CHAPTER

A QUICK OVERVIEW OF ELECTRONIC HARDWARE

2

CONTENTS

2.1 INTRODUCTION

Computing in the 21st century has become pervasive in our daily lives. What was once the domain accessible only to scientists and engineers has now become commonly available in almost every corner of the globe and to every citizen of the world. It is common to see cell phone being used by anyone anywhere in the world even in very remote areas, to see tens of microcontrollers in cars, and that many people carry computers in the form of fitness trackers. In short, computers are everywhere and many people today depend heavily on them for even the most basic tasks in their daily lives, namely shopping, paying bills, checking bank accounts, finding where to eat, etc.

The pervasiveness of modern computing systems is a direct result of constant advancements in integrated circuit (IC) design and fabrication technologies over the past half century. We can trace the history of computing back to inventions such as Charles Babbage's difference engine [1] or the ENIAC of the late 1940s [2]. Modern computer is a direct result of the advent of the transistor, first realized in the form of point-contact transistor fabricated by Bardeen, Shockley, and Brattain at Bell Laboratories in 1947 [3]. The point-contact transistor was relatively bulky and constructed from germanium. Later, other semiconductor materials, notably silicon, were used to realize bipolar transistors and, eventually, field-effect transistors (FETs) in the 1960s [3].

In 1965, Gordon Moore of Intel observed that the number of transistors integrated per square inch doubled every two years [3]. Transistor density, along with switching speeds, indeed continued to double almost every 18–24 months in the next five decades, which made this observation well known in semiconductor industry as "Moore's law". Figure 2.1 illustrates increases in transistor density from the early 1970s to the latest generation of integrated circuits (here, processors) in 2016 [4,5]. Although the continued increase into the early 2000s is quite impressive, it is also worth mentioning that performance increase has already begun to slow down. This is due mainly to the fact that process variations and environmental noises in an integrated circuit making it extremely difficult to achieve the expected performance as a result of technology scaling. Thus, we are now entering an interesting era, where alternative nanoscale technologies are actively pursued for future computing applications as demand continues to remain strong for higher performance and smaller area.

With the slow down of transistor scaling, often pointed to as a sign of the looming end of Moore's law, many researchers have begun to consider "Beyond Moore" or "More than Moore" technology alternatives [5]. One underlying goal of these investigations into new nanoelectronic devices is to find technologies that allow continuation of the performance scaling that has been enjoyed over the past several decades. However, many novel nanotechnologies are often not more robust than today's silicon-based complementary metal-oxide-semiconductor (CMOS) transistors. These emerging devices are often found to offer new opportunities for novel applications or to provide performance improvements through hybrid integration with CMOS [6–8]. Thus, "More than Moore" advocates would argue that nanotechnologies should be considered for the novel applications they are likely to enable, as opposed to simply enhancing the performance of existing computing systems and architectures. Nonetheless, the landscape of integrated circuit design and fabrication technologies are beginning to change right at the time that computing has become commonplace, especially with the advent of Internet of things (IoTs) and smart devices used everywhere.

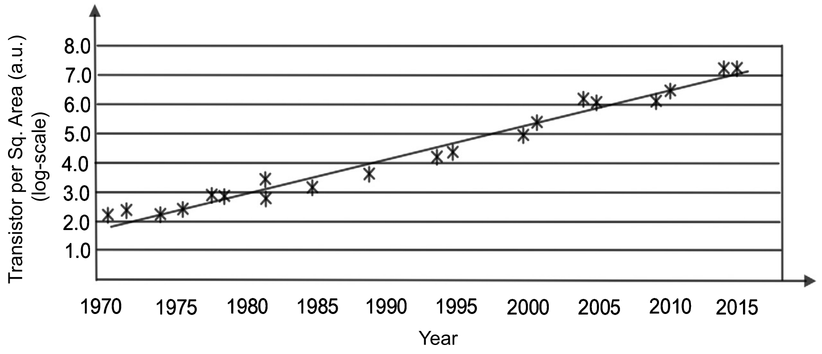

FIGURE 2.1

Illustration of Moore's law, showing a biannual doubling of the number of transistors per unit area.

Nanotechnology is most often defined as the field of study associated with the synthesis and integration of structures with feature sizes less than 100 nm [9]. Considering the fact that gate lengths for silicon CMOS transistors have been below 100 nm (now at 7 nm) for more than a decade now, one would argue that modern electronics has already been dominated by nanotechnology. To differentiate conventional CMOS from non-CMOS nanotechnologies, terms such as "nanoscale CMOS" and "deep submicron" are often used. That being said, several novel forms of semiconductor transistor technology have emerged that are certainly worth considering in the context of "Beyond Moore" nanoelectronics.

2.2 NANOSCALE TECHNOLOGIES

2.2.1 SILICON-ON-INSULATOR

Silicon-on-insulator (SOI) technology has emerged in recent years as a way to improve the performance of semiconductor devices. Specifically, SOI refers to a fabrication technique, where a semiconductor, typically silicon, is layered on top of an insulator, typically silicon dioxide. Since the top semiconductor layer can be very thin, it becomes possible to implement doped diffusion regions that extend all the way through to the insulator underneath. Further, some SOI transistors fall into the category of fully depleted, meaning when the device is on, the entire channel is inverted. In short, the SOI structure leads to a reduced parasitic capacitance and other nonideal effects, such that the performance is drastically improved relative to conventional, non-SOI approaches.

Manufacturing SOI-based devices and circuits can be challenging, depending on how the top semiconductor layer is fabricated on top of the insulator. Ideally, the top semiconductor would be grown via epitaxial techniques, such that the resulting layer is very thin. However, the crystallinity of the oxide/insulator layer typically does not match that of the desired semiconductor, meaning epitaxial growth leads to nonideal behavior. A more common technique for SOI is the use of a thick wafer of the same material as the top semiconductor that is flipped and essentially glued onto the insulator. Since the top semiconductor is manufactured independent of the insulator, it will have the necessary crystallinity required for the desired electronic devices and circuits. However, the process of flipping, glueing, and thinning that top layer tends to be quite expensive.

2.2.2 FinFET TECHNOLOGY

Another approach to yielding nanoscale field effect transistor (FET) is to go vertical. At the device level, going vertical can refer to the advent of FinFET transistors, meaning transistors whose semiconductor channels are fabricated vertically as fin structures. The so-called fin allows a gate to be wrapped around the channel from three sides, leaving only the bottom of the fin/channel open to the underlying bulk substrate. As is the case with SOI, the wrapped gate leads to a more fully depleted channel, hence reduced parasitic effects. The reduction in parasitic capacitance and other nonideal characteristics leads to improved performance. Further, since the fins are fabricated from an existing semiconductor substrate, as opposed to layered on top of an insulator as in the case of SOI, FinFET technology does not suffer from the same manufacturing challenges common to SOI. It is worth mentioning that FinFET technology has become the common approach for sub-32 nm CMOS technology, with companies such as Intel, Samsung, TSMC, and Global Foundries all offering technology nodes at 14 nm and, soon, even at 10 and 7 nm gate lengths [10].

2.2.3 3D INTEGRATED CIRCUITS

As feature sizes shrink to 10 nm and, possibly, even smaller to 7 nm and 5 nm in the near future, it is believed that CMOS transistor technology has been scaled down about as far as possible in terms of lateral dimensions. In order to continue gaining density improvements in modern semiconductor electronics, vertical dimensions must be better utilized. This is the primary objective for development of three-dimensional (3D) integrated circuit technology, known as 3DIC. 3DICs refer to layered approach to manufacturing, where multiple semiconductor substrates (fabricated in same or different foundries) are stacked on top of one another in order to implement circuits vertically as well as laterally. There are several approaches to build a 3DIC, including face-to-face, front-to-back, and SOI-based approaches.

The face-to-face approach to 3DIC is perhaps the simplest because no additional structures need to be implemented in silicon. Instead, the top metal layers of two die or wafers include contact points or landing pads for connecting the two layers together. One layer is then flipped and oriented on top of the other so that connections are made at the predefined contact points. Thus, the resulting 3DIC consists of two semiconductor layers oriented in a face-to-face arrangement. One challenge with the face-to-face approach arises when constructing a 3DIC with more than two layers. In this case, either off-chip connections are needed to connect to other pairs, or a second form of 3DIC is required that utilizes through silicon vias (TSVs) to connect layers oriented in a back-to-back structure.

Many 3D implementations are constructed in some form of a back-to-front arrangement, where each semiconductor layer is oriented with metal layers to the top. In this case, connections across layers require the use of TSVs. Each TSV tends to be larger in the cross-sectional area relative to conventional vias, limiting the number of total TSVs one could integrate onto a single die. However, such 3DIC technologies enable drastic reductions in total wire lengths, thereby reducing delay, improving performance. Further, the ability to stack transistors vertically enables a form of scaling where the number of transistors per unit area continues to rise with an increase in the number of layers. Such integration gives hope to continue meeting Moore's law requirement for both performance and area.

2.2.4 BULK-SILICON TECHNOLOGY

Bulk silicon CMOS continues to be the major workhorse in modern electronics. Although beyond-CMOS nanotechnologies have been emerging, CMOS devices still continue to play a significant role due to technological maturity, cost, performance, and ease of integration. This has led to hybrid CMOS-nanoelectronic approaches, where CMOS is used for functions such as I/O and gain, whereas a nanoscale technology is used for dense memory and/or logic implementations [6]. One major advantage for using nanotechnology is the increased density and ability to squeeze functionality into regular crossbar structures. Further, nanoelectronic materials are continually being explored as extreme low-power alternatives to their CMOS counterparts. This is particularly important for 3D-based architectures, where heat across upper layers becomes a major concern [11,12]. It is believed that CMOS continues to have its place in future ICs and electronic computing systems, emerging systems and application domains such as digital microfluidics, IoT, quantum computers, and neuromorphic computing. Thus, the future for ICs consist of a mixed bag of technologies, including many new devices constructed from emerging nanoscale materials.

2.3 DIGITAL LOGIC

Digital logic is the representation of signals and sequences of a digital circuit using numbers. It is the fundamental concept, underlying behind all modern computing systems, that provides an understanding on how hardware and circuit communicates within a device. This section introduces the basic concept of digital logic. Specifically, we introduce binary logic, combinational circuit, and sequential circuit, such as flip-flops, registers, and memories [13].

2.3.1 BINARY LOGIC

Binary logic or boolean logic is the core concept of boolean algebra that forms “Gates” which all digital electronic circuits and microprocessor based systems are constructed of. Basic digital logic gates perform logical operations of AND, OR, and NOT on binary numbers.

Information is stored in computer systems in binary form. A binary bit represents one of the two possible states, which are generally referred to as logic “1” and logic “0”. Specifically, the presence of a positive voltage can be represented as logic “1”, high, or true; the absence of a voltage can be represented as logic “0”, low, or false. In Boolean Algebra and truth table, these two states are represented as “1” and “0”, respectively [14]. Figure 2.2 shows a CMOS circuit, which typically consists of a p-type transistor and an n-type transistor. In digital logic, each transistor is either on or off, which represents a short circuit or an open circuit, respectively. As illustrated in Fig. 2.2, the left side provides logic “true” in binary form, whereas the right side provides logic “false” in binary form.

2.3.2 DIGITAL LOGIC GATES

Digital logic gate is the fundamental building block of digital circuits. There are a number of basic logic gates, which perform logic operations indicated by their names on binary numbers (see Fig. 2.3). As an example, the two-input logic gates have the following features:

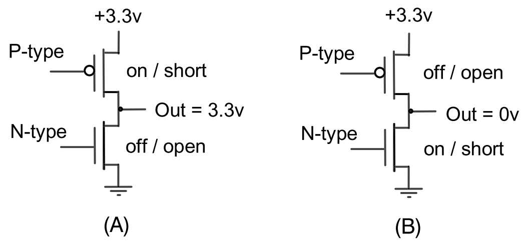

FIGURE 2.2

A binary bit is true (A) vs. false (B).

- AND gate: the output is 1 if all inputs are 1; otherwise, the output is 0.
- OR gate: the output is 1 if at least one input is 1; otherwise, the output is 0.
- XOR gate: the output is 0 if both inputs are same; otherwise, the output is 1.
- NAND gate: the output is 1 if at lease one input is 0; otherwise, the output is 0.
- NOR gate: the output is 1 if both inputs are 0; otherwise, the output is 0.
- NOT gate or inverter: the output is 1 if the input is 0 and the output is 0 if the input is 1.

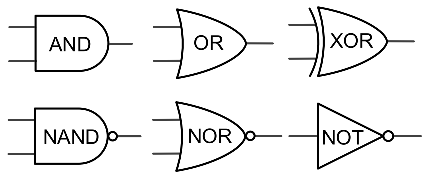

FIGURE 2.3

Basic two-input logic gates.

2.3.3 BOOLEAN ALGEBRA

Boolean Algebra is the mathematical representation for digital logic. The mathematical formats for the above basic logic operations are shown below.

- A AND B is written as AB or $A \cdot B$;
- A OR B is written as $A + B$;
- A XOR B is written as $A \oplus B$;
- NOT A is written as ∼A or A' or $\overline{A}$;
- A NAND B is written as $(AB)'$, $(A \cdot B)'$, or $\overline{(AB)}$;
- A NOR B is written as $(A + B)'$ or $\overline{(A + B)}$.

The laws of Boolean Algebra are listed in Table 2.1, where A, B, and C can be considered as Booleans or individual bits of a logic operation [14].

Table 2.1 Laws of Boolean Algebra [14]

$A\&B = B\&A$	Commutative Law
$A\|B = B\|A$	Commutative Law
$(A\&B)\&C = A\&(B\&C)$	Associative Law
$(A\|B)\|C = A\|(B\|C)$	Associative Law
$(A\|B)\&C = (A\&C)\|(B\&C)$	Distributive Law
$(A\&B)\|C = (A\|C)\&(B\|C)$	Distributive Law
$A\&0 = 0$	Identity of 0
$A\|0 = A$	Identity of 0
$A\&1 = A$	Identity of 1
$A\|1 = 1$	Identity of 1
$A\|A = A$	Property of OR
$A\|(\sim A) = 1$	Property of OR
$A\&A = A$	Property of AND
$A\&(\sim A) = 0$	Property of AND
$\sim(\sim A) = A$	Inverse
$\sim(A\|B) = (\sim A)\&(\sim B)$	De Morgan's Theorem
$\sim(A\&B) = (\sim A)\|(\sim B)$	De Morgan's Theorem

2.3.4 SEQUENTIAL CIRCUIT

Modern digital logic circuits can be divided into two main parts, combinational logic and sequential logic. Combinational logic changes after signal propagation delay when input changes, and its output only relies on its present input. In contrast, sequential logic has at least one clock signal, and consists of blocks of combinational logic divided by memory elements which are driven by clock signals. Therefore, the output of sequential logic depends on both the present and past inputs.

2.3.4.1 Sequential Circuit Elements

Sequential circuit elements (flip-flops and latches) are commonly used for storage of information. To be exact, a flip-flop is used to store a single binary bit and has two states; one of its two states represents "1", the other represents "0". Such data storage is used to store state, and the corresponding circuit is referred to as sequential logic. A flip-flop is clocked, that is, synchronous or edge-triggered, whereas a latch is level-sensitive. We briefly review different types of flip-flops here.

D-Type Flip-Flop

A D flip-flop is widely used as the basic building block of random access memory (RAM) and registers. The D flip-flop captures the D-input value at the specified edge (i.e., rising or falling) of the clock. After the rising/falling clock edge, the captured value is available at Q output. The truth table of D flip-flop is shown in Table 2.2.

Table 2.2 Truth table of D-type flip-flop

Clock	D	Q_{next}
Rising edge	0	0
Rising edge	1	1
Non-rising	X	Q

T-Type Flip-Flop

For a T-Type Flip-Flop, if T-input is high, the output toggles when the clock input is high. If T-input is low, the output remains the same. Hence, T flip-flop can be used for clock division. The truth table of T flip-flop is shown in Table 2.3.

Table 2.3 Truth table of T-type flip-flop

T	Q	Q_{next}	Comment
0	0	0	Hold state (no clk)
0	1	1	Hold state (no clk)
1	0	1	Toggle
1	1	0	Toggle

JK-Type Flip-Flop

The JK flip-flop has two inputs (J and K), and the output can be set as different values based on the inputs. The truth table of JK-type flip-flop is shown in Table 2.4.

Table 2.4 Truth table of JK-type flip-flop

J	K	Q_{next}	Comment
0	0	Q	Hold state
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}$	Toggle

2.3.4.2 Timing Parameters

Setup time, hold time, and propagation delay are three important parameters when designing a sequential circuit. These three timing parameters are briefly explained in this section and illustrated in Fig. 2.4.

Setup Time

Setup time (t_{su}) is the minimum amount of time that the data input is required to be stable before the rising/falling edge of the clock, so that the data can be correctly sampled by the clock.

Hold Time

Hold time (t_h) is the minimum amount of time that the data input is required to be stable after the rising/falling edge of the clock, so that the data can be correctly sampled by the clock.

Propagation Delay

Clock-to-output delay (t_{CO})/propagation delay (t_P) is the time that a flip-flop takes to change its output after the rising/falling edge of the clock.

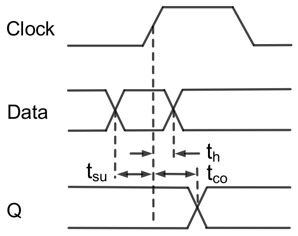

FIGURE 2.4

Timing parameters: setup time, hold time, and clock-to-output delay of a flip-flop.

2.4 CIRCUIT THEORY

A circuit is a network consisting of circuit elements and wires. To be specific, wires are typically designated as straight lines on a schematic, and nodes are locations, where wires connect. All other symbols on a schematic are circuit elements. Resistors, capacitors, and inductors, the three most passive linear circuit elements, which make up electronic circuits, are briefly reviewed in this section.

2.4.1 RESISTORS AND RESISTANCE

As common elements of electronic circuits, resistors are passive two-terminal components that implement electrical resistance. Resistors are typically used in circuits to reduce current flow, adjust signal levels, divide voltages, and bias active elements. There are different types of resistors, including high-power resistors, fixed resistors, and variable resistors, which are used in various applications. The typical schematic diagram of resistors is shown in Fig. 2.5A; the resistor symbol on the right is the International Electrotechnical Commission (IEC) resistor symbol.

Electrical resistance, the quantitative property of a resistor, is defined as

$$\gamma = \frac{\rho L}{A}, \tag{2.1}$$

where ρ is the resistivity of the material, L is the length of the resistor, and A is the cross-section area of the resistor.

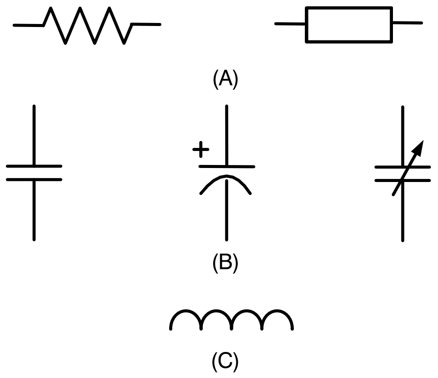

FIGURE 2.5

Typical schematic diagrams of resistor (A), fixed, polarized, and variable capacitors (B), and inductor (C).

2.4.2 CAPACITORS AND CAPACITANCE

Capacitors are passive two-terminal electrical components, which store potential energy in an electric field. They are characterized by capacitance. Capacitors are widely used in different applications. In electronic circuits, they are used to block direct current (DC) while allowing alternating current (AC) to pass. In analog filter networks, they are used to smooth the output of power supplies. In resonant circuits, they are used to tune radios to the specified frequencies. The typical schematic diagram of three types of capacitors is shown in Fig. 2.5B.

Capacitance is defined as the ratio of the electric (positive/negative) charge Q on each conductor to the voltage V between them, which is shown below in Eq. (2.2). The unit of capacitance is the farad (F), defined as one coulomb per volt (1 C/V). Typical values of capacitors in general electronics range from 1 femtofarad (pF $= 10^{-15}$ F) to 1 millifarad (mF $= 10^{-3}$ F).

$$C = \frac{Q}{V}, \tag{2.2}$$

where Q is the positive or negative charge on each conductor, and V is the voltage between them.

In practical uses, charge sometimes affects the capacitor mechanically, hence changing its capacitance. Therefore, capacitance can be calculated as

$$C = \frac{dQ}{dV}. \tag{2.3}$$

2.4.3 INDUCTORS AND INDUCTANCE

Inductors are passive two-terminal electrical components, which store energy in a magnetic field when current flows through it [15]. Typically, an inductor is composed of an insulated wire into a coil around a core. An inductor is characterized by its inductance. Inductors are widely used in AC electronic equipment. In electronic circuits, they are used to block AC while allowing DC to pass. In electronic filters, they are used to separate signals of different frequencies. In addition, along with capacitors, they are used to make tuned circuits for tuning radio and TV receivers. The typical schematic diagram of an inductor is shown in Fig. 2.5C.

Inductance is defined as the ratio of the voltage to the rate of change of current, which is shown below. The unit of inductance is henry (H), and typical values for inductors range from 1 millihenry (mH $= 10^{-3}$ H) to 1 microhenry (μH $= 10^{-6}$ H).

$$L = \frac{\Phi}{I}, \tag{2.4}$$

where Φ is the total amount of magnetic flux through a circuit, which is generated by current I, and depends on the circuit geometric shape.

2.4.4 KIRCHHOFF'S CIRCUIT LAWS

Kirchhoff's circuit laws are linear constraints on the branch voltages and node currents in the lumped element model of electrical circuits. Kirchhoff's circuit laws include Kirchhoff's current law (KCL) and Kirchhoff's voltage law (KVL), which are independent of the nature of the electrical elements [16].

2.4.4.1 Kirchhoff's Current Law

Kirchhoff's current law addresses the conservation of charge entering and leaving a circuit node. As one of the fundamental laws used for circuit analysis, it states that the sum of current flowing into a circuit's node is exactly equal to the sum of current flowing out the same node since it has no other place to go as no charge is lost [15,16].

In other words, the sum of currents meeting at a circuit's node is equal to zero. Also, since current can be seen as a signed quantity, this law is expressed as

$$\sum_{k=1}^{N} I_k = 0. \tag{2.5}$$

This principle is illustrated in Fig. 2.6. It can be seen that the current entering into the node is equal to the current leaving that node, that is, $i_1 + i_2 = i_3 + i_4$. In other words, the sum of the currents entering and leaving the same node is equal to zero; $i_1 + i_2 - (i_3 + i_4) = 0$.

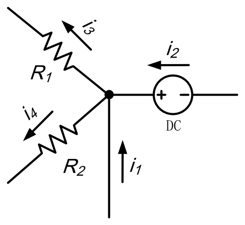

FIGURE 2.6

Kirchhoff's current law.

2.4.4.2 Kirchhoff's Voltage Law

Kirchhoff's voltage law addresses the conservation of energy around a closed circuit loop. It states that the sum of branch voltages around a closed circuit path is equal to zero [15,16].

Since voltage can be seen as a signed (that is, positive or negative) quantity reflecting the polarities and signs of the sources, and voltage drops around the loop, this law can be expressed as

$$\sum_{k=1}^{N} V_k = 0. \tag{2.6}$$

This principle is illustrated in Fig. 2.7. It can be seen that the sum of branch voltages around the loop is equal to zero, that is, $V_1 + V_2 + V_3 + V_4 = 0$.

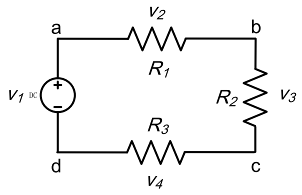

FIGURE 2.7

Kirchhoff's voltage law.

2.5 ASICs AND FPGAs

Application-specific integrated circuits (ASICs) and field programmable gate arrays (FPGAs) are integrated circuits, which serve different ends on the spectrum of applications for modern ICs. Due to their own design philosophy and features, their differences include non-recurring engineering (NRE), cost, flexibility, and performance [17].

2.5.1 ASICs

As the name indicates, an ASIC is an integrated circuit customized and created for a particular purpose rather than for general-purpose use. ASICs are used to implement analog, digital, as well as mixed-signal functionalities in high volume and high performance. Nowadays, the functionality of digital ASICs is generally described using a hardware description language (HDL), for example, Verilog and VHDL. Circuit diagrams were previously used to describe the functionality, but their use has dwindled over the past two decades, as size of the circuit continued to increase.

2.5.2 FPGAs

As the name implies, an FPGA is an integrated circuit, which is designed to be configured by customers after manufacturing; hence, it is field programmable. Similar to an ASIC, customers of an FPGA typically use HDL, such as verilog or VHDL, to specify the configuration of an FPGA.

An FPGA consists of a set of programmable logic blocks and a hierarchy of reconfigurable interconnects. Logic blocks are wired through the reconfigurable interconnects to be configured for different functions. In modern FPGAs, logic blocks include memory elements, such as simple flip-flops or complete memory blocks. Examples of FPGAs are shown in Fig. 2.8, where the left one is a Stratix IV FPGA developed by Altera, and the right one is a Spartan FPGA developed by Xilinx.

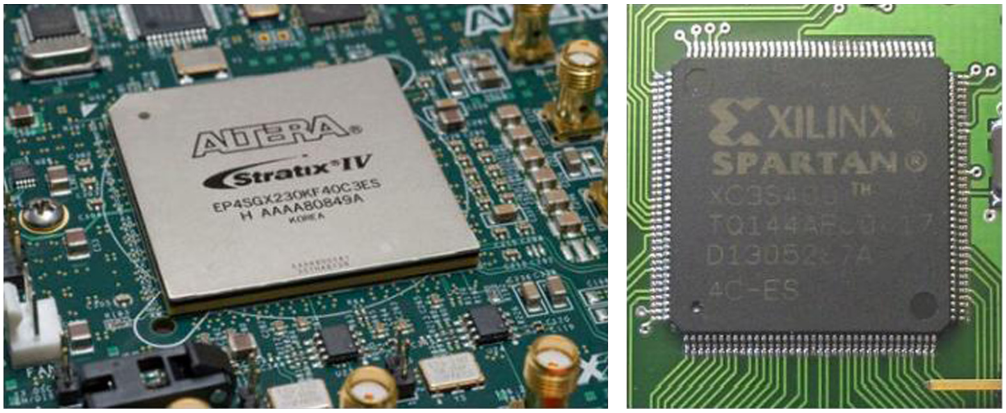

FIGURE 2.8

FPGAs from Altera (left) and Xilinx (right).

2.5.3 DIFFERENCE BETWEEN ASICs AND FPGAs

Since ASICs are semi- or full-custom designs, they require higher development costs and often reach into the millions during design and implementation stages. In addition, ASICs are non-reprogrammable once they are produced; hence, changes in the design incur additional cost. Although ASICs have a relatively higher nonrecurring cost, it is justified due to the following facts – (i) ASICs often have higher density and can integrate complex functionalities into a chip, thus providing limited size, low power, as well as low cost designs; (ii) due to its custom feature, the number of transistors is considered very carefully, and minimal resources would be wasted in an ASIC design; (iii) when making large quantities of designs for a specific use, ASICs would be the optimal choice.

FPGAs advantage lies in their flexibility, ability to be reprogrammed in the field, and cost-effectiveness. For example, the reprogrammable nature allows designers and manufacturers to change the design or to send out patches even after products are sold. Customers often utilize this feature to create their prototypes based on FPGAs, so that their designs can be fully debugged, tested, and updated in the real scenario before manufacturing. Although the nonrecurring cost is very limited and, therefore, time to market is fast, some resources on an FPGA are wasted since the package and resources for a specific type of FPGA are standard.

Moreover, when analyzing the production cost in relation to the production volume, using FPGAs becomes costly compared to ASICs as the volume increases. Also, since FPGAs cannot be fully customized, some specific analog blocks have to be added into FPGA platforms. Those functionalities typically require to be implemented by external ICs, thereby further increasing the size and the cost of the final product. The difference between ASICs and FPGAs is summarized in Table 2.5 [17].

Table 2.5 The difference between ASICs and FPGAs

	ASIC	FPGA
Time to market	Slow	Fast
NRE	High	Low
Design flow	Complex	Simple
Power consumption	Low	High
Performance	High	Medium
Unit size	Low	Medium
Unit cost	Low	High

Since modern designs are often cost-constrained, the cost comparison between ASICs and FPGAs is further illustrated in Fig. 2.9. It can be observed that FPGAs are cheaper than ASICs when building low-volume production circuits. However, ASICs become more cost-effective after the volume of 400K units (note that this number is subject to change as technology further scales). In other words, for lower-volume designs, FPGA is capable of reducing costs significantly, whereas ASICs are more efficient and cost-effective on high-volume productions [17].

2.6 PRINTED CIRCUIT BOARD

A printed circuit board (PCB) is a thin board made of laminate materials, such as fiberglass and composite epoxy. Conductive pathways are etched/printed on a board to electrically connect a variety of components on the board, for example, transistors, resistors, and integrated circuits (IC) [18]. In other words, a PCB is developed to mechanically support and electrically connect electronic components through conductive tracks and pads. Components are typically soldered onto the PCB to be mechanically and electrically connected to it. Figure 2.10 shows the picture of a PCB, which was built by the authors of this book with a purpose of hardware hacking. The figure includes conductive traces, vias, and electronic components.

PCBs are widely used in various applications, such as desktop computers, laptop computers, mobile devices, TVs, radios, IoTs, Automotive, digital cameras, and more. They serve as the foundation for many computer components, including graphic card, sound card, adapter card, and expansion card. All these components are further connected to a PCB, that is, the motherboard. While PCBs are universally used in computers, mobile devices, and electrical appliances, it should be noted that PCBs used in mobile devices are typically thinner and contain finer circuits than the ones used in other applications [18].

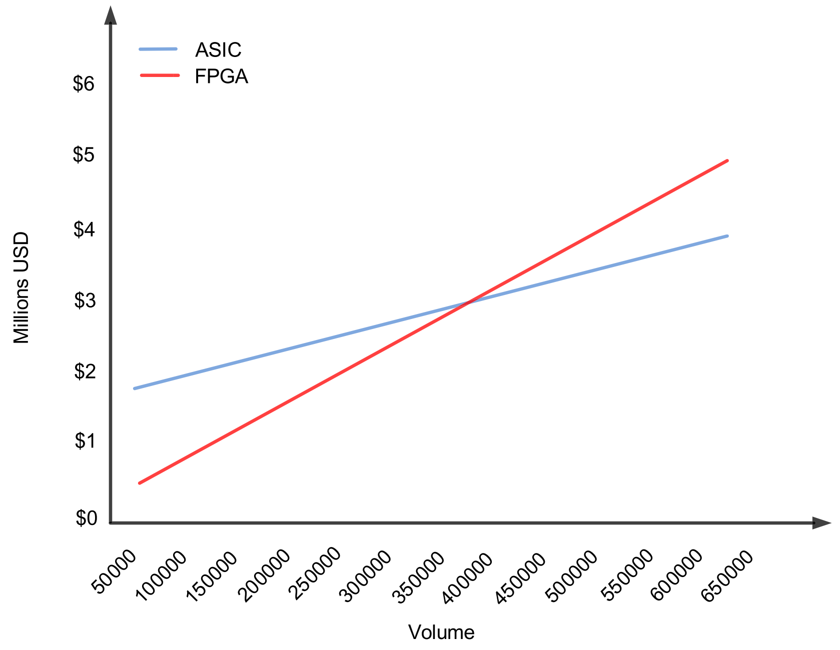

FIGURE 2.9

Total cost of ASIC vs. FPGA for different production volume.

2.6.1 CLASSIFICATION OF PCB

Depending upon requirement, PCBs can be single-sided bond (SSB, that is, one copper layer), double-sided bond (DSB, that is, two copper layers on both sides of one substrate layer), or multi-layer bond (MLB, that is, inner and outer layers of copper, alternating with layers of substrate).

SSB PCB

Single-sided bond PCB has only one side copper layer, the other side is insulated material. Hence, only one-sided copper can be used to manufacture the device as copper is a conductive material.

DSB PCB

Double-sided bond PCB has three layers; two of them are side copper layers. Both ends are coated with copper material, and the middle part is insulating material. Hence, both ends can be used for design, manufacturing, and electronic components placement.

ML PCB

Multi-layer bond PCB has more than two copper layers, where copper is placed in different layers as required. ML PCBs allow for much higher component density, since circuit traces on the inner layers would otherwise take up surface space between components. Nowadays, ML PCBs are mostly used

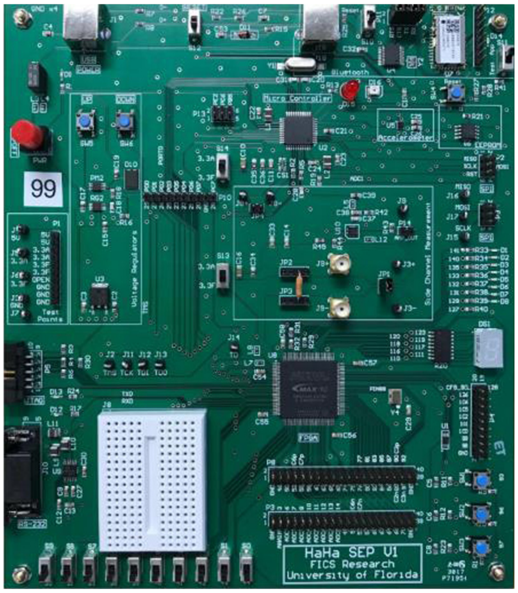

FIGURE 2.10

A sample printed circuit board.

in different applications. However, ML PCBs make analysis, repair, and circuits in-field modification much more difficult.

2.6.2 PCB DESIGN FLOW

The PCB design flow consists of four stages, that is, part selection, schematic capture and simulation, board layout, and board verification and validation [19]. The PCB design flow is shown in Fig. 2.11.

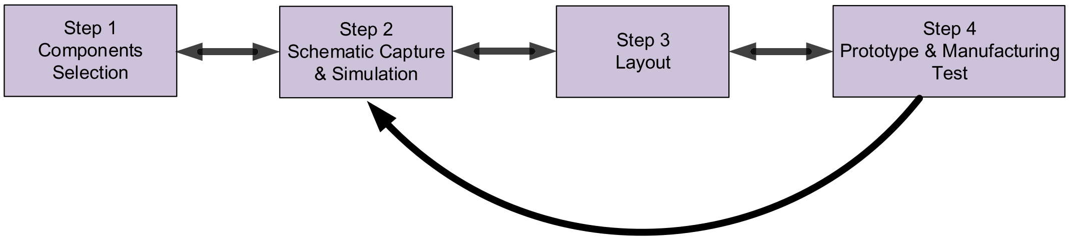
Step 1
Components Selection
Step 2
Schematic Capture & Simulation
Step 3
Layout
Step 4
Prototype & Manufacturing Test

FIGURE 2.11

The PCB design flow.

Part Selection

Components (for example, transistors, resistors, operational amplifiers, and digital components) are the most fundamental part of a design. Part selection stage evaluates and investigates how components coordinate with each other and work as a part of the overall design. Generally, information on physical components is available online, such as datasheets, which provides operating points, switching characteristics, and design considerations.

Schematic Capture and Simulation

As the fundamental stage of PCB design process, capture is a design interface in which graphical symbols of components are placed and connected to build the design topology. Once a schematic is captured, typically SPICE simulation is utilized to predict circuit behavior, to check the integrity of circuit designs, and to analyze the effects of components and signals upon the design. Simulation is able to quickly identify the majority of bugs and errors before the design is physically manufactured, thus greatly reducing both time-to-market and production cost.

Board Layout

Upon capturing the schematic and simulating the design, the physical prototype is built to test the design performance under real workload conditions. Layout is done through EDA tools and in a CAD environment, in which the symbols for components that represented the design in capture stage are seen in the format of the actual component physical dimensions. The final design form in this stage is exported to a Gerber format, which can be used by PCB manufacturers to turn into a physical representation of the board. Although the advanced EDA tools are able to automatically place and route a board, the critical elements and components have to be manually taken care of by experienced engineers with extra scrutiny to ensure the performance and stability of the design.

Prototype Test and Manufacturing Test

Prototype test and manufacturing test are the final steps in the validation of a PCB. Whereas prototype test validates if the design meets the target specifications, manufacturing test at the high-volume production is performed to ensure each device being shipped meets the testing principles and expected responses. If bugs or errors are discovered during the simulation stage are identified at this stage, iterations through the design flow have to be made to address the problem.

2.6.3 CREATING A PCB DESIGN

Typically, a PCB consists of multiple copper layers that are used to conduct electrical signals and various dielectric layers that are used for insulation. The green color found on most PCBs comes from a solder mask. Solder masks also come in other colors, such as blue or red. The fundamental components of a PCB are introduced below [19].

Board Outline

The board outline of a PCB is often cut into a specific shape for a form factor that meets a specific design. When working with devices of small size, the need for a specific shape (for example, round, rectangular, and zig-zag) is important to finalizing a product. Hence, a number of methods used to define the shape of the board outline, such as importing DXF files (a format used by mechanical CAD tools) to define a specific shape for the design.

Creating Copper Routes

Copper routes on a PCB board are used to conduct electrical signals to various components and connectors on the board. The copper pathways are created through layering copper on the board surface(s) and etching away excess copper. Etchings are created by placing a mask over regions of copper pathways and removing all unwanted copper.

Drilling Holes

Drilling holes on a PCB board is required to create signal pathways to different layers on a board, or create areas to attach components on a board. A plated-through hole (PTH) in a board is named as a via that provides electrical connection between a copper route on one layer to a copper route on another layer. Holes for vias are typically created/drilled using a fine drill bit, whereas holes for small micro-vias are created/drilled by way of a laser. There are several types of vias. For instance, a via starting on one outer layer and ending at an inner layer is called a blind via, which does not completely pass through a board. A via connecting copper routes on two inner layers of a board is called a buried via, which does not connect at the surface level of a board.

Components on a PCB

Components on a PCB refer to the semiconductor devices, such as through-hole technology (THT) components and surface-mount devices (SMD). THT parts are often larger with longer pins, which are inserted into drilled holes and soldered one-by-one onto a board. In contrast, SMD parts are often much smaller and allow you to solder much smaller leads to the board surface. Therefore, parts can be attached to the board top/bottom surface instead of having to solder through-hole parts.

Gerber Files

A Gerber file refers to a file format used for PCB manufacturing. Gerber files are utilized by fabrication machines to layout electrical connections, such as trace and pads. The file generally contains necessary information for drilling and milling the circuit board.

2.7 EMBEDDED SYSTEMS

As its name suggests, an embedded system is a microprocessor- or microcontroller-based system, which is designed for a specific function and embedded into a larger mechanical or electrical system. Since embedded systems are developed for some specific task rather than to be a general-purpose system for multiple tasks, they are typically of limited size, low power, and low cost. Embedded systems are widely used in various purposes, such as commercial, industrial, and military applications.

Typically, an embedded system consists of hardware and application software components. Some embedded systems have real-time operating system (RTOS). Some small embedded systems may not have RTOS. Therefore, an embedded system can be defined as a microprocessor- or microcontroller-based, software driven, reliable, and real-time control system. Figure 2.12 shows an embedded system on a plug-in card with multiple components such as processor, memory, power supply, and external interfaces.

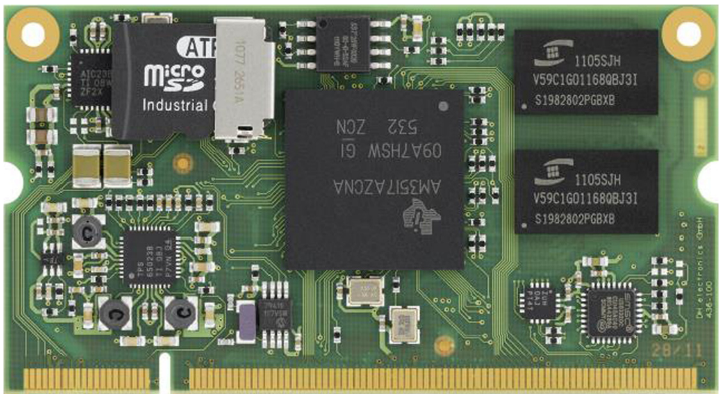

FIGURE 2.12

An embedded system on a plug-in card.

2.7.1 EMBEDDED SYSTEM HARDWARE

An embedded system contains a microprocessor or microcontroller that is typically designed to perform computation for real-time operations. Generally, a microprocessor is only a central processing unit (CPU). Hence, other components (for example, memories, communication interfaces) need to be integrated and work with the microprocessor as a whole system. In contrast, a microcontroller is a self-contained system, which includes a CPU, memories (e.g., RAM, flash memory), and peripherals (e.g., serial communication ports).

2.7.2 EMBEDDED SYSTEM SOFTWARE

Microprocessors or microcontrollers used in embedded systems are generally not as advanced when compared to general-purpose processors designed for managing multiple tasks. They often work on a simple, less-memory-intensive program environment [20]. As a result, embedded system software has specific hardware requirements and capabilities. It is tailored to the particular hardware and has time and memory constraints [21]. Programs and operating systems are generally stored in flash memory within embedded systems.

In like manner, the operating systems or language platforms are developed for embedded use, particularly where RTOS is required. Currently, simple versions of Linux operating system or other operating systems, such as Embedded Java and Windows IoT are generally adopted [20].

2.7.3 CHARACTERISTICS OF AN EMBEDDED SYSTEM

The characteristics of an embedded system can be summarized as presented below.

- Specific function: An embedded system is usually designed for a specific function.
- Tightly constrained: An embedded system is tightly resource- and time-constrained. For example, an embedded system has to be fast and tasks-tolerant of slight variations in reaction time (in real-time or near real-time manner), with limited memory and minimum power consumption.
- Real-time and reactive: Real-time or near real-time manner has to be served in many environments. For instance, a global positioning system (GPS) navigator needs to continually provide road and location information, and to send driver alerts to increase situation awareness in a near real-time manner or sometimes real-time manner. Likewise, a car cruise controller is required to continually monitor and react to speed and brake sensors, and also compute the acceleration or deacceleration in a real-time manner. Any delay would make the car out of control, which could give rise to catastrophic results.
- Hardware/Software codesign: An embedded system is typically a computer hardware system with software embedded in it. Hardware is designed for performance and security, while software is designed for more features and flexibility.
- Microprocessor-/Microcontroller-based: A microprocessor or microcontroller is often deployed at the heart of the embedded system and designed to perform operations.
- Memory: A memory is required for an embedded system since programs and operating systems are generally loaded and stored in the memory.
- Connected peripherals: Peripherals are needed to connect input and output devices.

2.8 HARDWARE-FIRMWARE-SOFTWARE INTERACTION

Hardware refers to the physical components of a system, such as the memory, hard disk drive, graphic card, sound card, central processing unit, motherboard, monitor, adapter card, and ethernet cable. Software refers to the instructions or the programs running on hardware, which direct a computer to perform specific tasks or operations, in contrast to hardware upon which the system is built. Computer software is the information processed by systems, for example, data, programs, and libraries. For example, software could be operating systems (OS). OS provides overall control for hardware system and applications, which are programs designed for a specific task. Software is installed and resides on the hard disk and is loaded into memory when it is needed.

Although hardware and software are independent concepts, they require each other to function and neither can be realistically used on its own. Figure 2.13 shows how users interact with application software running on the computer system. It can be observed that the application software interacts with the operating system, which in turn communicates with the hardware. Information flow is indicated by the arrows.

Specifically, most algorithms can be implemented in either hardware or software. Generally, hardware-based algorithm implementation is much faster than software-based, but it can only perform a limited number of instructions, such as additions, comparisons, moves, and copies. Hence, software is utilized to create complex algorithms based on these basic instructions. The software that directly controls hardware is machine language. Software could also be written in low-level assembly language, which is strongly corresponding to machine language instructions, and translated into machine language through the assembler. However, many instructions are required to create even the elementary algorithms since machine languages are too simple. Hence, the majority of software is written

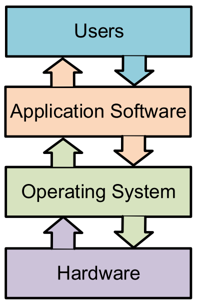

FIGURE 2.13

The diagram of application software, operating system, and hardware.

with high-level programming languages, which are much easier and more efficient for programmers to use, describe, and develop algorithms, since they are much closer than machine languages to natural languages. Then, high-level languages are translated into machine languages using a compiler and an interpreter [22]. The interaction between different software levels and hardware is illustrated in Fig. 2.14.

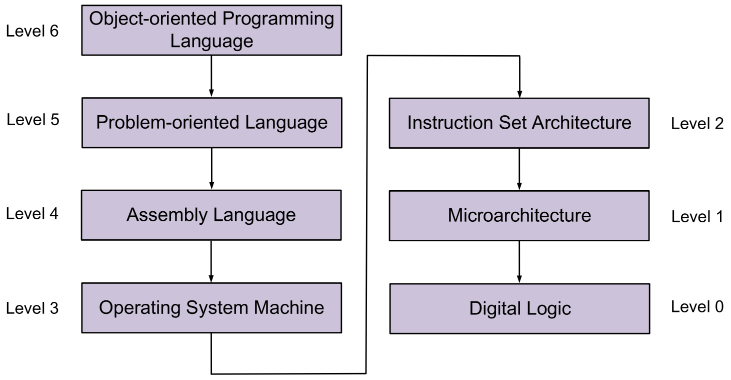

FIGURE 2.14

Multi-level computer systems.

Level 0 is hardware level. Programs in Levels 1, 2, and 3 consist of a series of numbers, which are hard for users to understand and interpret. Level 4 is the assembly language, which is a bit more user-friendly. The instructions at this level become readable and meaningful to users. Level 5 and 6 refer to the majority of software development. For instance, at Level 5, standard programming languages are generally used for development, such as C and C++. At Level 6, object-oriented programming languages become available, such as Java, Python, and .NET.

Firmware refers to a specific class of software that provides the low-level control for the specific hardware in a device. For instance, firmware can provide a standardized operating environment for the device's complex software, or act as the device's operating system that performs control, monitoring, and data manipulation functions. Firmware, such as the basic input-output system (BIOS) of computers, typically contains the basic functions of a device and provides services to higher-level software. Except the simplest, all electronic devices, such as computer systems, computer peripherals, embedded systems, consumer appliances, and Internet-of-thing (IoT) devices, contain firmware. It is stored in nonvolatile memories including ROM, EPROM, and flash memory, and is rarely or never changed after manufacture in contrast to the software. It can only be updated with special installation processes or with administration tools.Therefore, Firmware can be viewed as an intermediate form between hardware and software or a specific class of software embedded in hardware.

Sometimes both software and firmware are needed to be upgraded to correct errors or bugs, add features, or improve the device performance. For example, beta software or beta firmware is an intermediate version, which has not been thoroughly tested. Beta version is far more likely to have bugs than the polished final version since generally bugs or errors can only be manifested by putting the system in the real world.

2.9 EXERCISES

2.9.1 TRUE/FALSE QUESTIONS

1. Hold time is the minimum amount of time that the data input is required to be stable before the rising/falling edge of the clock.
2. Even though ASIC and FPGA have some similar features and are widely used in a variety of applications, they cannot replace each other.
3. Kirchhoff's current law addresses the conservation of charge entering and leaving a circuit node, while Kirchhoff's voltage law addresses the conservation of energy around a closed circuit loop.
4. Firmware can be considered as a specific class of hardware.
5. Resistance can be calculated through the length and the cross-section area of the resistor.

2.9.2 SHORT-ANSWER TYPE QUESTIONS

1. Explain the motivation for developing 3D integrated circuits.
2. Explain the timing constraints in an integrated circuit.
3. Considering a cylindrical resistor of radius 6.0 mm and length 2.0 cm, if the resistivity of the resistor material is $1.8 \times 10^{-6}\ \Omega$, calculate the resistance.
4. Explain the similarity between Kirchhoff's Current Law and Kirchhoff's Voltage Law.
5. Explain the difference between ASICs and FPGAs.
6. Explain the interaction between users, application software, and hardware.

2.9.3 LONG-ANSWER TYPE QUESTIONS

1. Find the three currents (I_1, I_2, I_3) and voltages (V_{ab}, V_{bc}, V_{bd}) in the circuit in Fig. 2.15.
2. Describe the typical PCB design flow.
3. Briefly summarize the features of a typical embedded system.
4. Explain the difference and similarity between software and firmware.

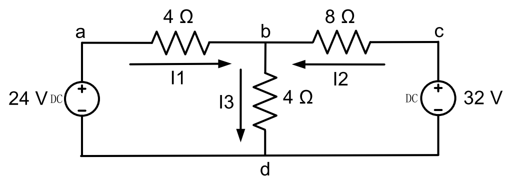

FIGURE 2.15

The circuit for Problem 1.

REFERENCES

[1] D. Harris, S. Harris, Digital Design and Computer Architecture, 2nd edition, Morgan Kaufmann, 2012.

[2] W. Stallings, Computer Organization and Architecture: Designing for Performance, 7th edition, Pearson Education India, 2005.

[3] N.H. Weste, D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th edition, Pearson Education India, 2010.

[4] G.E. Moore, Cramming more components onto integrated circuits, Electronics Magazine 38 (8) (1965) 114–117.

[5] M.M. Waldrop, More than Moore, Nature 530 (2016) 144–148.

[6] M.M. Ziegler, M.R. Stan, A case for CMOS/nano co-design, in: Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design, ACM, pp. 348–352.

[7] K.K. Likharev, D.B. Strukov, CMOL: devices, circuits, and architectures, in: Introducing Molecular Electronics, Springer, 2006, pp. 447–477.

[8] G.S. Rose, Y. Yao, J.M. Tour, A.C. Cabe, N. Gergel-Hackett, N. Majumdar, J.C. Bean, L.R. Harriott, M.R. Stan, Designing CMOS/molecular memories while considering device parameter variations, ACM Journal on Emerging Technologies in Computing Systems (JETC) 3 (2007) 3.

[9] V. Parihar, R. Singh, K. Poole, Silicon nanoelectronics: 100 nm barriers and potential solutions, in: Advanced Semiconductor Manufacturing Conference and Workshop, 1998. 1998 IEEE/SEMI, IEEE, pp. 427–433.

[10] T. Song, H. Kim, W. Rim, Y. Kim, S. Park, C. Park, M. Hong, G. Yang, J. Do, J. Lim, et al., 12.2 A 7nm FinFET SRAM macro using EUV lithography for peripheral repair analysis, in: Solid-State Circuits Conference (ISSCC), 2017 IEEE International, IEEE, pp. 208–209.

[11] J.H. Lau, T.G. Yue, Thermal management of 3D IC integration with TSV (through silicon via), in: Electronic Components and Technology Conference, 2009. ECTC 2009. 59th, IEEE, pp. 635–640.

[12] K. Tu, Reliability challenges in 3D IC packaging technology, Microelectronics Reliability 51 (2011) 517–523.

[13] M.M. Mano, Digital Logic and Computer Design, Pearson Education India, 2017.

[14] Embedded Systems: Introduction to ARM CORTEX-M Microcontrollers, Volume 1, ISBN 978-1477508992, 2014, http://users.ece.utexas.edu/~valvano/.

[15] C. Alexander, M. Sadiku, Fundamentals of Electric Circuits, 3rd edition, 2006.

[16] J.W. Nilsson, Electric Circuits, Pearson Education India, 2008.

[17] FPGA vs ASIC, what to choose?, anysilicon, https://anysilicon.com/fpga-vs-asic-choose/, Jan. 2016.

[18] R.S. Khandpur, Printed Circuit Boards: Design, Fabrication, Assembly and Testing, Tata McGraw-Hill Education, 2005.

[19] Best practices in printed circuit board design, http://www.ni.com/tutorial/6894/en/#toc6, Aug. 2017.

[20] Embedded system, https://internetofthingsagenda.techtarget.com/definition/embedded-system, Dec. 2016.

[21] E.A. Lee, Embedded Software, Advances in Computers, vol. 56, Elsevier, 2002, pp. 55–95.

[22] A.S. Tanenbaum, Structured Computer Organization, 5th edition, Pearson Prentice Hall, 2006.