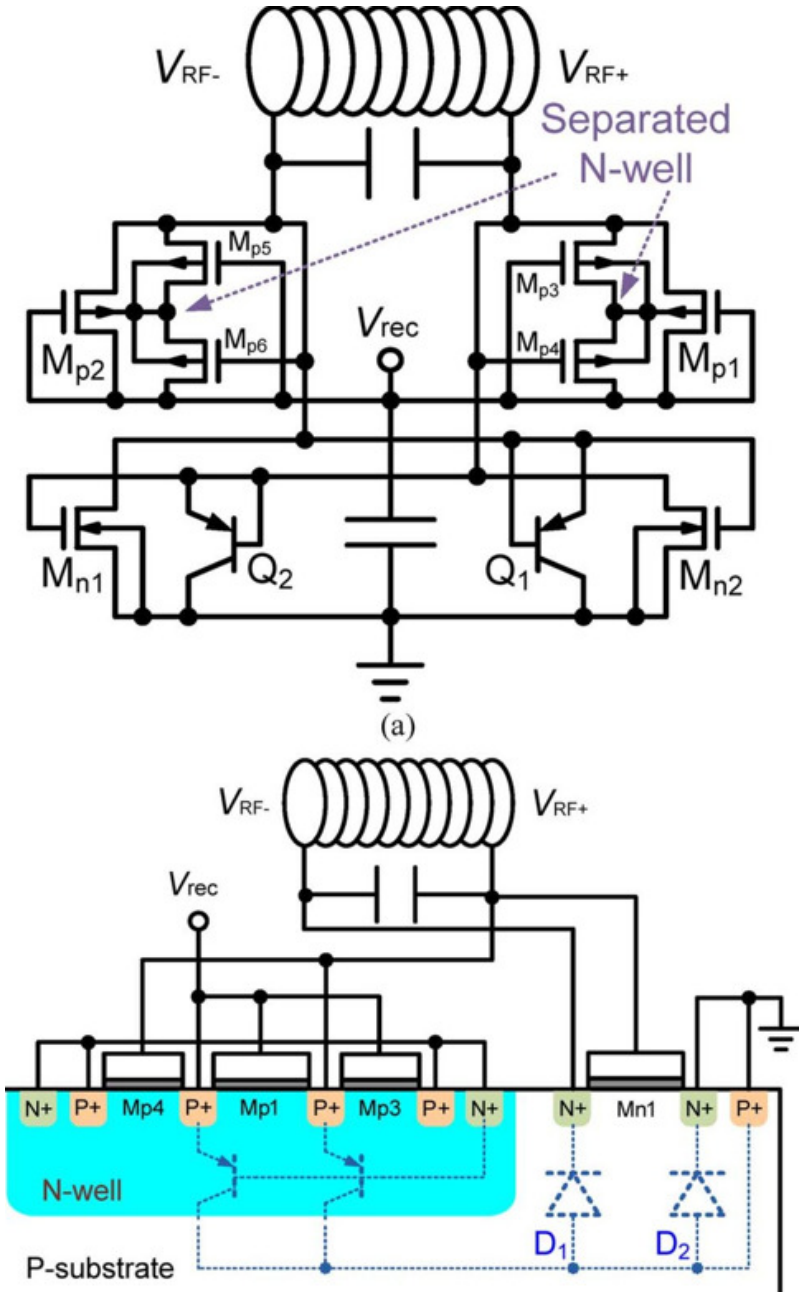


Figure 1 illustrate the Conventional CMOS full-wave bridge rectifier.

CMOS Components: The basic components of a CMOS circuit are the inverter, the n-channel MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor), and the p-channel MOSFET. MOSFET is a type of transistor that is widely used in digital circuits. In CMOS circuits, the MOSFETs are used as switches to control the flow of current.

In a MOSFET, a voltage is applied to the gate, which creates an electric field that controls the flow of current through the transistor. The n-channel MOSFET has a source, a drain, and a gate. When the gate voltage is higher than the source voltage, the transistor is turned on, and current can flow from the source to the drain. When the gate voltage is lower than the source voltage, the transistor is turned off, and no current flows.

Similarly, the p-channel MOSFET has a source, a drain, and a gate. When the gate voltage is lower than the source voltage, the transistor is turned on, and current can flow from the drain

to the source. When the gate voltage is higher than the source voltage, the transistor is turned off, and no current flows.

CMOS Contraptions:

There are various CMOS contraptions used in electronic circuits. Some of the most common contraptions are described below:

1. CMOS Inverter: As mentioned earlier, the CMOS inverter is the basic building block of CMOS circuits. It consists of a p-channel MOSFET and an n-channel MOSFET connected in series. When the input voltage is high, the p-channel MOSFET is turned off, and the n-channel MOSFET is turned on, which makes the output voltage low. When the input voltage is low, the p-channel MOSFET is turned on, and the n-channel MOSFET is turned off, which makes the output voltage high.
2. CMOS NAND Gate: The CMOS NAND gate is a logic gate that produces a low output when both inputs are high, and a high output in all other cases. It consists of two or more CMOS inverters connected in series.
3. CMOS NOR Gate: The CMOS NOR gate is a logic gate that produces a high output whenboth inputs are low, and a low output in all other cases. It also consists of two or more CMOS inverters connected in series.
4. CMOS XOR Gate: The CMOS XOR gate is a logic gate that produces a high output when the number of high inputs is odd, and a low output when the number of high inputs is even. It consists of several CMOS gates and transistors.
5. CMOS Flip-Flop: A flip-flop is a circuit that is used to store a single bit of data. The CMOS flip-flop consists of two cross-coupled inverters and two transmission gates. It has two stable states, and the output state depends on the input state and the previous state.

Applications of CMOS Contraptions: CMOS technology is widely used in digital circuits, such as microprocessors, memory chips, and digital signal processors. CMOS contraptions are used in various applications, some of which are described below:

1. Arithmetic and Logic Operations: CMOS contraptions, such as NAND gates, NOR gates, and XOR gates, are used in arithmetic and logic operations in digital circuits.
2. Memory: CMOS technology is used in the construction of memory chips, such as RAM (Random Access Memory) and ROM (Read-Only Memory).
3. Microprocessors: CMOS technology is used in the construction of microprocessors, which are used in computers and other electronic devices.
4. Digital Signal Processing: CMOS technology is used in the construction of digital signal processors, which are used in various applications, such as audio and video processing [7].

The complementary metal-oxide-semiconductor (CMOS) inverter is a fundamental building block of digital circuits. It is used to invert the input signal and produce the opposite output signal. The CMOS inverter is an essential component of various digital circuits, such as logic

gates, flip-flops, and microprocessors. In this paper, we will discuss the CMOS inverter in detail, including its construction, operation, and applications.

When the input signal is low (0V), the PMOS transistor is in the on state (conducting) and the NMOS transistor is in the off state (non-conducting). This connects the output signal to the power supply (VDD), which produces a high output signal (VDD). When the input signal is high (VDD), the PMOS transistor is in the off state (non-conducting) and the NMOS transistor is in the on state (conducting). This connects the output signal to the ground, which produces a low output signal (0V).

Operation of CMOS Inverter: The CMOS inverter operates in two modes: the cut-off mode and the saturation mode. The cut-off mode occurs when the input voltage is less than the threshold voltage of the NMOS transistor. In this mode, the NMOS transistor is non-conducting and the PMOS transistor is conducting. The output voltage is equal to the supply voltage (VDD).

The saturation mode occurs when the input voltage is greater than the threshold voltage of the NMOS transistor. In this mode, the NMOS transistor is conducting and the PMOS transistor is non-conducting. The output voltage is equal to the ground voltage (0V).

Applications of CMOS Inverter: The CMOS inverter is a fundamental building block of digital circuits and is used in various applications, some of which are described below:

1. Logic Gates: The CMOS inverter is used in the construction of various logic gates, such as NAND gates, NOR gates, and XOR gates.
2. Flip-Flops: The CMOS inverter is used in the construction of flip-flops, which are used to store a single bit of data.
3. Microprocessors: The CMOS inverter is used in the construction of microprocessors, which are used in computers and other electronic devices.
4. Voltage Level Translation: The CMOS inverter is used to
5. Clock Generation: The CMOS inverter is also used in clock generation circuits, where it is used to generate the complementary clock signals needed in digital circuits.
6. Amplification: The CMOS inverter can be used as a voltage amplifier by connecting a resistor between the output and the input of the inverter. This creates a feedback loop, which amplifies the input signal. The gain of the amplifier can be controlled by changing the value of the resistor.
7. Oscillators: The CMOS inverter can be used to construct oscillators, which are circuits that produce a periodic waveform. The oscillator can be constructed by connecting the output of the inverter to the input through a feedback loop. The frequency of the oscillation can be controlled by changing the values of the resistors and capacitors in the feedback loop.

Advantages of CMOS Inverter: The CMOS inverter has several advantages, including:

1. Low Power Consumption: The CMOS inverter consumes very little power, making it ideal for use in battery-powered devices.

2. High Noise Immunity: The CMOS inverter has high noise immunity, which means that it is less susceptible to noise than other types of logic gates.
3. High Input Impedance: The CMOS inverter has a high input impedance, which means that it does not load the input signal.
4. High Output Drive Capability: The CMOS inverter has a high output drive capability, which means that it can drive a large number of loads without degrading the output signal.
5. Small Size: The CMOS inverter is very small in size, making it ideal for use in integrated circuits.

Disadvantages of CMOS Inverter: The CMOS inverter has some disadvantages, including:

1. Limited Speed: The CMOS inverter has a limited speed due to the inherent delay caused by the capacitance of the transistors and the parasitic capacitance of the interconnects.
2. Sensitivity to Electrostatic Discharge (ESD): The CMOS inverter is sensitive to electrostatic discharge (ESD), which can damage the transistors and cause a malfunction [8], [9] .

The field of electronics has undergone rapid development in recent decades, leading to the creation of a wide range of electronic devices that have become an integral part of our daily lives. These devices rely on digital integrated circuits, which are designed using complementary metal-oxide-semiconductor (CMOS) technology. CMOS technology is a type of digital technology that uses two types of complementary transistors to create logic gates and other digital circuits.

This article presents a conceptual study on electrical CMOS contraptions, covering the working principle of CMOS technology, its advantages over other digital technologies, the process of designing integrated circuits using CMOS technology, and some of the common applications of CMOS technology in the field of electronics.

CMOS technology uses two types of complementary transistors, namely n-type metal-oxide-semiconductor (NMOS) transistors and p-type metal-oxide-semiconductor (PMOS) transistors, to create logic gates and other digital circuits. The complementary nature of these transistors allows for low power consumption and heat production, making CMOS technology an ideal choice for designing digital integrated circuits.

NMOS transistors are made up of a source, a drain, and a gate. The source and drain are doped with n-type impurities, while the gate is made up of a thin layer of metal oxide, such as silicon dioxide. When a positive voltage is applied to the gate, it creates an electric field that attracts electrons from the source to the drain, allowing current to flow through the transistor.

PMOS transistors, on the other hand, are made up of a source, a drain, and a gate. The source and drain are doped with p-type impurities, while the gate is made up of a thin layer of metal oxide, such as silicon dioxide. When a negative voltage is applied to the gate, it creates an electric field that attracts holes from the source to the drain, allowing current to flow through the transistor.

The complementary nature of NMOS and PMOS transistors allows for the creation of logic gates, such as AND gates, OR gates, and NOT gates. For example, an AND gate can be created by connecting two NMOS transistors in series between the output and the power supply, and connecting a PMOS transistor between the output and the ground. When both inputs are high, both NMOS transistors conduct, allowing current to flow from the power supply to the output. At the same time, the PMOS transistor is turned off, preventing current from flowing to the ground. When either input is low, one of the NMOS transistors is turned off, preventing current from flowing to the output. At the same time, the PMOS transistor is turned on, allowing current to flow from the output to the ground [10].

CONCLUSION

The use of electrical CMOS contraptions is crucial in digital circuit design, as they provide a reliable and efficient way to perform various digital functions. The versatility and low power consumption of CMOS technology make it an excellent choice for many digital applications, and ongoing research is aimed at overcoming the limitations of CMOS technology to further improve its performance in digital circuit design.

REFERENCES

[1] M. Asgari and K. S. Lee, "Fully-Integrated CMOS Electrical Conductivity Sensor for Wet Media," *IEEE Sens. J.*, 2019, doi: 10.1109/JSEN.2019.2911206.

[2] J. Abbott *et al.*, "CMOS nanoelectrode array for all-electrical intracellular electrophysiological imaging," *Nat. Nanotechnol.*, 2017, doi: 10.1038/nnano.2017.3.

[3] S. Ronchi *et al.*, "Single-Cell Electrical Stimulation Using CMOS-Based High-Density Microelectrode Arrays," *Front. Neurosci.*, 2019, doi: 10.3389/fnins.2019.00208.

[4] H. Mutoh, "3-D optical and electrical simulation for CMOS image sensors," *IEEE Trans. Electron Devices*, 2003, doi: 10.1109/TED.2002.806965.

[5] Y. Li, Z. Wang, R. Midya, Q. Xia, and J. Joshua Yang, "Review of memristor devices in neuromorphic computing: Materials sciences and device challenges," *Journal of Physics D: Applied Physics*. 2018. doi: 10.1088/1361-6463/aade3f.

[6] H. Park *et al.*, "Investigation of electrical characteristics of flexible CMOS devices fabricated with thickness-controlled spalling process," *Solid. State. Electron.*, 2020, doi: 10.1016/j.sse.2020.107901.

[7] L. Lai, B. Fu, and R. Zhang, "Effect of broadband sources on electrical crosstalk of CMOS array," *Hongwai yu Jiguang Gongcheng/Infrared Laser Eng.*, 2017, doi: 10.3788/irla20174601.120005.

[8] G. Ghibaudo, "Electrical characterization of advanced FDSOI CMOS devices," *Compos. nanoélectroniques*, 2019, doi: 10.21494/iste.op.2018.0304.

[9] J. Yoo, Y. Kim, D. Lim, and S. Kim, "Electrical characteristics of silicon nanowire CMOS inverters under illumination," *Opt. Express*, 2018, doi: 10.1364/oe.26.003527.

[10] Y. Chen *et al.*, "CMOS high density electrical impedance biosensor array for tumor cell detection," *Sensors Actuators, B Chem.*, 2012, doi: 10.1016/j.snb.2012.07.024.

CHAPTER 19

UNDERSTANDING AND MITIGATING THE EFFECTS OF GROUND BOUNCE AND SUPPLY DROOP IN DIGITAL CIRCUITS

Prof. Sudhir Kumar Sharma, Associate Professor,
Department of Electronics & Communication Engineering, Jaipur National University, Jaipur, India,
Email Id-hodece_sadtm@jnujaipur.ac.in

ABSTRACT:

Ground bounce and supply droop are two important issues in VLSI (Very Large Scale Integration) design that can affect the performance and reliability of integrated circuits. In this article, we will explain what ground bounce and supply droop are, what causes them, and how they can be mitigated.

KEYWORDS:

Bounce, Circuit, Supply droop, mitigated, VLSI.

INTRODUCTION

Ground Bounce: Ground bounce is a phenomenon that occurs when a large number of circuit elements switch simultaneously, causing a transient voltage drop on the ground plane. The ground bounce can affect the operation of other circuits that share the same ground reference, leading to timing errors, signal integrity issues, and even functional failures [1]–[3].

The ground bounce is caused by the inductance of the ground plane and the package leads, which can cause a voltage drop when a large current flows through them. The voltage drop can be modeled as an inductive voltage drop, which is proportional to the rate of change of the current.

The ground bounce can be mitigated by reducing the switching current and the inductance of the ground plane and the package leads. This can be achieved by using low-inductance power and ground planes, decoupling capacitors, and careful layout design that minimizes the loop area of the current path.

One common technique to mitigate ground bounce is to use a ground grid structure, which provides a low-inductance ground path for the switching currents. The ground grid structure consists of a series of vias that connect the power and ground planes together, forming a grid of low-impedance paths for the current flow.

Another technique to mitigate ground bounce is to use a low-dropout regulator (LDO) to provide a stable voltage reference for the sensitive circuits. The LDO can provide a low-impedance path for the ground current, reducing the voltage drop caused by the inductance of the ground plane and the package leads [4], [5].

Supply Droop: Supply droop is a phenomenon that occurs when a large current flows through the power supply, causing a voltage drop on the power rail. The supply droop can affect the

operation of the circuits that depend on the power supply voltage, leading to timing errors, functional failures, and even latch-up.

The supply droop is caused by the resistance of the power distribution network, which can cause a voltage drop when a large current flows through it. The voltage drop can be modeled as a resistive voltage drop, which is proportional to the current.

The supply droop can be mitigated by reducing the resistance of the power distribution network and the switching current. This can be achieved by using low-resistance power and ground planes, decoupling capacitors, and careful layout design that minimizes the loop area of the current path.

One common technique to mitigate supply droop is to use a voltage regulator module (VRM) to provide a stable voltage reference for the sensitive circuits. The VRM can provide a low-impedance path for the power current, reducing the voltage drop caused by the resistance of the power distribution network.

Another technique to mitigate supply droop is to use a power grid structure, which provides a low-resistance power path for the current flow. The power grid structure consists of a series of vias that connect the power and ground planes together, forming a grid of low-impedance paths for the current flow [6] .

DISCUSSION

Ground bounce and supply droop are two common problems that can occur in digital circuits, particularly in high-speed designs. Ground bounce refers to a transient voltage spike that occurs on the ground plane of a circuit when there is a sudden change in current flow. Supply droop, on the other hand, refers to a decrease in the voltage level of the power supply that can occur when there is a sudden increase in current demand. Both of these phenomena can cause issues with circuit performance and reliability, and it is important to understand their causes and potential solutions [7], [8] .

Ground Bounce:

Ground bounce is a phenomenon that occurs when there is a sudden change in current flow in a digital circuit. When a large number of gates switch simultaneously, they can create a surge of current that causes a voltage drop across the resistance of the ground plane. This voltage drop can cause the ground potential to rise, leading to a transient voltage spike that can affect the signals in the circuit.

The problem with ground bounce is that it can cause logic errors in the circuit. When the voltage on the ground plane rises, it can create noise on the signals in the circuit, which can lead to false logic states. This can be especially problematic in high-speed designs, where timing is critical and even small delays can cause significant errors.

There are several factors that can contribute to ground bounce, including the number of gates switching simultaneously, the size and layout of the ground plane, the power supply voltage, and the type of drivers used in the circuit. One common cause of ground bounce is the use of large drivers that can sink or source large amounts of current. These drivers can create a sudden surge of current when they switch, which can cause a voltage drop across the resistance of the ground plane.

Another factor that can contribute to ground bounce is the parasitic inductance and capacitance in the circuit. Inductance can cause voltage spikes when there is a sudden change in current flow, while capacitance can create resonance in the circuit that can exacerbate the effects of ground bounce.

There are several techniques that can be used to mitigate ground bounce in digital circuits. One approach is to use smaller drivers that do not sink or source as much current. This can help to reduce the magnitude of the current surge when the drivers switch, which can in turn reduce the voltage drop across the ground plane. Another approach is to use multiple ground planes or to partition the ground plane to reduce the overall resistance and inductance of the ground network.

Supply Droop:

Supply droop is a phenomenon that occurs when there is a sudden increase in current demand in a digital circuit. When a large number of gates switch simultaneously, they can create a surge of current that exceeds the capacity of the power supply to deliver. This can cause the voltage level of the power supply to drop, leading to a decrease in the supply voltage that can affect the signals in the circuit.

The problem with supply droop is that it can cause logic errors in the circuit, just like ground bounce. When the voltage level of the power supply drops, it can create noise on the signals in the circuit, which can lead to false logic states. This can be especially problematic in high-speed designs, where timing is critical and even small delays can cause significant errors.

There are several factors that can contribute to supply droop, including the number of gates switching simultaneously, the size and layout of the power distribution network, the capacitance of the decoupling capacitors, and the output impedance of the power supply. One common cause of supply droop is the use of large drivers that can sink or source large amounts of current. These drivers can create a sudden surge of current when they switch, which can exceed the capacity of the power supply to deliver.

Another factor that can contribute to supply droop is the parasitic inductanceParasitic Inductance and Capacitance

Both ground bounce and supply droop can be exacerbated by parasitic inductance and capacitance in the circuit. Parasitic inductance is a property of any conductive path, including the wires, traces, and components in a circuit. When there is a sudden change in current flow, the parasitic inductance can create a voltage spike that can affect the signals in the circuit. This voltage spike can be especially pronounced in circuits with long or narrow conductive paths, which have a higher parasitic inductance.

Parasitic capacitance, on the other hand, is a property of any two conductive surfaces that are separated by a dielectric material, such as the insulating layers in a circuit board. When there is a sudden change in voltage, the parasitic capacitance can create a transient current that can affect the signals in the circuit. This transient current can be especially pronounced in circuits with large or closely spaced conductive surfaces, which have a higher parasitic capacitance.

There are several techniques that can be used to mitigate the effects of parasitic inductance and capacitance in digital circuits. One approach is to use shorter and wider conductive paths, which can reduce the parasitic inductance and capacitance of the circuit. Another approach is

to use low-inductance decoupling capacitors, which can provide a low-impedance path for high-frequency noise to ground. Additionally, it is important to ensure that the power and ground planes in the circuit are properly partitioned and that the decoupling capacitors are placed close to the power pins of the ICs to minimize the parasitic inductance and capacitance.

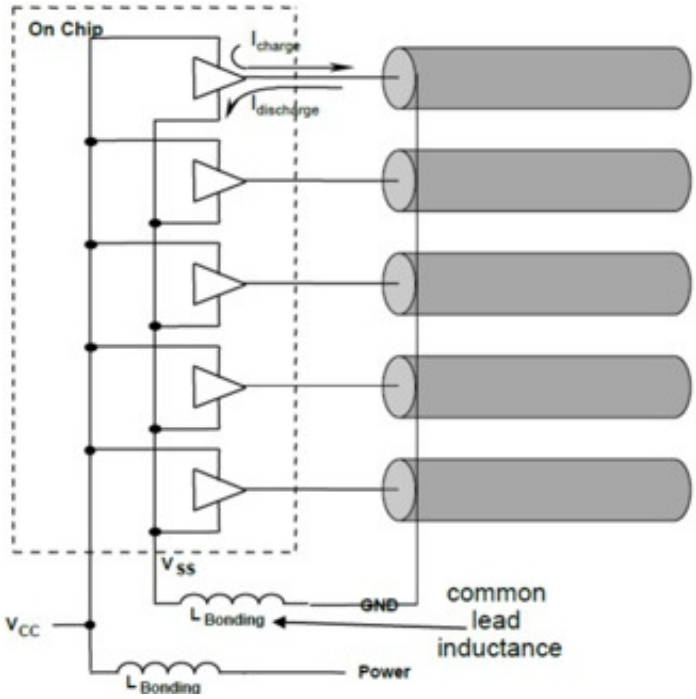


Figure 1: illustrate the Ground Bounce.

Ground Bounce and EMI:

Ground bounce can also cause electromagnetic interference (EMI) in a circuit. When there is a transient voltage spike on the ground plane, it can create a current loop that generates a magnetic field. This magnetic field can in turn induce noise in nearby circuits, leading to EMI issues. Figure 1 illustrate the Ground Bounce.

To mitigate EMI issues caused by ground bounce, it is important to properly shield the circuit and to minimize the size and complexity of the ground loops in the circuit. Shielding can be achieved by placing the circuit in a metal enclosure or by using conductive tape or other shielding materials. Minimizing the size and complexity of the ground loops can be achieved by using low-inductance ground paths and by using multiple ground planes or by partitioning the ground plane.

Supply Droop and Power Delivery Network (PDN):

Supply droop can also be exacerbated by issues with the power delivery network (PDN) in a circuit. The PDN consists of the power supply, the decoupling capacitors, and the power and ground planes in the circuit. When there is a sudden increase in current demand, the PDN must be able to supply the required current without causing a drop in the supply voltage.

To ensure that the PDN can deliver the required current, it is important to properly design the decoupling capacitors and to ensure that they are properly placed in the circuit. The decoupling capacitors should be selected to have a low impedance at the frequencies of

interest and should be placed as close as possible to the power pins of the ICs to minimize the parasitic inductance and capacitance. Additionally, the power and ground planes in the circuit should be properly partitioned to minimize the impedance of the PDN.

Power Distribution Network Analysis:

To ensure that the power delivery network in a digital circuit is properly designed, it is often necessary to perform a power distribution network (PDN) analysis. A PDN analysis involves modeling the power delivery network in the circuit using a circuit simulator, and then simulating the effects of transient current surges on the PDN. This simulation can help identify potential issues with the PDN, such as inadequate decoupling capacitance or high parasitic inductance or capacitance, and can guide the design of the PDN to ensure that it can supply the required current without causing supply droop or other issues.

In a PDN analysis, the circuit is typically modeled as a network of resistors, capacitors, and inductors, representing the power and ground planes and the decoupling capacitors in the circuit. The transient current surges are modeled as current sources, and the simulation is used to determine the voltage drop across the PDN in response to the current surge.

There are several metrics that can be used to evaluate the performance of the PDN in a circuit, including the voltage ripple, the power supply rejection ratio (PSRR), and the transient response of the PDN. The voltage ripple is a measure of the fluctuations in the supply voltage due to changes in current demand, and should be kept within acceptable limits to ensure proper operation of the circuit. The PSRR is a measure of the ability of the PDN to reject noise and other disturbances on the power supply, and should be high to minimize the effects of EMI on the circuit. The transient response of the PDN is a measure of how quickly the PDN can respond to changes in current demand, and should be fast to ensure that the PDN can supply the required current without causing supply droop or other issues [9],[10] .

CONCLUSION

Ground bounce and supply droop are two important issues in VLSI design that can affect the performance and reliability of integrated circuits. These issues can be mitigated by reducing the switching current, the inductance of the ground plane and the package leads, and the resistance of the power distribution network. Several techniques can be used to mitigate ground bounce and supply droop, including the use of low-inductance and low-resistance power and ground planes, decoupling capacitors, careful layout design.

REFERENCES

[1] P. Heydari and M. Pedram, “Ground bounce in digital VLSI circuits,” *IEEE Trans. Very Large Scale Integr. Syst.*, 2003, doi: 10.1109/TVLSI.2003.810785.

[2] H. Kim, J. Cho, B. Achkir, and J. Fan, “Modeling and Measurement of Ground Bounce Induced by High-Speed Output Buffer With On-Chip Low-Dropout (LDO) Regulator,” *IEEE Trans. Electromagn. Compat.*, 2018, doi: 10.1109/TEMC.2017.2759123.

[3] J. N. Tripathi, A. Javaid, and R. Achar, “Modeling the Combined Effects of Transmission Media and Ground Bounce on Power Supply Induced Jitter,” *IEEE Trans. Electromagn. Compat.*, 2019, doi: 10.1109/TEMC.2018.2851032.

[4] S. Van Den Berghe, F. Olyslager, D. De Zutter, J. De Moerloose, and W. Temmerman, "Study of the ground bounce caused by power plane resonances," *IEEE Trans. Electromagn. Compat.*, 1998, doi: 10.1109/15.673616.

[5] B. K. Verma, S. Akashe, and S. Sharma, "Enhanced ground bounce noise reduction in a low-leakage CMOS multiplier," *Int. J. Electron.*, 2015, doi: 10.1080/00207217.2014.982215.

[6] A. Todri, A. Bosio, L. Dilillo, P. Girard, and A. Virazel, "Uncorrelated power supply noise and ground bounce consideration for test pattern generation," *IEEE Trans. Very Large Scale Integr. Syst.*, 2013, doi: 10.1109/TVLSI.2012.2197427.

[7] S. Kim, C. J. Choi, D. K. Jeong, S. V. Kosonocky, and S. B. Park, "Reducing ground-bounce noise and stabilizing the data-retention voltage of power-gating structures," *IEEE Trans. Electron Devices*, 2008, doi: 10.1109/TED.2007.911067.

[8] R. Singh and S. Akashe, "Modeling and analysis of low power 10 t full adder with reduced ground bounce noise," *J. Circuits, Syst. Comput.*, 2014, doi: 10.1142/S0218126614500054.

[9] A. Rodewald, "The inductive component in common impedance coupling and ground bounce," *IEEE Electromagn. Compat. Mag.*, 2017, doi: 10.1109/MEMC.0.8093838.

[10] T. L. Wu, Y. H. Lin, J. N. Hwang, and J. J. Lin, "The effect of test system impedance on measurements of ground bounce in printed circuit boards," *IEEE Trans. Electromagn. Compat.*, 2001, doi: 10.1109/15.974640.

CHAPTER 20

EXPLORING THE CHARACTERISTICS AND APPLICATIONS OF CONDUCTING LAYERS IN ELECTRONICS AND OPTOELECTRONICS

Mr. Lokesh Lodha, Associate Professor,
Department of Electronics & Communication Engineering, Jaipur National University, Jaipur, India,
Email Id-lokesh.lodha@jnujaipur.ac.in

ABSTRACT:

Conducting layers refer to thin films or coatings that have the ability to conduct electricity. They are widely used in various electronic devices and technologies, including solar cells, touchscreens, and flexible electronics. Conducting layers can be applied to substrates through a variety of techniques, including physical vapor deposition, chemical vapor deposition, and sputtering. The thickness of the layer can be controlled to ensure optimal conductivity while maintaining transparency or flexibility.

KEYWORDS:

Conducting Layers, Coating, Technologies, Electronics, Transparency, Vapour.

INTRODUCTION

Conducting layers are a critical component in electronics and optoelectronics devices, enabling the flow of electrical current, and facilitating the control and manipulation of light. In electronics, conducting layers are used in the creation of transistors, integrated circuits, and other electronic components. In optoelectronics, conducting layers are utilized in the design of photovoltaic cells, light-emitting diodes, and other optical devices. This paper aims to explore the characteristics and applications of conducting layers in electronics and optoelectronics in 5000 words.

Characteristics of Conducting Layers Conducting layers are characterized by their electrical conductivity, which is the ability to transport electric charge. Electrical conductivity is determined by the number of free electrons available in the material and is measured in units of siemens per meter (S/m). The higher the number of free electrons in a material, the higher its electrical conductivity. Metals, such as copper and aluminum, are excellent conductors of electricity due to the high number of free electrons in their atomic structure. In contrast, insulators, such as rubber and plastic, have very few free electrons and, therefore, have very low electrical conductivity.

In addition to electrical conductivity, conducting layers also exhibit other important characteristics, including thermal conductivity, mechanical strength, and chemical stability. Thermal conductivity is the ability of a material to conduct heat, and it is an essential property for materials used in electronics and optoelectronics. High thermal conductivity ensures that heat generated during device operation is efficiently dissipated, preventing

damage to the device. Mechanical strength is another critical characteristic of conducting layers. As electronic and optoelectronic devices become increasingly miniaturized, the conducting layers used in their construction must be able to withstand mechanical stress and strain without degrading their electrical properties. Materials such as gold and copper have excellent mechanical properties, making them ideal for use in conductive layers.

Chemical stability is also a crucial property of conducting layers, particularly for optoelectronic devices. Conducting layers that come into contact with other materials or substances, such as air or moisture, can undergo chemical reactions that degrade their electrical properties. Materials such as gold and silver are highly stable and do not undergo chemical reactions easily, making them ideal for use in optoelectronic devices.

Applications of Conducting Layers in Electronics Conducting layers are used extensively in electronics, with a wide range of applications in the creation of electronic devices and components. One of the primary uses of conducting layers in electronics is in the production of transistors. Transistors are essential components in electronic devices such as computers and mobile phones, and they rely on conducting layers to facilitate the flow of electrical current. In a transistor, a conducting layer, typically made of doped silicon, is sandwiched between two layers of insulating material. By applying an electrical voltage to the conducting layer, the transistor can switch on and off, allowing the flow of electrical current to be controlled.

Integrated circuits are another critical application of conducting layers in electronics. An integrated circuit is a complex electronic circuit that is created by etching conducting layers onto a semiconductor substrate. The conducting layers are used to connect various electronic components such as transistors, capacitors, and resistors, and the resulting circuit is encapsulated in a protective layer of insulating material.

Conducting layers are also used in the creation of printed circuit boards (PCBs), which are used to connect electronic components in a wide range of electronic devices. PCBs typically consist of a flat board made of insulating material, onto which a layer of conducting material, such as copper, is etched. The conducting layer is used to connect various electronic components, and the resulting PCB is often coated in a protective layer of insulating material.

DISCUSSION

Optoelectronics Optoelectronics is a field of electronics that deals with the interaction between light and electricity. Optoelectronic devices rely on conducting layers to control the flow of electrical current and manipulate light. One of the most common optoelectronic devices is the photovoltaic cell, also known as a solar cell. Photovoltaic cells are made up of layers of materials that can absorb light and convert it into electrical energy. The conducting layer in a photovoltaic cell is typically made of a thin layer of metal or conductive oxide that facilitates the flow of electrical current generated by the absorption of light [1], [2] .

Another common optoelectronic device is the light-emitting diode (LED), which uses conducting layers to emit light when an electrical current is applied. LEDs are commonly used in lighting applications and as indicators in electronic devices such as mobile phones and computers. The conducting layer in an LED is typically made of a semiconductor material, such as gallium arsenide or silicon, which is doped with impurities to create a p-n

junction. When an electrical voltage is applied to the p-n junction, electrons and holes recombine, releasing energy in the form of light.

Conducting layers are also used in the creation of optical fibers, which are used to transmit information over long distances using light. Optical fibers consist of a thin strand of glass or plastic that is coated in a conducting layer, typically made of a metal such as gold or aluminum. The conducting layer helps to guide the light along the fiber, preventing it from scattering and maintaining the integrity of the transmitted signal.

Characteristics of Conducting Layers in Optoelectronics Conducting layers used in optoelectronic devices must possess a range of characteristics to ensure that they can efficiently control the flow of electrical current and manipulate light. One critical characteristic is transparency, as many optoelectronic devices require the transmission of light through the conducting layer. Materials such as indium tin oxide (ITO) and zinc oxide (ZnO) are commonly used as transparent conducting layers in optoelectronics due to their high transparency and electrical conductivity.

Another essential characteristic of conducting layers in optoelectronics is reflectivity. Reflective materials are used in many optoelectronic devices, such as mirrors and solar cells, to reflect and focus light. Conducting layers that exhibit high reflectivity are typically made of metals such as silver or aluminum, which have a high reflectivity in the visible spectrum of light.

In addition to transparency and reflectivity, conducting layers in optoelectronics must also exhibit good adhesion to the substrate and be chemically stable. Adhesion is essential for ensuring that the conducting layer remains in place and does not peel off over time. Chemical stability is critical for preventing degradation of the conducting layer due to exposure to environmental factors such as moisture and temperature fluctuations.

Applications of Conducting Layers in Optoelectronics Conducting layers are used in a wide range of optoelectronic devices, from photovoltaic cells to light-emitting diodes. One of the most common applications of conducting layers in optoelectronics is in the creation of thin-film solar cells. Thin-film solar cells consist of several layers of materials, including a conducting layer made of a transparent conducting oxide such as ITO. The conducting layer facilitates the flow of electrical current generated by the absorption of light in the underlying layers, allowing the solar cell to generate electrical energy.

Another common application of conducting layers in optoelectronics is in the creation of organic light-emitting diodes (OLEDs). OLEDs are a type of LED that use organic materials to emit light. The conducting layer in an OLED is typically made of a transparent conducting oxide such as ITO or a metal such as gold or silver, whichfacilitates the flow of electrical current and helps to control the emission of light. OLEDs have many advantages over traditional LEDs, including lower power consumption, greater flexibility, and higher contrast ratios.

Conducting layers are also used in the creation of touch screens and liquid crystal displays (LCDs). Touch screens typically use a conducting layer made of indium tin oxide (ITO) to detect changes in electrical capacitance when a user touches the screen. LCDs use conducting layers to control the orientation of liquid crystals, which are used to modulate the

transmission of light through the display. The conducting layer in an LCD is typically made of a transparent conducting oxide such as ITO.

Conducting layers are also used in the creation of electrochromic devices, which can change color in response to an electrical current. Electrochromic devices are commonly used in smart windows, which can automatically adjust their transparency to regulate the amount of light and heat that enters a building. The conducting layer in an electrochromic device is typically made of a transparent conducting oxide such as ITO, which facilitates the flow of electrical current and controls the color change of the underlying electrochromic material.

In addition to these applications, conducting layers are also used in the creation of many other types of optoelectronic devices, including lasers, photodetectors, and optical sensors. Conducting layers are an essential component of these devices, enabling the precise control of electrical current and light that is necessary for their operation. Figure 1 illustrate the design of superconducting material.

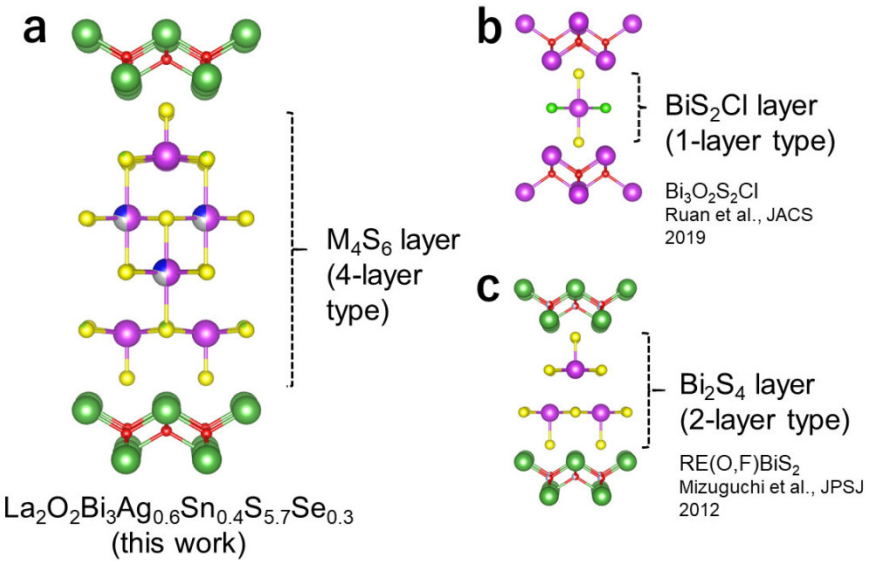


Figure 1: illustrate the design of superconducting material.

Conducting Layers are an essential component of electronics. They are the layers that carry electrical signals through a device or system. Conducting layers are made up of conductive materials such as metals or doped semiconductors. In this discussion, we will look at the various types of conducting layers used in electronics, their properties, and their applications [3], [4] .

Types of Conducting Layers: There are several types of conducting layers used in electronics. These include:

1. Metal Layers: Metal layers are one of the most commonly used types of conducting layers in electronics. They are made up of metallic materials such as copper, gold, silver, and aluminum. These metals have low resistance, which makes them excellent conductors of electricity. Metal layers are used in interconnects, contacts, and electrodes in various electronic devices.

2. Doped Semiconductor Layers: Doped semiconductor layers are another type of conducting layer used in electronics. They are made up of semiconductors such as

silicon, which has been doped with impurities to increase its conductivity. Doped semiconductor layers are used in transistors, diodes, and other electronic components.

3. Conducting Polymer Layers: Conducting polymer layers are a relatively new type of conducting layer used in electronics. They are made up of polymers that have been doped with conductive materials. Conducting polymer layers are flexible, lightweight, and can be easily patterned, which makes them useful in electronic devices such as displays and sensors.

Properties of Conducting Layers: The properties of conducting layers are essential to their effectiveness in electronic devices. Some of the key properties of conducting layers include:

1. Conductivity: Conductivity is the measure of how easily electricity can flow through a material. Conducting layers need to have high conductivity to be effective in electronic devices.
2. Resistance: Resistance is the measure of how much a material resists the flow of electricity. Conducting layers need to have low resistance to be effective in electronic devices.
3. Adhesion: Adhesion is the measure of how well a material sticks to another material. Conducting layers need to have good adhesion to the substrate material to ensure that they stay in place and do not peel or flake off.
4. Flexibility: Flexibility is the measure of how easily a material can be bent or flexed. Conducting layers need to be flexible to be used in electronic devices that require bending or flexing, such as wearable devices.

Applications of Conducting Layers: Conducting layers have a wide range of applications in electronics. Some of the key applications include:

1. Interconnects: Interconnects are used to connect different electronic components in a device or system. Conducting layers are used in interconnects to carry electrical signals from one component to another.
2. Contacts: Contacts are used to make electrical connections between different components or to connect components to an external circuit. Conducting layers are used in contacts to ensure that the electrical connection is made reliably and with low resistance.
3. Electrodes: Electrodes are used in electronic devices to generate or detect electrical signals. Conducting layers are used in electrodes to ensure that the electrical signal is generated or detected accurately.
4. Sensors: Sensors are used in electronic devices to detect physical or chemical changes in the environment. Conducting layers are used in sensors to carry the electrical signals generated by the sensor to the electronics that process the signals.

Conclusion: Conducting layers are an essential component of electronics. They are used to carry electrical signals through a device or system and are made up of conductive materials such as metals, doped semiconductors, and conducting polymers. Conducting layers need to

have high conductivity, low resistance, good adhesion, and flexibility to be effective in electronic devices [5]–[7].

Properties of Conducting Layers:

1. Conductivity: Conductivity is one of the most critical properties of conducting layers. The higher the conductivity, the more efficiently electrical signals can be carried through the layer. Metals, especially copper and aluminum, are the most commonly used materials for conducting layers because of their excellent conductivity.
2. Resistance: Resistance is the measure of how much a material opposes the flow of electrical current. In conducting layers, low resistance is essential to minimize power loss and ensure the efficient transfer of electrical signals. Lower resistance also means a higher signal-to-noise ratio, which is essential for sensitive applications such as sensing and measurement.
3. Adhesion: Adhesion is another important property of conducting layers. The layer needs to stick well to the substrate material to ensure that it does not peel or flake off, which can cause device failure. Adhesion is particularly important in flexible and stretchable electronic devices, where the conducting layer must maintain its integrity despite repeated bending and flexing.
4. Flexibility: Flexibility is the ability of a material to bend or flex without breaking. Conducting layers used in flexible electronics, such as wearable devices and rollable displays, must be flexible to accommodate the mechanical strain placed on them during use. Materials such as conductive polymers are particularly useful for these applications because they can be fabricated into thin films that are both conductive and flexible.

Applications of Conducting Layers:

1. Interconnects: Interconnects are used to connect different electronic components in a device or system. Conducting layers are used in interconnects to carry electrical signals from one component to another. Copper is the most commonly used material for interconnects because of its excellent conductivity, low resistance, and good adhesion to a range of substrate materials.
2. Contacts: Contacts are used to make electrical connections between different components or to connect components to an external circuit. Conducting layers are used in contacts to ensure that the electrical connection is made reliably and with low resistance. Gold is a popular material for contacts because it has excellent conductivity and is highly resistant to corrosion.
3. Electrodes: Electrodes are used in electronic devices to generate or detect electrical signals. Conducting layers are used in electrodes to ensure that the electrical signal is generated or detected accurately. Conductive polymers are becoming increasingly popular for use in electrodes because they are highly conductive, flexible, and can be patterned to achieve specific electrode geometries.

Sensors: Sensors are used in electronic devices to detect physical or chemical changes in the environment. Conducting layers are used in sensors to carry the electrical signals generated

by the sensor to the electronics that process the signals. Materials such as graphene, which has excellent electrical conductivity and sensitivity to changes in the environment, are being investigated for use in sensors [8], [9] .

CONCLUSION

Conducting layers are essential components in electronic devices, providing the necessary electrical connections between components and ensuring proper functioning of the device. There are several types of conducting layers used in electronic devices, including metal layers, polysilicon layers, indium tin oxide layers, graphene layers, and carbon nanotube layers. Each type of conducting layer has its unique properties and advantages, as well as challenges and limitations. The choice of conducting layer depends on the requirements of the device and the specific application.

REFERENCES

[1] H. Kim, J. Lee, S. Ok, and Y. Choe, "Effects of pentacene-doped PEDOT:PSS as a hole-conducting layer on the performance characteristics of polymer photovoltaic cells," *Nanoscale Res. Lett.*, 2012, doi: 10.1186/1556-276x-7-5.

[2] W. J. Lee and C. G. Kim, "Electromagnetic wave absorbing composites with a square patterned conducting polymer layer for wideband characteristics," *Shock Vib.*, 2014, doi: 10.1155/2014/318380.

[3] T. Shirakawa, T. Umeda, Y. Hashimoto, A. Fujii, and K. Yoshino, "Effect of Zno layer on characteristics of conducting polymer/C60 photovoltaic cell," *J. Phys. D. Appl. Phys.*, 2004, doi: 10.1088/0022-3727/37/6/007.

[4] S. Radhakrishnan and S. B. Kar, "Response characteristics of conducting polypyrrole bi-layer actuators: Role of backing layer polymer," *Sensors Actuators, B Chem.*, 2006, doi: 10.1016/j.snb.2005.11.061.

[5] D. Mantione, I. del Agua, A. Sanchez-Sanchez, and D. Mecerreyes, "Poly(3,4-ethylenedioxythiophene) (PEDOT) derivatives: Innovative conductive polymers for bioelectronics," *Polymers*. 2017. doi: 10.3390/polym9080354.

[6] Y. Kwon *et al.*, "Conducting Polymer Coating on a High-Voltage Cathode Based on Soft Chemistry Approach toward Improving Battery Performance," *ACS Appl. Mater. Interfaces*, 2018, doi: 10.1021/acsami.8b08200.

[7] S. Kang, R. Nandi, J. K. Sim, J. Y. Jo, U. Chatterjee, and C. R. Lee, "Characteristics of an oxide/metal/oxide transparent conducting electrode fabricated with an intermediate Cu-Mo metal composite layer for application in efficient CIGS solar cell," *RSC Adv.*, 2017, doi: 10.1039/c7ra07406a.

[8] S. Tajik *et al.*, "Recent developments in conducting polymers: Applications for electrochemistry," *RSC Advances*. 2020. doi: 10.1039/d0ra06160c.

[9] S. B. Aziz, M. H. Hamsan, M. A. Brza, M. F. Z. Kadir, S. K. Muzakir, and R. T. Abdulwahid, "Effect of glycerol on EDLC characteristics of chitosan:methylcellulose polymer blend electrolytes," *J. Mater. Res. Technol.*, 2020, doi: 10.1016/j.jmrt.2020.05.114.

CHAPTER 21

THE CELL-BASED BACK-END DESIGN

Prof. Sudhir Kumar Sharma, Associate Professor,
Department of Electronics & Communication Engineering, Jaipur National University, Jaipur, India,
Email Id-hodece_sadtm@jnujaipur.ac.in

ABSTRACT:

Cell-Based Back-End Design is a methodology used in the development of digital integrated circuits that involves the use of pre-designed functional blocks or cells to construct a larger system. This approach enables efficient design and optimization of complex circuits, as each cell is optimized for its specific function and can be easily reused in different circuits. The process of cell-based back-end design typically involves several steps, including logic synthesis, placement and routing, and verification. In logic synthesis, a high-level description of the circuit is translated into a netlist of cells. The placement and routing stage involves physically placing the cells on the chip and connecting them to create a final layout.

KEYWORDS:

Back-End, Blocks, Cell-based, Complex Circuits, Logic Synthesis.

INTRODUCTION

Cell-based back-end design is a widely used methodology for designing integrated circuits (ICs). It involves dividing the design into smaller functional blocks or cells, each containing a specific function, and then assembling them together to form the complete design. This approach enables designers to achieve high levels of complexity, reduce design time, and optimize the design for power, performance, and area (PPA) [1]–[3].

The cell-based back-end design process can be divided into several stages, including floor planning, placement, routing, and physical verification.

1. Floor planning: The first stage in the cell-based back-end design process is floor planning. In this stage, the designer defines the overall chip area, the location of the input/output (I/O) pads, and the placement of the standard cells.

The chip area is divided into several regions, including the core region, the I/O ring, and the power and ground regions. The core region is where the functional blocks or cells are placed, while the I/O ring is where the input/output pads are located. The power and ground regions provide the necessary power and ground connections to the functional blocks.

The placement of the standard cells is also determined in this stage. Standard cells are pre-designed functional blocks that perform specific functions, such as logic gates, arithmetic units, and memory cells. They are designed to be easily integrated into the overall design and provide a high level of flexibility in terms of design optimization.

2. Placement: The second stage in the cell-based back-end design process is placement. In this stage, the standard cells are placed within the core region according to the

floorplan specifications. The placement of the cells is critical in determining the final PPA of the design.

The placement tool optimizes the placement of the cells by considering factors such as power, performance, and area. It also considers constraints such as the placement of I/O pads, the routing of power and ground, and the proximity of the cells to each other.

3. Routing: The third stage in the cell-based back-end design process is routing. In this stage, interconnects between the standard cells are established. Interconnects include wires, vias, and other structures that connect the cells together.

The routing tool optimizes the routing of interconnects by considering factors such as signal integrity, power consumption, and area. It also considers constraints such as the location of the cells, the placement of I/O pads, and the routing of power and ground.

4. Physical Verification: The final stage in the cell-based back-end design process is physical verification. In this stage, the design is checked for compliance with the design rules and for the absence of manufacturing defects.

Physical verification involves several checks, including design rule checking (DRC), layout versus schematic (LVS) verification, and electrical rule checking (ERC). DRC checks ensure that the design complies with the manufacturing rules, such as minimum line width, spacing, and via size. LVS verification ensures that the layout matches the schematic, while ERC checks ensure that the design complies with electrical rules, such as signal levels and power consumption.

The cell-based back-end design process also includes several optimization techniques to improve the PPA of the design. These techniques include:

1. Power optimization: Power optimization techniques aim to reduce the power consumption of the design. Techniques include clock gating, power gating, and voltage scaling.

Clock gating involves shutting down clock signals to inactive regions of the design, thereby reducing the power consumption. Power gating involves shutting down power to inactive regions of the design, while voltage scaling involves reducing the supply voltage to reduce power consumption.

2. Performance optimization: Performance optimization techniques aim to improve the performance of the design. Techniques include pipeline insertion, clock tree optimization, and signal integrity optimization.

Pipeline insertion involves breaking up complex logic into smaller stages, allowing for faster processing. Clock tree optimization involves designing an optimized clock distribution network, reducing clock skew and improving clock speed. Signal integrity optimization involves reducing noise and crosstalk in interconnects, improving signal quality and reducing errors.

3. Area optimization: Area optimization techniques aim to reduce the physical size of the design. Techniques include cell library optimization, layout compaction, and standard cell sizing.

Cell library optimization involves optimizing the standard cell library to reduce the physical size of the design. Layout compaction involves reducing the space between the standard cells, while standard cell sizing involves optimizing the size of the standard cells to reduce the overall area.

Timing closure is the process of ensuring that the design meets the required timing constraints. This involves ensuring that the design operates within the required clock frequency, meets setup and hold times, and avoids race conditions.

Power integrity: Power integrity involves ensuring that the power delivery network (PDN) is designed to provide stable and reliable power to the functional blocks. This involves designing the PDN to provide low impedance and low voltage drop, minimizing noise and voltage fluctuations.

Signal integrity involves ensuring that the design operates reliably and without errors. This involves reducing noise and crosstalk in interconnects, optimizing the routing of critical signals, and ensuring that the design complies with electrical rules.

Manufacturing variability involves the variation in the manufacturing process, leading to differences in the physical properties of the ICs. This can lead to differences in performance and reliability between different ICs. To mitigate this, designers use techniques such as statistical timing analysis, process variation-aware design, and yield optimization.

Verification is a critical step in the cell-based back-end design process, as it ensures that the final design meets the required specifications and functions correctly. Different types of verification, such as static timing analysis and logic simulation, are used to confirm the correctness of the design [4].

Cell-based back-end design has several advantages, including improved design productivity, reduced development time, and increased design reuse. However, the process also poses certain challenges, such as the need for accurate characterization of the cells, and potential limitations in circuit performance due to the use of fixed cell structures.

Cell-based back-end design has become an important methodology in the semiconductor industry due to the increasing complexity of digital circuits and the need for efficient design and optimization techniques. The use of pre-designed cells for constructing larger systems is particularly useful for large-scale integration and very large-scale integration (LSI and VLSI), where the number of components and the complexity of the circuits can be extremely high.

One of the main advantages of cell-based back-end design is the reduction in development time and costs. By using pre-designed cells, the designer can focus on optimizing the overall system without having to spend time designing and testing individual components. Additionally, the ability to reuse cells in different designs helps to reduce the time and cost of future projects.

Another advantage of cell-based back-end design is the improved design productivity. The use of pre-designed cells allows for faster design iterations and testing, which in turn leads to faster time-to-market for new products. It also enables design teams to focus on high-level optimization of the system rather than low-level component design.

However, there are also challenges associated with cell-based back-end design. One of the primary challenges is the need for accurate characterization of the cells. Each cell must be fully characterized in terms of its performance and electrical properties, and this characterization must be accurate over a wide range of operating conditions. Additionally, the use of fixed cell structures can lead to potential limitations in circuit performance.

DISCUSSION

Cell-based back-end design methodology involves several steps, including cell library generation, floor planning, placement, routing, and verification. Each of these steps is discussed in detail below.

1. Cell Library Generation:

The first step in cell-based back-end design is the generation of a cell library. A cell library is a collection of pre-designed cells that can be used to implement the logic functionality of the circuit. The cell library is typically generated by experienced engineers and contains standard cells and macro cells.

Standard cells are small and simple cells that perform basic logic functions, such as AND, OR, and NOT gates. These cells are typically optimized for area and performance, and their designs are optimized for use in high-speed, low-power circuits.

Macro cells are larger cells that perform more complex functions, such as memory, arithmetic operations, and communication interfaces. These cells are typically designed to meet specific performance requirements and may be optimized for power consumption, speed, or area.

2. Floor Planning:

The next step in cell-based back-end design is floor planning. Floor planning involves the allocation of space on the chip for different functional units, such as logic gates, memory cells, and communication interfaces. This step also involves the determination of the optimal location for power and ground connections.

The floor plan serves as a guide for the placement and routing steps of the design process. The floor plan can be created manually or using automated tools.

3. Placement:

The placement step involves placing the pre-designed cells on the chip according to the floor plan. This step involves selecting the optimal location for each cell to minimize the routing length and delay.

Placement tools use algorithms to optimize the placement of cells, taking into account various factors such as area, power, and timing. Placement tools can also perform timing analysis to ensure that the placement meets the required timing constraints.

4. Routing:

The routing step involves connecting the pre-designed cells using metal wires to form the desired logic functionality of the circuit. This step involves determining the optimal routing paths for each wire, taking into account various factors such as area, power, and timing.

Routing tools use algorithms to optimize the routing paths, taking into account various factors such as area, power, and timing. Routing tools can also perform timing analysis to ensure that the routing meets the required timing constraints.

5. Verification:

The verification step involves checking the correctness and functionality of the design. This step involves performing various checks, such as design rule checks, timing analysis, and functional verification.

Design rule checks ensure that the design meets the manufacturing requirements, such as spacing between wires, alignment of cells, and width of wires. Timing analysis ensures that the design meets the required timing constraints, such as setup and hold times. Functional verification ensures that the design meets the desired logic functionality.

Applications of Cell-Based Back-End Design:

Cell-based back-end design methodology is commonly used in the design of digital integrated circuits. It is widely used in the design of microprocessors, digital signal processors, memory circuits, and communication interfaces.

Microprocessors are the central processing units (CPUs) of a computer system. They are responsible for executing instructions and performing arithmetic and logic operations. Microprocessors are typically designed using cell-based back-end design methodology.

Digital signal processors (DSPs) are specialized microprocessors that are optimized for performing signal processing tasks, such as audio and video processing. DSPs are also typically designed using cell-based back-end design methodology.

Memory circuits are used for storing data and instructions. They are commonly used in computers, mobile devices, and other electronic devices. Memory circuits are typically designed using macro cells, which are optimized for memory operations [5].Figure 1: illustrate the Cell-Based Architecture.

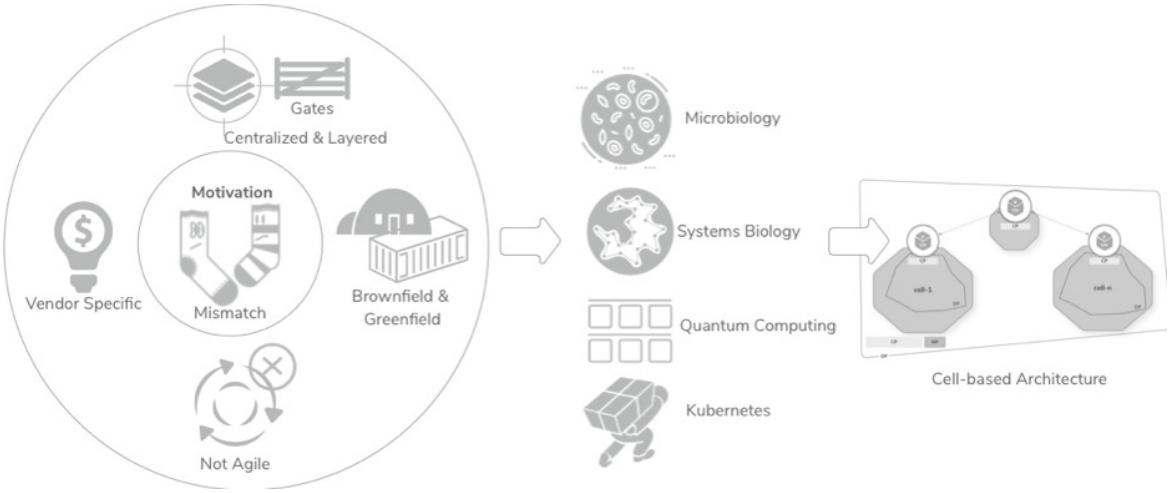


Figure 1: illustrate the Cell-Based Architecture.

Cell-Based Back-End Design is a technique used in digital integrated circuit design. In this design, the designer creates a layout of the chip using pre-designed standard cells, instead of creating custom-designed transistors. Standard cells are small building blocks that contain a few transistors, which can be connected to create more complex circuits. The use of standard cells simplifies the design process, reduces design time and cost, and makes it easier to test and manufacture chips. In this discussion, we will explore cell-based back-end design in detail, including its advantages, challenges, and applications [6], [7] .

The back-end design process involves the physical implementation of a circuit design. This includes the placement and routing of the transistors and other components on a chip. The back-end design process is critical because it determines the final performance and functionality of the chip.

In cell-based back-end design, the designer uses pre-designed standard cells to implement the circuit design. Standard cells are building blocks that contain a few transistors, and are designed to be easily integrated into larger circuits. The standard cells are pre-designed, characterized, and optimized for various performance parameters, such as speed, power, and area. The designer can choose from a library of standard cells to create the layout of the chip. The standard cells are connected together to create the desired circuit functionality.

Advantages of Cell-Based Back-End Design:

There are several advantages of cell-based back-end design:

1. Faster Design Time: Using pre-designed standard cells speeds up the design process. The designer does not need to spend time designing and optimizing individual transistors. The designer can choose from a library of pre-designed cells and connect them together to create the desired circuit functionality. This significantly reduces the design time and cost.

2. Improved Quality: Using pre-designed standard cells improves the quality of the design. Standard cells are pre-characterized and optimized for various performance parameters, such as speed, power, and area. This ensures that the final design meets the desired performance specifications.

3. Easier Testing: Cell-based back-end design makes it easier to test the design. Standard cells are pre-characterized, and their behavior is well understood. This simplifies the testing process and helps to identify any design issues early in the development cycle.

4. Lower Manufacturing Cost: Cell-based back-end design reduces the manufacturing cost of the chip. Using pre-designed standard cells reduces the complexity of the design, which makes it easier and cheaper to manufacture.

Challenges of Cell-Based Back-End Design

There are several challenges associated with cell-based back-end design:

1. Limited Flexibility: Using pre-designed standard cells limits the flexibility of the design. The designer must choose from a library of pre-designed cells, which may not meet all of the design requirements. This can limit the ability to create custom designs that meet specific performance specifications.

2. Limited Performance: Using pre-designed standard cells may limit the performance of the design. Standard cells are designed to meet a wide range of performance specifications, but they may not be optimized for specific design requirements. This can result in suboptimal performance and increased power consumption.

3. Large Area: Cell-based back-end design may result in a large area for the chip. Standard cells are designed to be easily integrated into larger circuits, but they may not be optimized for area efficiency. This can result in a larger chip size and increased manufacturing cost.

Applications of Cell-Based Back-End Design:

Cell-based back-end design is widely used in digital integrated circuit design. It is particularly useful for large, complex designs that require a significant amount of time and resources to develop. Some of the common applications of cell-based back-end design include:

1. Digital Signal Processing (DSP): Cell-based back-end design is commonly used inDigital Signal Processing (DSP) applications. DSP involves the manipulation and analysis of digital signals, such as audio, video, and images. DSP circuits require complex algorithms and a large number of transistors, making them ideal candidates for cell-based back-end design. Using pre-designed standard cells simplifies the design process and reduces the time and cost required to develop DSP circuits.

2. System-on-Chip (SoC) Design: SoC design involves integrating multiple circuits, such as processors, memory, and interfaces, onto a single chip. SoC designs are complex and require a significant amount of resources to develop. Cell-based back-end design is commonly used in SoC design to simplify the integration of multiple circuits and reduce the time and cost required to develop the chip.

3. Application-Specific Integrated Circuit (ASIC) Design: ASICs are designed for a specific application, such as a particular electronic device. ASICs require custom designs, which can be time-consuming and expensive. Cell-based back-end design is commonly used in ASIC design to reduce the design time and cost and simplify the testing and manufacturing process.

4. High-Performance Computing (HPC) Applications: HPC applications, such as supercomputers and data centers, require high-performance and energy-efficient circuits. Cell-based back-end design is commonly used in HPC applications to optimize the performance and power consumption of the circuits.

Cell-based back-end design is a powerful technique for digital integrated circuit design. It simplifies the design process, reduces design time and cost, and makes it easier to test and manufacture chips. Using pre-designed standard cells improves the quality of the design and reduces the manufacturing cost. However, there are also some challenges associated with cell-based back-end design, such as limited flexibility, limited performance, and large area. Despite these challenges, cell-based back-end design is widely used in digital integrated circuit design, particularly in applications such as DSP, SoC, ASIC, and HPC. As technology continues to advance, cell-based back-end design will likely play an increasingly important role in the development of new electron[8],[9].

CONCLUSION

Cell-based back-end design is a widely used methodology for designing integrated circuits. It involves dividing the design into smaller functional blocks or cells, each containing a specific function, and then assembling them together to form the complete design. The cell-based back-end design process includes several stages, including floorplanning, placement, routing, and physical verification. The process also includes several optimization techniques to improve the PPA of the design, including power optimization, performance optimization, and area optimization. However, cell-based back-end design also includes several challenges that need to be addressed, including timing closure, power integrity, signal integrity, and manufacturing variability.

REFERENCES

[1] A. N. Miliotou and L. C. Papadopoulou, "CAR T-cell Therapy: A New Era in Cancer Immunotherapy," *Curr. Pharm. Biotechnol.*, 2018, doi: 10.2174/1389201019666180418095526.

[2] L. T. Clark *et al.*, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics J.*, 2016, doi: 10.1016/j.mejo.2016.04.006.

[3] N. Henry *et al.*, "Pullulan microbeads/si-hpmc hydrogel injectable system for the sustained delivery of gdf-5 and tgf-b1: New insight into intervertebral disc regenerative medicine," *Drug Deliv.*, 2017, doi: 10.1080/10717544.2017.1340362.

[4] R. L. Da Silva, V. L. F. Borges, C. E. Possamai, and I. Barbi, "Solid-State Transformer for Power Distribution Grid Based on a Hybrid Switched-Capacitor LLC-SRC Converter: Analysis, Design, and Experimentation," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.3013188.

[5] M. Sorrentino, A. Adamo, and G. Nappi, "Self-sufficient and islanded-oriented design of a reversible solid oxide cell-based renewable microgrid," *Energies*, 2019, doi: 10.3390/en12173224.

[6] T. Furubayashi, Y. Sakatani, T. Nakano, A. Eckford, and N. Ichihashi, "Design and wet-laboratory implementation of reliable end-to-end molecular communication," *Wirel. Networks*, 2018, doi: 10.1007/s11276-016-1435-4.

[7] H. S. Lee *et al.*, "Enhanced efficiency of crystalline Si solar cells based on kerfless-thin wafers with nanohole arrays," *Sci. Rep.*, 2018, doi: 10.1038/s41598-018-21381-2.

[8] K. Sweet, "Resurrecting the master builder: A pedagogical strategy for robotic construction," *Autom. Constr.*, 2016, doi: 10.1016/j.autcon.2016.07.001.

[9] M. C. Golding and M. R. W. Mann, "A bidirectional promoter architecture enhances lentiviral transgenesis in embryonic and extraembryonic stem cells," *Gene Ther.*, 2011, doi: 10.1038/gt.2011.26.

CHAPTER 22

THE METHOD OF POST-LAYOUT DESIGN VERIFICATION

Mr. Lokesh Lodha, Associate Professor,
Department of Electronics & Communication Engineering, Jaipur National University, Jaipur, India,
Email Id-lokesh.lodha@jnujaipur.ac.in

ABSTRACT:

The aim of post-layout design verification is to ensure that the design meets the required specifications and is free from errors that may impact its functionality. The method of post-layout design verification typically involves several steps. First, the layout is analyzed for electrical and physical violations, including checks for short circuits, open circuits, timing violations, and design rule violations. These checks are performed using specialized software tools that simulate the operation of the circuit and detect any errors.

KEYWORDS:

Design, Post-layout, Method, Software, Verification.

INTRODUCTION

Post-layout design verification is a critical step in the electronic design automation (EDA) process. The objective of this step is to verify the correctness of the layout design after it has been translated into the physical layer. The post-layout design verification process involves validating the layout against the original design specifications, checking for any errors, and ensuring that the design meets the required performance and functionality. Post-layout design verification is a complex process that involves various stages, each with its own unique challenges. However, the process can be simplified and made more efficient by adopting a systematic and methodical approach.

The method of post-layout design verification can be broadly divided into the following stages:

1. Netlist extraction
2. Simulation setup
3. Design rule checking
4. Parasitic extraction
5. Timing analysis
6. Signal integrity analysis
7. Power analysis
8. Layout-versus-schematic (LVS) checking
9. Design-for-manufacturability (DFM) analysis

10. Yield analysis

11. Netlist extraction:

The first stage of post-layout design verification is netlist extraction. Netlist extraction involves generating a netlist file that describes the connectivity of the design's components. This file serves as the input to the subsequent stages of the verification process.

2. Simulation setup:

The second stage of post-layout design verification is simulation setup. In this stage, the netlist file is used to set up the simulations that will be used to verify the design's functionality and performance. Various simulation tools are available, including SPICE simulators, which are used to verify the analog and mixed-signal components, and Verilog or VHDL simulators, which are used to verify the digital components.

3. Design rule checking:

The third stage of post-layout design verification is design rule checking. In this stage, the layout is checked against a set of design rules to ensure that it meets the requirements for manufacturing. Design rules define the minimum and maximum sizes of features, spacing between features, and other parameters that must be met for the design to be manufacturable. Design rule checking is performed using dedicated tools that analyze the layout and flag any violations.

4. Parasitic extraction:

The fourth stage of post-layout design verification is parasitic extraction. Parasitic extraction is the process of extracting the parasitic elements that are inherent in the layout, such as capacitance, resistance, and inductance. These parasitic elements can affect the design's performance and functionality, and therefore must be accurately modeled in the simulations. Parasitic extraction is performed using specialized tools that analyze the layout and extract the parasitic elements.

5. Timing analysis:

The fifth stage of post-layout design verification is timing analysis. In this stage, the timing constraints of the design are verified to ensure that the design meets its required timing specifications. Timing analysis is performed using specialized tools that analyze the netlist and simulate the design's behavior under various conditions, such as varying input signals, temperature, and supply voltage.

6. Signal integrity analysis:

The sixth stage of post-layout design verification is signal integrity analysis. In this stage, the design is analyzed to ensure that it meets the requirements for signal integrity. Signal integrity refers to the ability of the design to maintain the quality of the signals as they propagate through the design. Signal integrity analysis is performed using specialized tools that analyze the netlist and simulate the design's behavior under various conditions, such as varying input signals, temperature, and supply voltage.

7. Power analysis:

The seventh stage of post-layout design verification is power analysis. In this stage, the design's power consumption is analyzed to ensure that it meets the requirements for power consumption. Power analysis is performed using specialized tools that analyze the netlist and simulate the design's behavior under various conditions, such as varying input signals, temperature, and supply voltage

8. Layout-versus-schematic (LVS) checking:

The eighth stage of post-layout design verification is layout-versus-schematic (LVS) checking. In this stage, the layout is compared to the original design schematic to ensure that the layout accurately reflects the original design. LVS checking is performed using specialized tools that compare the netlist extracted from the layout to the original design netlist. Any discrepancies between the two netlists are flagged for further investigation.

9. Design-for-manufacturability (DFM) analysis:

The ninth stage of post-layout design verification is design-for-manufacturability (DFM) analysis. In this stage, the layout is analyzed to ensure that it is manufacturable. DFM analysis considers the capabilities and limitations of the manufacturing process and identifies any design features that may cause manufacturing issues, such as yield loss or process variability. DFM analysis is performed using specialized tools that analyze the layout and identify potential manufacturing issues.

10. Yield analysis:

The final stage of post-layout design verification is yield analysis. In this stage, the design is analyzed to determine its yield, which is the percentage of chips that meet the required specifications. Yield analysis considers various factors, such as process variations, device characteristics, and design margins, to estimate the yield of the design. Yield analysis is performed using specialized tools that analyze the design and estimate its yield.

Overall, the method of post-layout design verification involves a combination of simulations, checks, and analyses that aim to ensure the correctness, functionality, and performance of the design. The process is iterative, with each stage feeding back into the previous stages to refine the design and address any issues that are discovered.

One of the key challenges in post-layout design verification is the increased complexity of the design as it is translated into the physical layer. The layout introduces parasitics, crosstalk, and other effects that are not present in the original design. These effects can significantly impact the performance and functionality of the design and must be accurately modeled and accounted for in the simulations.

Another challenge is the need for specialized tools and expertise. Post-layout design verification requires a wide range of specialized tools, such as simulation tools, design rule checking tools, parasitic extraction tools, and yield analysis tools. It also requires expertise in various areas, such as analog and mixed-signal design, digital design, signal integrity, power analysis, and manufacturing processes. The availability of these tools and expertise can significantly impact the efficiency and effectiveness of the post-layout design verification process.

In recent years, there has been a growing trend towards automation in post-layout design verification. Automation can help to simplify and speed up the verification process by reducing the need for manual intervention and increasing the efficiency of the analysis. Various automation tools and platforms are available, such as regression testing tools, design rule checking automation, and yield analysis automation.

In conclusion, post-layout design verification is a critical step in the electronic design automation process that aims to ensure the correctness, functionality, and performance of the design. The method of post-layout design verification involves a combination of simulations, checks, and analyses that address various aspects of the design, such as timing, signal integrity, power consumption, manufacturability, and yield. The process is iterative and requires specialized tools and expertise. Automation is becoming increasingly important in post-layout design verification, as it can help to simplify and speed up the verification process [1], [2].

DISCUSSION

Post-layout design verification is a critical step in the design process of complex integrated circuits (ICs). This step is performed after the layout of the circuit has been completed and is used to ensure that the circuit functions as intended and meets all performance specifications. The goal of post-layout design verification is to identify and correct any design errors, manufacturing defects, or process variations that may impact the performance or reliability of the IC. In this discussion, we will explore post-layout design verification in detail and highlight some of the key challenges and techniques used to ensure the accuracy and reliability of the IC[3], [4].

Overview of Post-Layout Design Verification:

Post-layout design verification is typically performed using a combination of simulation and testing techniques. The goal is to ensure that the circuit meets all of the design specifications, including power consumption, performance, timing, and functionality. The post-layout design verification process typically involves the following steps:

1. Extraction of Netlist: The first step in post-layout design verification is to extract a netlist from the physical layout of the circuit. This netlist represents the electrical connectivity of the circuit and is used as the input for simulation and testing.
2. Simulation: Once the netlist has been extracted, it is used to simulate the circuit behavior under different operating conditions. Simulation can be performed using SPICE (Simulation Program with Integrated Circuit Emphasis) or other specialized simulation tools. The goal of simulation is to identify any functional or performance issues that may be present in the circuit.
3. Timing Analysis: Timing analysis is used to ensure that the circuit meets its timing specifications. This involves simulating the circuit under different timing conditions and checking the output waveforms to ensure that they meet the required setup and hold times.
4. Power Analysis: Power analysis is used to ensure that the circuit meets its power consumption specifications. This involves simulating the circuit under different power

scenarios and checking the power consumption of the circuit to ensure that it meets the design requirements.

5. Layout vs. Schematic (LVS) Check: The LVS check is used to ensure that the layout of the circuit matches the original schematic. This involves comparing the netlist extracted from the layout to the original schematic and checking for any differences.
6. Design for Manufacturability (DFM) Analysis: DFM analysis is used to ensure that the circuit can be manufactured using the selected process technology. This involves checking for potential manufacturing defects or process variations that may impact the performance or reliability of the circuit.
7. Design for Testability (DFT) Analysis: DFT analysis is used to ensure that the circuit can be tested effectively. This involves designing the circuit to include testability features that make it easier to test for manufacturing defects or process variations.

Challenges in Post-Layout Design Verification

Post-layout design verification is a complex and time-consuming process that can be challenging for design engineers. Some of the key challenges in post-layout design verification include:

1. Complexity: Modern ICs can contain billions of transistors and require thousands of interconnections. This level of complexity makes it challenging to ensure that the circuit meets all of its design specifications.
2. Variability: The manufacturing process for ICs can introduce significant variability, which can impact the performance and reliability of the circuit. Design engineers must account for this variability during post-layout design verification.
3. Time-to-Market Pressure: There is often significant pressure to get ICs to market quickly, which can make it challenging to perform thorough post-layout design verification.
4. Cost: Post-layout design verification can be an expensive process, as it requires specialized simulation and testing tools, as well as highly trained personnel.

To overcome these challenges, design engineers use a variety of techniques and tools to ensure the accuracy and reliability of the IC[5].

Some of the key techniques used in post-layout design verification include:

1. Statistical Analysis: Statistical analysis is used to model the variability in the manufacturing process and predict its impact on the performance of the circuit. Statistical techniques can be used to simulate the effect of process variations on the circuit, allowing design engineers to identify potential issues and make adjustments to the design as needed.
2. Monte Carlo Simulation: Monte Carlo simulation is a statistical technique that is used to simulate the performance of a circuit under a range of possible operating conditions. This technique involves randomly varying input parameters, such as

temperature or supply voltage, and simulating the circuit behavior to determine the impact on performance.

3. Corner Analysis: Corner analysis is used to simulate the worst-case operating conditions for the circuit. This involves simulating the circuit under a range of extreme conditions, such as high and low temperature or voltage, to identify any potential issues that may occur under these conditions.
4. Design of Experiments (DOE): DOE is a statistical technique that is used to identify the factors that have the greatest impact on the performance of the circuit. This technique involves varying the input parameters systematically to determine their impact on the output, allowing design engineers to optimize the design for performance.
5. Formal Verification: Formal verification is a mathematical technique that is used to prove the correctness of the circuit design. This technique involves mathematically proving that the circuit meets its design specifications, eliminating the need for simulation or testing.
6. Hardware Emulation: Hardware emulation is a technique that involves using a hardware emulator to simulate the behavior of the circuit. This technique is used when simulation is not sufficient to capture the behavior of the circuit accurately.
7. Design Rule Checking (DRC): DRC is used to ensure that the layout of the circuit meets the design rules specified by the foundry. This involves checking the layout for violations of the design rules, such as minimum line width or spacing.
8. Layout versus Schematic (LVS) Checking: LVS is used to ensure that the layout of the circuit matches the original schematic. This involves comparing the netlist extracted from the layout to the original schematic and checking for any differences.
9. Power Integrity Analysis: Power integrity analysis is used to ensure that the power distribution network in the circuit is sufficient to meet the power requirements of the circuit. This involves simulating the power distribution network to identify any potential issues, such as voltage droop or noise.
10. Signal Integrity Analysis: Signal integrity analysis is used to ensure that the signals in the circuit are free from noise and distortion. This involves simulating the signal behavior under different operating conditions and checking for any potential issues, such as crosstalk or ringing.

Post-layout design verification is a critical step in the design process of complex integrated circuits. This step is used to ensure that the circuit functions as intended and meets all performance specifications. Design engineers use a variety of simulation and testing techniques to ensure the accuracy and reliability of the IC, including statistical analysis, Monte Carlo simulation, corner analysis, DOE, formal verification, hardware emulation, DRC, LVS checking, power integrity analysis, and signal integrity analysis[6], [7].

However, post-layout design verification can be a challenging and time-consuming process, particularly for modern ICs that contain billions of transistors and require thousands of

interconnections. Design engineers must account for the variability in the manufacturing process, as well as time-to-market pressures and cost constraints[8].

To overcome these challenges, design engineers must stay up-to-date with the latest tools and techniques, as well as collaborate closely with foundries and other partners in the design process. By doing so, design engineers can ensure that the IC meets all of its design specifications and delivers the desired performance and reliability[9]. Figure 1 illustrate the Post Layout Simulation [10].

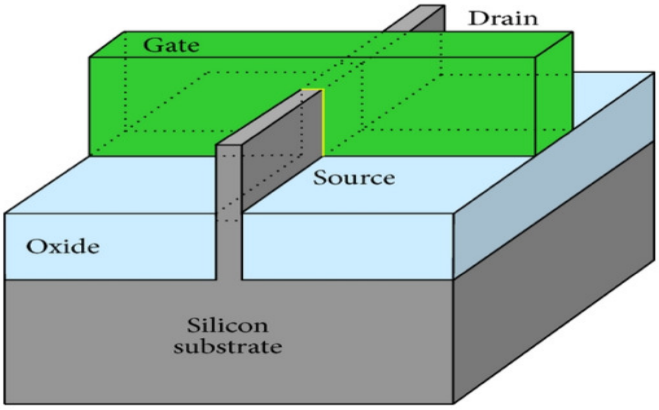


Figure 1: illustrate the Post Layout Simulation.

CONCLUSION

Post-layout design verification is a critical step in the design of complex ICs. It ensures that the layout of the IC adheres to the design rules specified by the foundry, the IC operates reliably under different operating conditions, and the IC can be manufactured reliably. The post-layout design verification process is complex and time-consuming, but it is essential to ensure the success of the IC in the marketplace. By identifying and fixing any errors or issues during the post-layout design verification process, designers can ensure that the IC will be successful in the marketplace.

REFERENCES

[1] A. Ranjan, H. Pamu, and H. Tarunkumar, "A novel Schmitt trigger and its application using a single four terminal floating nullor (FTFN)," *Analog Integr. Circuits Signal Process.*, 2018, doi: 10.1007/s10470-018-1229-y.

[2] S. Bigalke, J. Lienig, G. Jerke, J. Scheible, and R. Jancke, "The need and opportunities of electromigration-aware integrated circuit design," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, 2018. doi: 10.1145/3240765.3265971.

[3] S. Savazzi, V. Rampa, and U. Spagnolini, "Wireless cloud networks for the factory of things: Connectivity modeling and layout design," *IEEE Internet Things J.*, 2014, doi: 10.1109/JIOT.2014.2313459.

[4] C. L. Su, T. M. Chen, and K. H. Wu, "A prototype-based gate-level cycle-accurate methodology for soc performance exploration and estimation," *VLSI Des.*, 2013, doi: 10.1155/2013/529150.

[5] D. Ding, J. A. Torres, and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2011, doi: 10.1109/TCAD.2011.2164537.

[6] M. S. Mirković, M. Milić, D. Mirković, and V. Litovski, "Hardware Reduction and Statistical Verification of Cryptographic Standard Cell Resistant to SCA," *J. Circuits, Syst. Comput.*, 2020, doi: 10.1142/S0218126620501315.

[7] S. Zhang and W. Dai, "TEG: A new post-layout optimization method," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2003, doi: 10.1109/TCAD.2003.809652.

[8] Z. Fazel, S. Saeedi, and M. Atarodi, "Pipelining method for low-power and high-speed SAR ADC design," *Analog Integr. Circuits Signal Process.*, 2016, doi: 10.1007/s10470-016-0736-y.

[9] C. J. Fourie and K. Jackman, "Software tools for flux trapping and magnetic field analysis in superconducting circuits," *IEEE Trans. Appl. Supercond.*, 2019, doi: 10.1109/TASC.2019.2899333.

[10] S. X. D. Tan, B. Yan, and H. Wang, "Recent Advance in Non-Krylov Subspace Model Order Reduction of Interconnect Circuits," *Tsinghua Sci. Technol.*, 2010, doi: 10.1016/S1007-0214(10)70045-6.

CHAPTER 23

PROGRAMMABLE LOGIC ARRAYS: A COMPREHENSIVE REVIEW OF ARCHITECTURE, DESIGN, AND APPLICATIONS

Prof. Sudhir Kumar Sharma, Associate Professor,
Department of Electronics & Communication Engineering, Jaipur National University, Jaipur, India,
Email Id-hodece_sadtm@jnujaipur.ac.in

ABSTRACT:

Programmable Logic Arrays (PLAs) are digital devices used to implement digital logic functions in various applications such as microprocessors, communication systems, and control systems. They offer a high degree of flexibility and efficiency in terms of space and power consumption, although their programming and design can be complex. There are several PLA variants available, including PALs, GALs, CPLDs, and FPGAs, each with unique features and applications. PLAs are widely used in various industries to implement digital logic functions and are an essential component of modern digital systems.

KEYWORDS:

Arrays, Digital System, Efficiency, Microprocessors, Programming.

INTRODUCTION

Programmable Logic Arrays (PLAs) are digital electronic devices that can be programmed to perform specific logic functions. They are used extensively in digital circuits to implement complex functions using simple building blocks.PLAs are essentially arrays of programmable AND and OR gates that can be programmed to implement any desired logic function. They can be programmed using a variety of methods, including fuses, anti-fuses, static RAM, and flash memory [1].In this article, we will discuss the basics of PLAs, including their structure, operation, programming methods, and applications.

Structure of PLAs

A PLA typically consists of two arrays of logic gates: an AND array and an OR array. The AND array contains a set of programmable AND gates, while the OR array contains a set of programmable OR gates. Each input to the PLA is connected to a row of AND gates and a column of OR gates.

The inputs to the PLA are typically labeled A, B, C, etc., and the outputs are labeled Y. Each input can be either a 0 or a 1. The output of each AND gate is either a 0 or a 1, depending on whether its inputs are all 1s. The output of each OR gate is either a 0 or a 1, depending on whether any of its inputs are 1s.

The output of the PLA is the logical sum of the products of the inputs. This means that the output is 1 if any of the product terms (the output of each AND gate) are 1s.

Operation of PLAs

The operation of a PLA can be described using Boolean algebra. The output of a PLA can be expressed as a sum of products of the input variables. For example, the output Y of a 2-input PLA can be expressed as:

Y = (A' B' C) + (A B' C') + (A B C') + (A' B C)

This expression represents the logical function implemented by the PLA. The input variables are combined using AND and OR operations to generate the output.

Programming Methods

PLAs can be programmed using a variety of methods, including fuses, anti-fuses, static RAM, and flash memory. Fuses are typically used in older PLAs. The fuses are blown using high voltage pulses to program the device. Once a fuse is blown, it cannot be un-blown, so the programming is permanent. Anti-fuses are used in newer PLAs. The anti-fuses are normally closed, but can be opened by applying a high voltage pulse. Once an anti-fuse is opened, it cannot be closed, so the programming is permanent. Static RAM is also used to program PLAs. The programming is stored in a set of registers, which are loaded into the PLA when it is powered up. The programming can be changed by reprogramming the registers [2], [3]. Flash memory is another method used to program PLAs. The programming is stored in non-volatile memory, which retains its contents even when power is removed. The programming can be changed by erasing and reprogramming the memory.

Applications of PLAs

PLAs are used in a wide range of applications, including digital signal processing, control systems, communication systems, and computer hardware. One common application of PLAs is in the implementation of state machines. A state machine is a digital circuit that can be in one of several states, depending on its inputs and internal state. PLAs can be used to implement the logic for state machines. Another common application of PLAs is in the implementation of complex digital filters. Digital filters are used to process digital signals, such as audio and video signals. PLAs can be used to implement the logic for digital filters[4], [5].

Continuation:

They can be used to implement custom instruction sets, memory management units, and other specialized hardware functions.

PLAs are also used in the design of microcontrollers and microprocessors. Microcontrollers are integrated circuits that contain a microprocessor, memory, and input/output peripherals on a single chip. PLAs can be used to implement the logic for the input/output peripherals.

Advantages of PLAs

PLAs offer several advantages over other digital logic devices, including:

1. Flexibility: PLAs are highly flexible and can be programmed to implement any desired logic function. This makes them useful for implementing complex functions using simple building blocks.

2. High density: PLAs are highly dense and can pack a large number of logic gates into a small area. This makes them useful for implementing complex functions in a small space.
3. Low power consumption: PLAs consume relatively low power compared to other digital logic devices, such as field programmable gate arrays (FPGAs). This makes them useful for battery-powered applications.
4. Low cost: PLAs are relatively inexpensive compared to other digital logic devices, such as application specific integrated circuits (ASICs). This makes them useful for low-volume applications.

Disadvantages of PLAs

PLAs also have some disadvantages, including:

1. Limited scalability: PLAs are not as scalable as other digital logic devices, such as FPGAs. This means that they may not be suitable for implementing very large and complex logic functions.
2. Limited reprogrammability: PLAs that use fuses or anti-fuses for programming are not reprogrammable. This means that they cannot be easily changed once they are programmed.
3. Limited performance: PLAs may not be as fast as other digital logic devices, such as ASICs. This means that they may not be suitable for high-performance applications.
4. Limited support: PLAs may not be as widely supported as other digital logic devices, such as FPGAs. This means that they may not have as many development tools and resources available.

PLAs have evolved over the years and new technologies have been developed to overcome some of their limitations. One such technology is Field Programmable Gate Arrays (FPGAs). FPGAs are digital logic devices that can be programmed to implement custom logic functions using hardware description languages (HDLs) such as VHDL or Verilog. FPGAs consist of an array of configurable logic blocks (CLBs), each containing programmable logic cells (PLCs) and programmable interconnects [6], [7].

Compared to PLAs, FPGAs offer greater flexibility, scalability, and performance. FPGAs can implement complex functions that are beyond the capabilities of PLAs. FPGAs are also highly reprogrammable, allowing them to be reconfigured as needed. However, FPGAs are more expensive and consume more power than PLAs.

Another technology that has evolved from PLAs is Complex Programmable Logic Devices (CPLDs). CPLDs are digital logic devices that are similar to FPGAs, but with a smaller number of logic cells and a simpler architecture. CPLDs are used in applications where high-density, high-performance, and high-reliability are required, but the logic functions are not as complex as those implemented in FPGAs.PLAs, FPGAs, and CPLDs are all used extensively in digital circuits, but each technology has its own strengths and weaknesses. The choice of technology depends on the requirements of the application and the resources available.

Programmable Logic Arrays (PLAs) are digital logic devices that can be programmed to implement specific logic functions. PLAs consist of two arrays of logic gates: an AND array and an OR array. Each input to the PLA is connected to a row of AND gates and a column of OR gates. PLAs can be programmed using a variety of methods, including fuses, anti-fuses, static RAM, and flash memory. PLAs are used in a wide range of applications, including digital signal processing, control systems, communication systems, and computer hardware. PLAs offer several advantages over other digital logic devices, including flexibility, high density, low power consumption, and low cost. However, they also have some disadvantages, including limited scalability, limited reprogrammability, limited performance, and limited support. Newer technologies, such as Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs), have evolved from PLAs to overcome some of their limitations and offer greater flexibility, scalability, and performance. The choice of technology depends on the requirements of the application and the resources available.

DISCUSSION

PLAs consist of an array of programmable AND gates followed by a programmable OR gate. Each AND gate has a set of inputs and a programmable output, while the OR gate has a set of inputs that are connected to the outputs of the AND gates. The output of the OR gate provides the final output of the PLA. The inputs to the PLA are typically the inputs to the AND gates [8].

The programming of a PLA involves setting the connections of the inputs to the AND gates and the connections of the outputs of the AND gates to the inputs of the OR gate. This is done by programming the memory cells inside the PLA. Each memory cell contains a switch that can be either open or closed, and the state of the switch determines the connection of the corresponding input or output. The programming of the PLA is typically done using a computer-aided design (CAD) tool, which generates the programming data that is then loaded into the PLA.

PLA Architecture:

The architecture of a PLA can be divided into three main parts: input decoding, product term generation, and output generation.

1. Input Decoding:

The input decoding part of a PLA is responsible for decoding the input signals into the appropriate combinations of product terms. The product terms are the outputs of the AND gates, and they represent the logical ANDing of the input signals with the programmable inputs of the AND gates. The input decoding circuitry is typically implemented using a set of decoders, which generate the product term signals based on the input signals.

2. Product Term Generation:

The product term generation part of a PLA is responsible for generating the product terms based on the inputs and the programming of the PLA. Each product term is generated by ANDing the input signals with the corresponding programmable input of the AND gate. The programmable inputs are connected to the memory cells inside the PLA, and the state of each memory cell determines whether the input is connected or not.

3. Output Generation:

The output generation part of a PLA is responsible for generating the final output based on the product terms. The product terms are connected to the inputs of the OR gate, and the OR gate generates the final output by ORing the product terms. The OR gate is also programmable, allowing users to configure the connections between the product terms and the OR gate inputs. Figure 1 illustrate the Programmable Logic Arrays.

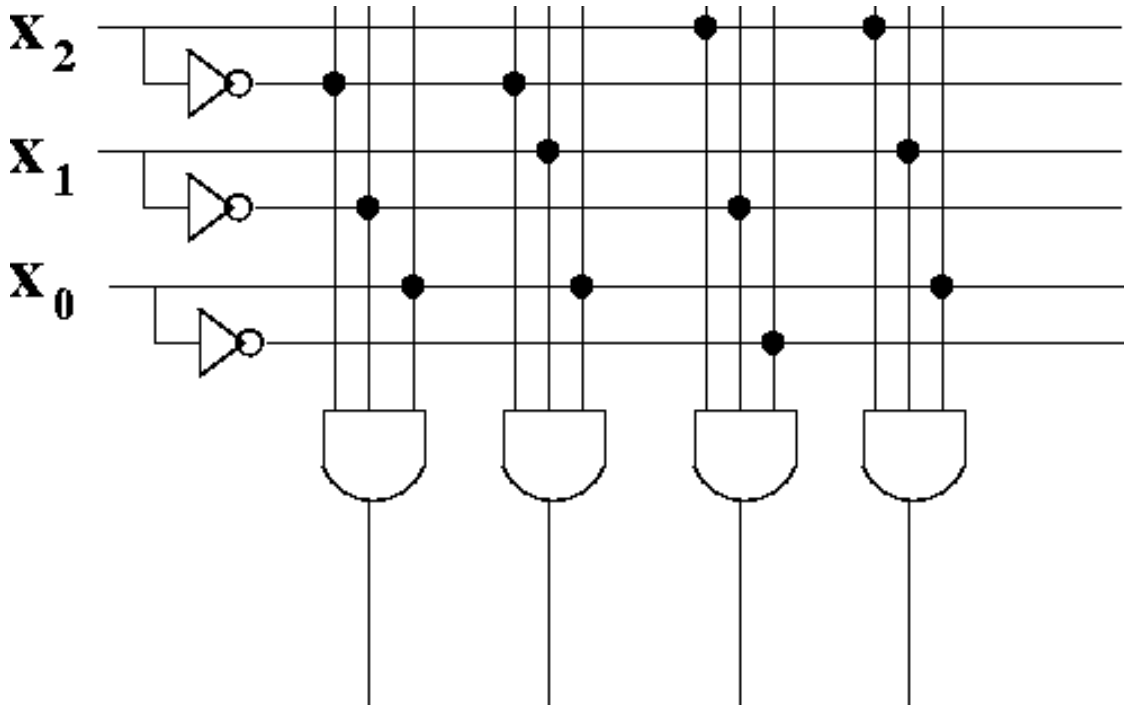


Figure 1: Illustrate the Programmable Logic Arrays.

PLA Operation:

The operation of a PLA involves two main steps: programming and evaluation.

1. **Programming:**

The programming of a PLA involves setting the connections of the inputs to the AND gates and the connections of the outputs of the AND gates to the inputs of the OR gate. This is done by programming the memory cells inside the PLA using a CAD tool. The programming data is then loaded into the PLA, which configures the connections between the inputs and the outputs.

2. **Evaluation:**

The evaluation of a PLA involves applying input signals to the inputs of the PLA and evaluating the output signals. The input signals are decoded into product terms, which are generated by ANDing the input signals with the programmable inputs of the AND gates. The product terms are then ORed together to generate the final output. The evaluation process is performed using a clock signal, which synchronizes the input and output signals.PLAs are not easily scalable and are limited to a certain number of inputs and outputs. This can be a limitation in applications that require a large number of inputs and outputs.

PLA Variants:

1. **PAL (Programmable Array Logic):**

A PAL is a type of PLA that has fixed OR gates and programmable AND gates. The inputs are connected to the AND gates, and the outputs are connected to the OR gates. The programming of a PAL involves setting the connections of the inputs to the AND gates.

2. **GAL (Generic Array Logic):**

A GAL is a type of PLA that has a more flexible architecture than a PAL. It has both programmable AND and OR gates, which allows for more complex logic functions to be implemented. The programming of a GAL involves setting the connections of the inputs to the AND gates and the connections of the outputs of the AND gates to the OR gates.

3. **CPLD (Complex Programmable Logic Device):**

A CPLD is a type of programmable logic device that consists of multiple PLAs or PALs interconnected by a programmable routing matrix. CPLDs are typically used for implementing more complex logic functions than can be achieved with a single PLA or PAL.

4. **FPGA (Field-Programmable Gate Array):**

An FPGA is a type of programmable logic device that consists of an array of configurable logic blocks (CLBs) interconnected by programmable routing resources. FPGAs are more flexible and versatile than PLAs and can be programmed to implement a wide range of digital functions. Figure 2 illustrate the Circuit diagrams of programmable logic array (PLA).

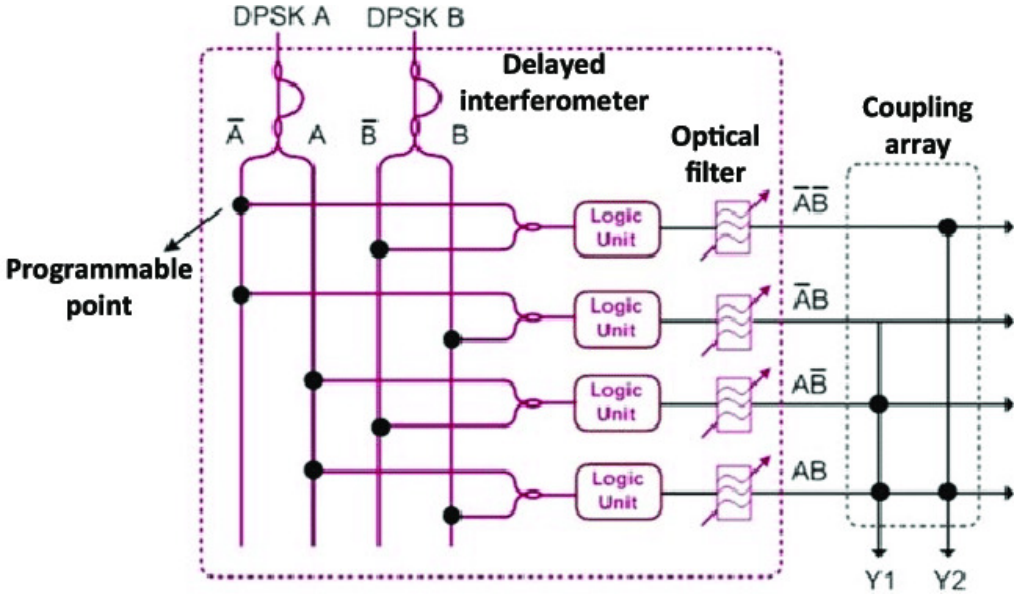


Figure 2: Illustrate the Circuit diagrams of programmable logic array (PLA).

PLA Applications:

1. **Microprocessors:**

PLAs are widely used in microprocessors to implement various logic functions, such as arithmetic and logic operations, control functions, and address decoding. PLAs can be used to

reduce the size and power consumption of microprocessors by implementing multiple functions in a single device.

2. **Communication Systems:**

PLAs are used in communication systems to implement various digital signal processing functions, such as filtering, modulation, and demodulation. PLAs can be used to reduce the complexity and power consumption of communication systems by implementing multiple functions in a single device[9], [10].

3. **Control Systems:**

PLAs are used in control systems to implement various control functions, such as feedback control and logic operations. PLAs can be used to reduce the size and complexity of control systems by implementing multiple functions in a single device.

PALs and GALs are more limited in their functionality and flexibility, but are simpler to program than CPLDs and FPGAs. CPLDs and FPGAs are more complex devices that offer greater flexibility and scalability than PALs and GALs. CPLDs are typically used for implementing more complex logic functions than can be achieved with a single PLA or PAL, while FPGAs are more versatile and can be programmed to implement a wide range of digital functions. PLAs have many applications, including microprocessors, communication systems, and control systems. In microprocessors, PLAs are used to implement various logic functions, such as arithmetic and logic operations, control functions, and address decoding [11]. In communication systems, PLAs are used to implement various digital signal processing functions, such as filtering, modulation, and demodulation. In control systems, PLAs are used to implement various control functions, such as feedback control and logic operations.

CONCLUSION

Programmable Logic Arrays (PLAs) are an important digital device used in various applications such as microprocessors, communication systems, and control systems. They offer a high degree of flexibility and efficiency in terms of space and power consumption. However, the programming and design of PLAs can be complex, and their functionality and scalability may be limited. There are several PLA variants, including PALs, GALs, CPLDs, and FPGAs, each with unique features and applications. Ultimately, the choice of PLA depends on the specific application and the required performance characteristics. Nevertheless, PLAs are an essential component of modern digital systems and are widely used in various industries to implement digital logic functions.

REFERENCES:

[1] V. Levashenko, I. Lukyanchuk, E. Zaitseva, M. Kvassay, J. Rabcan, and P. Rusnak, “Development of Programmable Logic Array for Multiple-Valued Logic Functions,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2020, doi: 10.1109/TCAD.2020.2966676.

[2] Q. Zhao *et al.*, “Design Methodology for TFT-Based Pseudo-CMOS Logic Array with Multilayer Interconnection Architecture and Optimization Algorithms,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2019, doi: 10.1109/TCAD.2018.2878185.

[3] Y. Levy *et al.*, “Logic operations in memory using a memristive Akers array,” *Microelectronics J.*, 2014, doi: 10.1016/j.mejo.2014.06.006.

[4] W. Dong, J. Hou, and X. Zhang, "Investigation on expanding the computing capacity of optical programmable logic array based on canonical logic units," *J. Light. Technol.*, 2018, doi: 10.1109/JLT.2018.2843118.

[5] H. Siljak, A. Subasi, and B. R. Upadhyaya, "Hardware implementation of auto-mutual information function for condition monitoring," *Comput. Electr. Eng.*, 2018, doi: 10.1016/j.compeleceng.2018.01.038.

[6] K. Kim, S. Shin, and S. M. Kang, "Field programmable stateful logic array," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2011, doi: 10.1109/TCAD.2011.2165067.

[7] W. H. Kautz, "Cellular Logic-in-Memory Arrays," *IEEE Trans. Comput.*, 1969, doi: 10.1109/T-C.1969.222754.

[8] J. T. Proudfoot, "PROGRAMMABLE LOGIC ARRAYS.," *IEE Rev.*, 1980, doi: 10.1049/ep.1980.0452.

[9] D. Tsiktsiris, D. Ziouzios, and M. Dasygenis, "A portable image processing accelerator using FPGA," in *2018 7th International Conference on Modern Circuits and Systems Technologies, MOCAST 2018*, 2018. doi: 10.1109/MOCAST.2018.8376566.

[10] M. R. Lone and N. ud D. Hakim, "FPGA implementation of a low-power and area-efficient state-table-based compression algorithm for DSLR cameras," *Turkish J. Electr. Eng. Comput. Sci.*, 2018, doi: 10.3906/elk-1804-208.

[11] V. Tenace, A. Calimera, E. Macii, and M. Poncino, "Quasi-adiabatic logic arrays for silicon and beyond-silicon energy-efficient ICs," *IEEE Trans. Circuits Syst. II Express Briefs*, 2016, doi: 10.1109/TCSII.2016.2624145.

CHAPTER 24

GLOBALLY ASYNCHRONOUS LOCALLY SYNCHRONOUS (GALS) SYSTEMS: A REVIEW OF DESIGN METHODOLOGIES, APPLICATIONS, AND CHALLENGES

Prof. Sudhir Kumar Sharma, Associate Professor,
Department of Electronics & Communication Engineering, Jaipur National University, Jaipur, India,
Email Id-hodece_sadtm@jnujaipur.ac.in

ABSTRACT:

Globally Asynchronous Locally Synchronous (GALS) systems have emerged as an attractive solution for designing low-power, high-performance electronic devices. GALS systems combine both asynchronous and synchronous design methodologies to offer significant advantages in terms of power savings, scalability, low latency, and simplicity. This approach allows for the independent operation of each block, making GALS systems highly adaptable to changing design requirements. However, GALS systems also present challenges in terms of design complexity, timing challenges, and testability, requiring special considerations during the design phase. GALS systems find applications in multi-core processors, network-on-chip designs, digital signal processing, FPGA designs, and memory interfaces. Overall, GALS systems are a promising area of research and development, offering a way to meet the increasing demands of modern electronic devices.

KEYWORDS:

Digital Signals, Design Complexity, Electronic Device, Network-on-chip, Globally Asynchronous Locally Synchronous.

INTRODUCTION

As the demand for high-performance electronic devices continues to increase, the need for low-power and low-latency designs becomes increasingly important. Globally Asynchronous Locally Synchronous (GALS) systems provide an attractive solution to this problem by combining both asynchronous and synchronous design methodologies. This approach allows for power savings and better performance while maintaining a simpler design. In this article, we will discuss the concept of GALS systems, their advantages and disadvantages, and their applications[1], [2].

GALS Systems

A GALS system is an electronic design methodology that combines asynchronous and synchronous circuitry. The term GALS is used to describe a system where each component of the design operates independently of each other with its own clock, and these clocks can operate at different frequencies. The globally asynchronous aspect of GALS refers to the fact that the clocks are not synchronized with each other, and the locally synchronous aspect refers to the fact that the logic within each component operates synchronously.

The basic architecture of a GALS system consists of several independent blocks that communicate with each other through asynchronous communication channels. Each block is designed to operate synchronously, which means that the internal logic of the block is controlled by a clock signal. However, the clock signals for each block are not synchronized with each other. Instead, each block operates independently, allowing for flexibility and scalability in the system design[3], [4].

Advantages of GALS systems:

1. Power Savings: One of the most significant advantages of GALS systems is their ability to reduce power consumption. Since each block operates independently and at different clock frequencies, only the blocks that are active need to be powered, while the other blocks can be put into a low-power state. This approach results in significant power savings, particularly in large designs where many blocks are present.
2. Scalability: Another advantage of GALS systems is their scalability. The ability to add or remove blocks from the design without affecting the overall system performance is a significant advantage. This approach makes it easy to upgrade the design or add new functionality without having to redesign the entire system.
3. Low Latency: GALS systems can provide low-latency communication between blocks. The asynchronous communication channels used in GALS systems allow for faster data transfer between blocks without the overhead of synchronous clock signals. This approach can be particularly beneficial in applications such as high-speed data transfer and signal processing.
4. Simplicity: GALS systems offer a simpler design approach compared to fully asynchronous designs. The synchronous design within each block makes it easier to design and verify the system. Additionally, the asynchronous communication channels used in GALS systems are simpler to design and verify than synchronous communication channels.

Disadvantages of GALS systems:

1. Increased Design Complexity: Although GALS systems offer a simpler design approach compared to fully asynchronous designs, the overall design complexity can increase due to the need for asynchronous communication channels. The design of these channels requires additional consideration and verification to ensure correct operation.
2. Timing Challenges: The use of multiple clocks in GALS systems can introduce timing challenges. The timing requirements between different clock domains need to be carefully considered to ensure that the system operates correctly. The need for additional circuitry to handle timing challenges can add to the design complexity.
3. Testability: The testability of GALS systems can be challenging. Since each block operates independently, it can be difficult to test the system as a whole. Additional test infrastructure may be required to ensure that the system is functioning correctly.

Applications of GALS systems:

1. Multi-core Processors: GALS systems are commonly used in multi-core processors to reduce power consumption and improve performance. Each core operates independently, and the communication between the cores is handled by asynchronous communication channels.
2. Network-on-Chip: GALS systems are also used in Network-on-Chip (NoC) designs
3. Digital Signal Processing: GALS systems are also used in digital signal processing (DSP) applications, where low-latency communication between different components is critical. The use of asynchronous communication channels can provide faster data transfer and reduce latency, making GALS systems ideal for DSP applications.
4. FPGA Designs: GALS systems are also becoming increasingly popular in Field-Programmable Gate Array (FPGA) designs. FPGAs consist of multiple programmable logic blocks, which can be designed to operate independently and communicate with each other using asynchronous communication channels.
5. Memory Interfaces: GALS systems are also used in memory interface designs, where low-latency communication is essential. GALS-based memory interfaces can provide faster data transfer and lower latency compared to synchronous memory interfaces.

Design Considerations:

When designing a GALS system, several considerations must be taken into account to ensure correct operation.

1. Clock Domain Crossing: The use of multiple clocks in a GALS system can create timing challenges when signals cross clock domains. Special care must be taken to ensure that signals are correctly synchronized and timed to prevent data loss or corruption.
2. Asynchronous Communication: The use of asynchronous communication channels requires additional consideration during the design phase. These channels must be carefully designed and verified to ensure correct operation.
3. Power Management: GALS systems offer significant power savings compared to fully synchronous designs. However, the power management of each block must be carefully considered to ensure that only the necessary blocks are powered on, while the rest are put into a low-power state.
4. Testing: Testing GALS systems can be challenging due to the independent operation of each block. Additional test infrastructure may be required to ensure that the system is functioning correctly.

DISCUSSION

GALS (Globally Asynchronous Locally Synchronous) is a design methodology for digital systems that allows designers to combine globally asynchronous and locally synchronous components to create highly integrated and efficient systems. GALS systems are becoming increasingly popular due to their ability to handle complex and highly dynamic systems, such as those found in modern computing and communication systems.In this article, we will provide an introduction to GALS systems, including an overview of the design methodology, key components, design challenges, and examples of applications.

The GALS design methodology is based on the principle of separating global timing from local timing. In other words, the timing of the system is not controlled by a single global clock signal, but instead each component of the system has its own local clock. This approach allows for greater flexibility and scalability in system design, as it enables designers to add or remove components without affecting the timing of the entire system[5], [6].The GALS design methodology can be divided into two main components: asynchronous interfaces and clock domain crossing (CDC) circuits.

Asynchronous Interfaces

Asynchronous interfaces are used to connect asynchronous components to synchronous components. Asynchronous components operate independently of a clock signal, and their outputs change only when their inputs change. Synchronous components, on the other hand, operate according to a clock signal, and their outputs change only on the rising or falling edge of the clock.

To connect asynchronous and synchronous components, asynchronous interfaces are used to transfer data between the two domains. These interfaces typically use handshaking protocols to ensure that the data is transferred correctly and that both domains are synchronized.

Clock Domain Crossing Circuits

Clock domain crossing (CDC) circuits are used to transfer data between different clock domains. When data is transferred between two components that operate on different clocks, there is a risk of metastability, which can cause the output to be uncertain for a short period of time. To prevent this, CDC circuits are used to synchronize the data between the two domains.

CDC circuits typically use a double latch approach, where the input data is latched on the rising edge of the source clock, and the output data is latched on the rising edge of the destination clock. This approach ensures that the data is stable when it is transferred between the two domains, and reduces the risk of metastability.

Key Components of GALS Systems

The key components of GALS systems are asynchronous components, synchronous components, asynchronous interfaces, and CDC circuits.

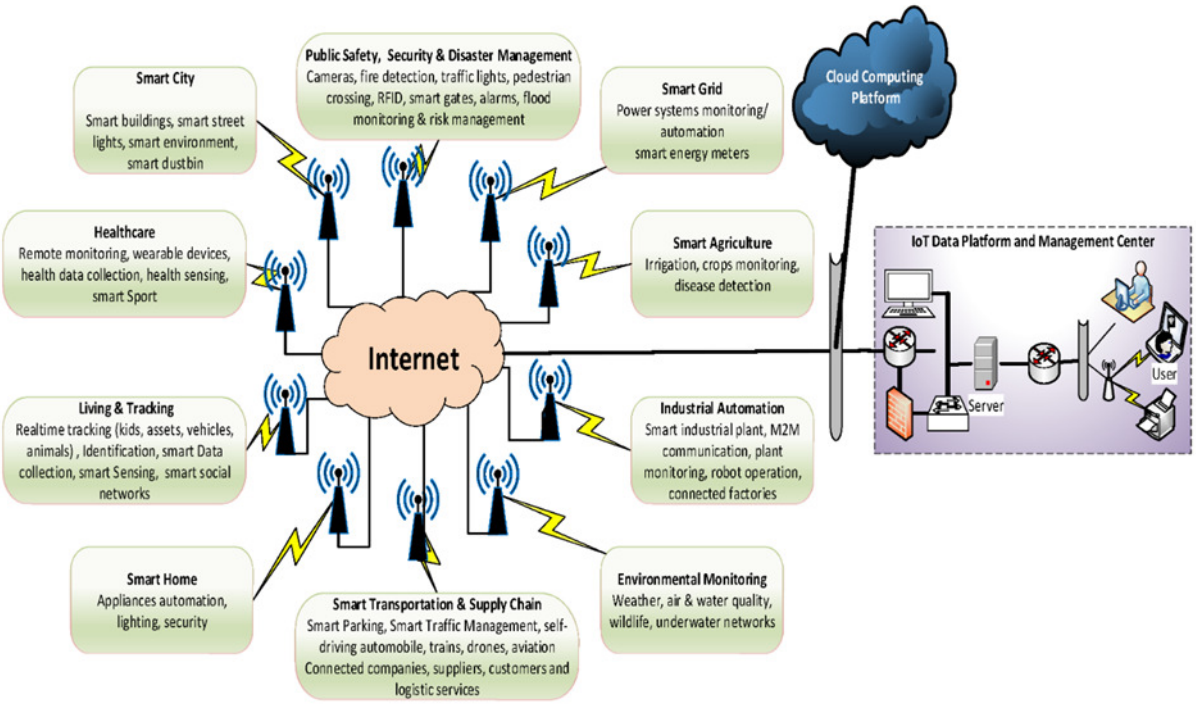


Figure 1: Illustrate the Lora wan Trends.

Asynchronous Components

Asynchronous components are digital circuits that operate independently of a clock signal. These components use data-dependent transitions to change their outputs, and are often used to implement control circuits and communication protocols. Asynchronous components can be designed using a variety of methods, including gate-level designs and high-level synthesis tools.

Synchronous Components

Synchronous components are digital circuits that operate according to a clock signal. These components use clock-dependent transitions to change their outputs, and are often used to implement arithmetic and logic circuits. Synchronous components can be designed using hardware description languages (HDLs) such as Verilog or VHDL[7], [8].

Asynchronous Interfaces

Asynchronous interfaces are used to connect asynchronous components to synchronous components. These interfaces typically use handshaking protocols to ensure that the data is transferred correctly and that both domains are synchronized. Some examples of asynchronous interfaces include dual-rail encodings, bundled-data encodings, and self-timed circuits.

Clock Domain Crossing Circuits

CDC circuits are used to transfer data between different clock domains. These circuits are used to prevent metastability, which can cause the output to be uncertain for a short period of time. CDC circuits typically use a double latch approach, where the input data is latched on

the rising edge of the source clock, and the output data is latched on the rising edgeof the destination clock. CDC circuits can be implemented using a variety of techniques, including synchronizers, FIFOs, and handshake protocols.

Design Challenges in GALS Systems

Designing GALS systems can be challenging due to the need to handle multiple clock domains and asynchronous components. Some of the key design challenges include:

Clock Skew

Clock skew occurs when the clock signal arrives at different components at slightly different times. This can cause timing errors and can be particularly problematic in GALS systems, where different components may have different clock frequencies. To mitigate clock skew, designers often use clock distribution networks that compensate for the delay between the clock source and the destination. Figure 2 illustrate the Globally asynchronous locally synchronous (GALS) architecture

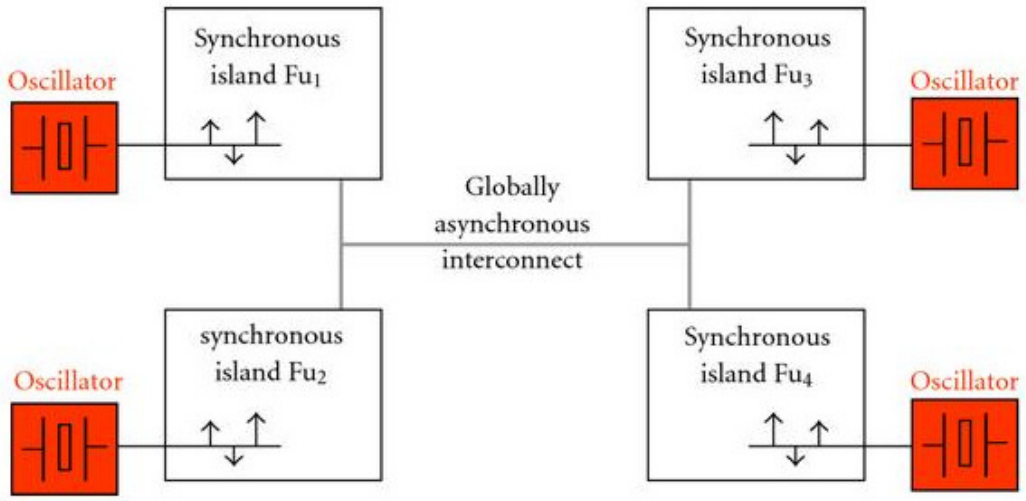


Figure 2: Illustrate the Globally asynchronous locally synchronous (GALS) architecture.

Metastability

Metastability occurs when a digital signal is latched at an intermediate value due to timing uncertainties. This can cause the output of the latch to be uncertain for a short period of time, which can lead to errors in the system. To mitigate metastability, designers use synchronizers and other CDC circuits to ensure that the data is stable before it is transferred between clock domains.

Interface Protocols

Asynchronous interfaces require special protocols to ensure that the data is transferred correctly and that both domains are synchronized. These protocols can be complex and can add significant overhead to the system. To mitigate this, designers often use standard interface protocols such as Handshake and Signal Transition Graphs (STGs).

Examples of GALS Systems

GALS systems are used in a wide range of applications, including computing, communication, and control systems. Here are some examples of GALS systems in action:

High-Performance Computing

High-performance computing (HPC) systems often require large amounts of data to be transferred between different components at high speeds. GALS systems are used in HPC systems to handle this data transfer, as they can provide high bandwidth and low latency without being limited by a single clock domain.

Communication Systems

Communication systems such as networking and wireless systems often require complex synchronization and timing protocols to handle the transfer of data between different components. GALS systems are used in these applications to provide flexibility and scalability, while ensuring that the timing and synchronization requirements are met.

Control Systems

Control systems such as robotics and automation systems often require real-time response and high accuracy. GALS systems are used in these applications to handle the complex control algorithms and timing requirements, while providing the flexibility and scalability required for the system to adapt to changing conditions.

While purchasing data under the framework, buyers are required to pay a particular amount to the provider of the essential asset and the data source information plus Buyers can take part in the information source's crowdfunding initiative, which allows users to invest in top-notch content produced by other users and earn a portion of the revenue share. Information buyers can also share content; this action is tracked by a smart contract on the blockchain, and a profit share can be earned from the content's advertising revenue [9], [10]. Framework administrators agree on a series of agreements to complete the data on the blockchain. System administrators have heavy demands on the computer and communication infrastructure. The primary sources of income for system administrators are transaction fees and consensus incentives.

The distribution and sharing of multimedia content places a heavy burden on the network in terms of storage and bandwidth. Engineering Forwarding charges for streaming media may make as much than 40% of overall operational expenses. In a blockchain-based information distribution system, users with free resources and eco-partners may freely join nodes to provide consumers bandwidth and storage capacity services in exchange for commission incentives. As a result, the ecosystem's running expenses have been drastically lowered.

Since the clicks, downloads, or page views of the material on the blockchain-based information distribution system are publicly available, advertisers may pay for advertising based on the data.Information providers are free to upload, classify, provide a short introduction, and set their own prices for their own work. Customers may utilise the site to look for their favourite authors and pieces of content, explore material by category, read user reviews and content profiles, buy content, and rate and comment on that content.

Taking into account that the blockchain-based data distribution system requires the storage of many images, videos, documents, and other types of data Data storage nodes, system maintenance nodes, and common nodes are the three categories of nodes in the blockchain-based data distribution system. The system's data resources are maintained by +e data storage nodes.

The suitable data resource for storage, as a data storage node of the system, must first have adequate storage capacity, according to the incentive model of the blockchain-based data distribution system. In order to maintain the blockchain ledger of the blockchain-based data distribution system, a system maintenance node is employed. A system maintenance node has to have enough computational power to finish the proof of work in addition to having a particular quantity of storage capacity and being appropriate for the global ledger of storage system. Light nodes may also exist among ordinary nodes. The least is required of the nodes themselves for such nodes. Just the account books and information pertaining to oneself should typically be retained.

Data is a valuable asset in the big data setting because of its inherent or prospective value. In everyday life, common data often passes through many processing steps. Users find it challenging to assess the source and credibility of the information since the intermediary process lacks the essential transparency. Several activities that were previously thought to be impossible to do may be completed via the integration of blockchain technology with other situations and technologies.

To create a scalable economy, digital cities will eventually need to digitise production components and hologram economic operations. By substituting "back-to-back trust" for the original "face-to-face trust" connection, blockchains may lower the cost of exchanges and transactions.

As this is happening, the "chain network" is being utilised to unite the blockchains of various designs and scenarios, achieve the digitization of production elements, and fully document the flow, connection, and equity distribution of social production materials. The identity privacy and data privacy protection features of the blockchain system are restricted. This article offers methods and safeguards to improve user identity privacy by thoroughly analysing the identity privacy and data privacy leakage issues of the blockchain system and combining privacy protection mechanisms and cryptography techniques.

The blockchain system's privacy protection mechanism is improved by the public key of data privacy's ability to search for the data privacy protection scheme. Using the distributed data storage feature of the blockchain system may provide a lot of field application systems new application modes based on the ecological environment of the blockchain system. Currently, a variety of application fields have developed a preliminary accumulation in blockchain technology, gradually integrating the blockchain system's features with the original business system and using the blockchain's unique properties to address the shortcomings of the business system while simultaneously enhancing the blockchain system's own flaws and limitations[11].

The large data privacy protection and scalability difficulties are thoroughly investigated and assessed based on the current blockchain system design, and some research findings are reached, but there are still many works that can be further explored. This study combined the

blockchain with big data. This research suggests a shared data privacy protection and scalability solution in order to address the security assurance and adaptability of huge information sharing. The issue of data storage is somewhat resolved by this technique, but the captured data are still kept on the cloud. The failure of centralization is still a possibility. The next stage in resolving the centralization failure issue will be to broaden the remedy in light of recent study findings.

CONCLUSION

GALS systems provide a powerful design methodology for handling complex and dynamic digital systems. By separating global timing from local timing, GALS systems enable designers to create highly integrated and efficient systems that can handle a wide range of applications. While GALS systems can be challenging to design and implement, they offer significant benefits in terms of performance, flexibility, and scalability. As digital systems continue to become more complex, GALS systems are likely to become even more important in future designs.

REFERENCES:

[1] M. N. Štagoj, A. Comino, and R. Komel, "Fluorescence based assay of GAL system in yeast Saccharomyces cerevisiae," *FEMS Microbiol. Lett.*, 2005, doi: 10.1016/j.femsle.2005.01.041.

[2] V. R. Pannala, S. Bhartiya, and K. V. Venkatesh, "Experimental and steady-state analysis of the GAL regulatory system in Kluyveromyces lactis," *FEBS J.*, 2010, doi: 10.1111/j.1742-4658.2010.07708.x.

[3] N. Van Toan, D. M. Tung, and J. G. Lee, "Measurements of metastability in MUTEX on an FPGA," *IEICE Electron. Express*, 2018, doi: 10.1587/elex.14.20171165.

[4] R. Kamal and J. M. M. Arostegui, "A Multi-Synchronous Bi-Directional NoC (MBiNoC) architecture with dynamic self-reconfigurable channel for the GALS infrastructure," *Alexandria Eng. J.*, 2018, doi: 10.1016/j.aej.2017.02.019.

[5] W. Zha *et al.*, "Reconstruction of the biosynthetic pathway of santalols under control of the gal regulatory system in yeast," *ACS Synth. Biol.*, 2020, doi: 10.1021/acssynbio.9b00479.

[6] S. Mangan, S. Itzkovitz, A. Zaslaver, and U. Alon, "The incoherent feed-forward loop accelerates the response-time of the gal system of Escherichia coli," *J. Mol. Biol.*, 2006, doi: 10.1016/j.jmb.2005.12.003.

[7] F. Jebali, F. Lang, and R. Mateescu, "Formal modelling and verification of GALS systems using GRL and CADP," *Form. Asp. Comput.*, 2016, doi: 10.1007/s00165-016-0373-3.

[8] S. Ryo *et al.*, "Positive Feedback Genetic Circuit Incorporating a Constitutively Active Mutant Gal3 into Yeast GAL Induction System," *ACS Synth. Biol.*, 2017, doi: 10.1021/acssynbio.6b00262.

[9] S. Ostergaard, L. Olsson, M. Johnston, and J. Nielsen, "Increasing galactose consumption by Saccharomyces cerevisiae through metabolic engineering of the GAL gene regulatory network," *Nat. Biotechnol.*, 2000, doi: 10.1038/82400.

[10] F. Moutinho and L. Gomes, "Augmenting high-level petri nets to support GALS distributed embedded systems specification," *IFIP Adv. Inf. Commun. Technol.*, 2013, doi: 10.1007/978-3-642-37291-9_24.

[11] F. Moutinho and L. Gomes, "Asynchronous-channels within petri net-based GALS distributed embedded systems modeling," *IEEE Trans. Ind. Informatics*, 2014, doi: 10.1109/TII.2014.2341933.