# ARCHITECTURE AND CAD FOR NANOSCALE AND 3D FPGA

BY

CHEN DONG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Assistant Professor Deming Chen, Chair
Assistant Professor Eric Pop
Professor Naresh R. Shanbhag
Professor Martin D. F. Wong

# ABSTRACT

FPGAs (field programmable gate arrays) are attractive alternatives compared to ASICs (application-specific integrated circuits) for significantly lowering amortized manufacturing costs and dramatically improving design productivity. The architecture of an FPGA is very regular. It is relatively easy to design a highly optimized tile, with consideration of various manufacturing related issues, and then to replicate it many times across the chip. The configurability of FPGAs also enables yield improvement and defect tolerance. However, FPGAs are still facing serious challenges in terms of delay, power consumption, and logic density compared to ASICs. FPGA is estimated to be over twenty times less efficient in logic density, over three times worse in delay, and over ten times higher in power consumption compared to a functionally equivalent ASIC.

One promising way to improve FPGA performance is to incorporate three-dimensional (3D) integration, which increases the number of active layers and optimizes the interconnect network vertically. Another solution is to apply novel nanoelectronic materials (nanomaterials) and devices. This dissertation introduces three novel reconfigurable architectures, named 3D nFPGA, FPCNA (field programmable carbon nanotube array), and NEM FPGA (nanoelectromechanical FPGA), which utilize 3D integration techniques and new nanoscale materials synergistically. Customized CAD flows that consider process variation have been developed for different architectures to evaluate their potential performances. Also described is a 3D variation aware routing flow, which is an essential tool for future 3D FPGA architecture exploration.

3D nFPGA is based on CMOS (complementary metal-oxide-semiconductor) and nano hybrid techniques that incorporate nanomaterials such as nanowire crossbars and carbon nanotube bundles into the CMOS fabrication process. Using unique features of FPGAs and a novel 3D stacking method enabled by the application of nanomaterials, 3D nFPGA obtains a 4× footprint reduction comparing to the traditional CMOS-based 2D FPGAs. The performance and

power of 3D nFPGA driven by the 20 largest MCNC (microelectronics center of North Carolina) benchmarks have been evaluated. Results demonstrate that 3D nFPGA is able to provide a performance gain of 2.6× with a small power overhead compared to the traditional 2D FPGA.

FPCNA includes lookup tables created entirely from continuous carbon nanotube (CNT) ribbons. To determine the performance of the building blocks, variation aware physical design tools are used, with statistical static timing analysis (SSTA) that can handle both Gaussian and non-Gaussian random variables. A 2.75× performance improvement is seen over an equivalent CMOS FPGA at a 95% yield. In addition, FPCNA offers a 5× footprint reduction compared to a baseline FPGA.

3D NEM FPGA is the architecture that utilizes nanoelectromechanical (NEM) relays and 3D integration techniques synergistically. This proposed architecture has unique features including a hybrid CMOS-NEM FPGA lookup table (LUT) and configurable logic block (CLB), NEM-based switch block (SB) and connection block (CB), and face-to-face 3D stacking. This architecture also has a built-in feature called direct link, which takes advantage of the short vertical wire length provided by 3D stacking to further enhance performance. An overall 46.3% critical path delay reduction has been observed compared to its CMOS counterpart.

To maximize the potential performance gain of 3D integrated circuit architectures, an SSTA engine was developed to deal with both uncorrelated and correlated variations in 3D FPGAs. The effects of intra-die and inter-die variation are considered. Using the 3D physical design tool TPR as a base, a new 3D routing algorithm is developed, which improves the average performance of two-layer designs by over 22% and three-layer designs by over 27%.

*To Father and Mother*

# ACKNOWLEDGMENTS

It has been a great opportunity for me to attend University of Illinois at Urbana-Champaign to pursue my Ph.D. degree. The ECE department at the University of Illinois has had a shining reputation in the engineering world throughout its long history. During my four years study here, I have been greatly inspired by many alumni who have made remarkable contributions to modern society and also enriched by interactions with professors, colleagues, and friends.

Foremost, I would like to thank my adviser Professor Deming Chen for the continuous support of my Ph.D. study and research, for his advice, motivation, and enthusiasm. His guidance helped me during the research and writing of this dissertation. Besides my adviser, I would like to thank the rest of my thesis committee: Professor Eric Pop, Professor Naresh Shanbhag, and Professor Martin Wong for their encouragement and insightful comments.

I would like to thank my colleagues, Scott Chilstedt, Lu Wan, Greg Lucas, Christine Chen, Chi-chen Peng, and Alex Papakonstantinou. Collaboration and discussion with them have improved the quality of this work tremendously.

I am deeply grateful to my girlfriend, Lu Bai, whose love and understanding have been great supports for me throughout my years at the University of Illinois. Without her encouragement and assistance in life, I would not have finished this dissertation. Finally, I would like to give my deepest gratitude and love to my parents for their dedication and the many years of support during my study that provided the foundation for this work.

# TABLE OF CONTENTS


Chapter 1 Introduction ..... 1

Chapter 2 Background ..... 4
2.1 Carbon Nanomaterials ..... 4
2.2 Carbon Nanotube FETs (CNFETs) ..... 7
2.3 CNT Logic ..... 12
2.4 NRAM ..... 13
2.5 CNT Bundle Interconnect ..... 14
2.6 Nano-Switches for Routing ..... 16
2.7 Island-Style Baseline FPGA ..... 18

Chapter 3 3D nFPGA: A 3D CMOS/Nano Hybrid Reconfigurable Architecture ..... 20
3.1 Existing Works ..... 20
3.2 CMOS-Nano 3D nFPGA ..... 22
3.3 3D nFPGA Characterization and Evaluation ..... 31
3.4 Experimental Results ..... 38

Chapter 4 FPCNA：Carbon Nanotube-Based Programmable Architecture ..... 43
4.1 FPCNA Architecture ..... 43
4.2 Nanotube Lookup Table Fabrication ..... 50
4.3 Circuit Characterization ..... 52
4.4 CAD Flow ..... 56
4.5 Experimental Results ..... 60

Chapter 5 Variation Aware Routing for Three-Dimensional FPGAs ..... 69
5.1 Related Work ..... 70
5.2 3D FPGA Architecture ..... 72
5.3 Variation Modeling ..... 74
5.4 CAD Tools ..... 77
5.5 Experimental Results ..... 81

Chapter 6 3D Nanoelectromechanical Relay-Based Reconfigurable Architecture ..... 84
6.1 NEM Devices ..... 85
6.2 NEM-Based FPGA Tiles ..... 86
6.3 3D NEM FPGA Architecture ..... 90
6.4 CAD Flow ..... 95

6.5 Experimental Results ..... 100

Chapter 7 Conclusion ..... 103

References ..... 105

# CHAPTER 1
# INTRODUCTION

Historically, CMOS scaling has provided the means to realize higher performance with every technology node, as predicted by Moore's law. Ever since the 90 nm node, the gate length of MOSFETs (metal–oxide–semiconductor field-effect transistors) has entered the nano regime. Nowadays, the 45 nm technology has become the mainstream since 2008, and 32 nm technology was announced in 2009.

As CMOS continues to scale deeper into the nanoscale, quantum physical effects cause the IV characteristics to be substantially different from well-studied MOSFETs. Ballistic mobility and saturation velocity play an important role in limiting MOSFET performance. Degraded drain-induced barrier lowering (DIBL) increases the off-state current ($I_{off}$). Decreased $T_{ox}$ provides better channel control but comes with a penalty of increased gate leakage current ($I_{gate}$). In the meantime, the traditional design and fabrication approach needs to be modified to cope with various non-idealities such as increased process variation and optical lithography difficulties. It becomes more and more difficult to further improve device and circuit performance by reducing the physical gate length.

The Overall Roadmap Technology Characteristics (ORTC) published by the International Technology Roadmap for Semiconductors (ITRS) gives a detailed summary of the key parameters for future technology nodes [1]. The scaling trend of final physical gate length was set at a two-year cycle (0.5×/4 years; 0.71×/2 years) from 1999/90 nm through the 2005/32 nm point, and then the scaling trend slows down to a three-year cycle (0.5×/6 years; 0.71×/3 years) through the end of the roadmap due to the aforementioned scaling difficulties. A good question to ask is: Is there a way to extend the silicon roadmap?

One promising way to continue Moore's law is to incorporate 3D integration [2]-[4], which increases the number of active layers and optimizes the interconnect network vertically. The main advantage of 3D IC technology is that it significantly enhances interconnect resources and increases logic density. If used correctly, 3D IC provides improved bandwidth and throughput, as well as reduced wire length. For $N_{layers}$ stacking, in the best scenario, if the inter-layer vias are ignored, average wire length would be expected to drop by a factor of $(N_{layers})^{1/2}$. Both wire resistance and wire (RC) delay would drop by a factor of $(N_{layers})$. Hence, for interconnect-dominated architectures such as FPGAs, a significant reduction in chip delay and energy can be expected.

Another promising long-term solution is the use of nanoelectronic materials (nanomaterials) and devices. Carbon-based devices and interconnects have shown strong promise. The Emerging Research Devices and Emerging Research Materials working groups of the ITRS have selected carbon-based nanoelectronics as their recommended "Beyond CMOS" technology [1]. A metallic single-walled carbon nanotube (SWCNT) has a mean free path of several micrometers. Within this length, ballistic transport is observed in SWCNT [5]-[7]. Thus, its resistance is a constant without scattering effects. A rope or bundle of SWCNTs conduct current in parallel and significantly reduce the resistance value [8]-[10]. Thus, the SWCNT bundle interconnect can outperform copper interconnect for propagation delay, especially for intermediate and long interconnects [9]-[10]. In addition, SWCNT bundle vias offer high performance and high thermal conductivity. This thermal property of SWCNT bundles is specifically useful for 3D ICs to combat thermal penalty. Large bundles of SWCNTs can be used as thermal vias to directly connect to the heat sink and efficiently dissipate the excessive heat [10]-[11].

Semiconducting SWCNTs have been actively explored to construct carbon nanotube field effect transistors (CNFETs) [12]-[17]. One of the promises of SWCNTs for transistors is the high carrier mobility [18]-[20] because electrical transport in nanotubes can be ballistic. Therefore, CNFETs are promising candidates as extensions to CMOS due to excellent CV/I

device performance. It has been reported that a single CNFET device can be 13× and 6× faster than pMOS and nMOS with a similar gate length based on the intrinsic CV/I gate delay metric when local interconnect capacitances and CNT imperfections are not considered [21].

Instead of completely replacing the CMOS technology, future chips with nanotechnology can be built as a hybrid using both CMOS and nanomaterials, thus taking advantage of both mature CMOS technology and novel advances in nanotechnology [22]-[24]. In the meantime, 3D integration will definitely be a viable solution for ultimate logic density.

However, exiting fabrication techniques of nanodevices offer less control over individual device location. Integrating and interfacing nano components and CMOS components will be another challenge. Taking these limitations into consideration, tile-based structures such as FPGA become the preferred platform in which single tiles can be optimized and replicated many times across the chip. In addition, the programmability of FPGAs allows reconfiguration around the large number of fabrication defects inherent in nanoscale processes, which helps to provide the high level of fault tolerance needed for correct nanocircuit operation. Besides the architecture exploration, a complete 3D-nano-centric CAD flow is essential in designing such hybrid architecture. With this design automation flow, performance, power, and logic density can be evaluated considering issues such as process variation and reliability.

This dissertation presents research conducted since 2005. The following chapters are arranged as follows. Chapter 2 provides related background knowledge including carbon-based devices, logic circuits, interconnects and memory, molecular programmable switches, and baseline CMOS island FPGA architecture. In Chapter 3, a novel reconfigurable architecture, named 3D nFPGA, which utilizes 3D integration techniques and new nanoscale materials synergistically, is introduced. Chapter 4 presents a CNT-based FPGA solution called FPCNA (field programmable carbon nanotube array). A 3D FPGA variation aware routing flow is discussed in Chapter 5. Chapter 6 discusses nanoelectromechanical relay (NEM) based 3D FPGA architecture and related 3D CAD, and Chapter 7 concludes this dissertation.

# CHAPTER 2
# BACKGROUND

## 2.1 Carbon Nanomaterials

Carbon nanomaterials have received significant interest and investment from the research community due to their unique electrical and physical characteristics. This chapter explores the structure and properties of these devices and shows why they can offer such high performance.

### 2.1.1 Atomic Composition

Carbon nanomaterials are composed primarily of benzene-like hexagonal rings of carbon atoms. Each edge of the hexagon is composed of a single or double carbon-carbon bond with a bond length of roughly 0.14 nm. These bonded carbon rings can be connected together in a number of ways that exhibit different properties. If the rings reside on a single plane in a repeating honeycomb-like structure, it makes crystalline monolayer graphene. If the hexagonal graphitic pattern is rolled to form a cylindrical tube, it forms the allotropes known as carbon nanotubes. If sheets of graphene are stacked on top of one another, they form bilayer graphene, trilayer graphene, multi-layer graphene (4+ layers), and eventually graphite (10+ layers). The relationships between these allotropes can be seen in Figure 2.1 [25].

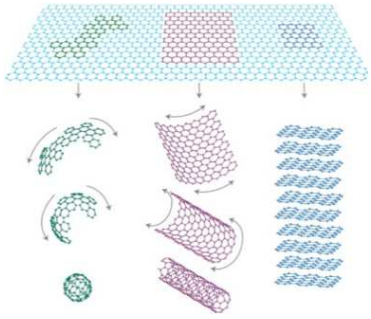


Figure 2.1 Relationships between graphene, buckminsterfullerenes, carbon nanotubes, and graphite

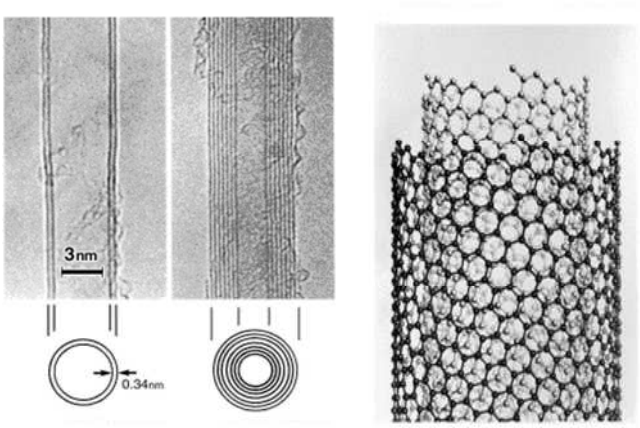



Figure 2.2 Multi-walled carbon nanotubes discovered in 1991

Carbon nanotubes can be categorized into two groups: single-walled carbon nanotubes (SWCNTs) and multi-walled carbon nanotubes (MWCNTs) (Figure 2.2) [26]. SWCNTs are hollow cylinders with a diameter of roughly 1 nm, and can be thought of as a rolled sheet of monolayer graphene. MWCNTs are composed of a number of SWCNTs nested inside one another in order of diameter and can be thought of as a rolled sheet of multi-layer graphene. MWCNTs have dimensions greater than SWCNTs and are typically several tens of nanometers in diameter. Carbon nanotubes vary in length and have been produced in lengths of up to 1 mm. With diameters of less than 10 nm, this allows for exceptionally high aspect ratios, making nanotubes an essentially one-dimensional material.

Due to cylinder symmetry, there is a discrete set of directions that a graphene sheet can be rolled in to form a SWCNT. To characterize each direction, two atoms in the graphene sheet are chosen, one of which serves the role as the origin. The sheet is rolled until the two atoms coincide. The vector pointing from the first atom towards the other is called the chiral vector and its length is equal to the circumference of the nanotube [27]. The direction of the nanotube axis is perpendicular to the chiral vector.

A given SWCNT can be characterized by its chiral vector ($n$, $m$) or, in other words, the direction that the graphene sheet has been rolled. A SWCNT with a chiral vector ($n$, $m$) indicates that during rolling, the carbon atom at the origin is superimposed with the carbon atom at the lattice location ($n$, $m$). Figure 2.3 (a) illustrates possible chiral vectors. Depending on the rolling

method, three different types of SWCNT can be synthesized: the armchair nanotube with $m = n$, the zigzag nanotube with $m = 0$, and chiral nanotubes with $n \neq m \neq 0$ (Figure 2.3(b)-(d)). More interestingly, if $n = m$, the SWCNT is metallic; if $n-m$ is a multiple of 3, then the SWCNT is semiconducting with a very small band gap; otherwise, the SWCNT is a moderate semiconductor. Thus all armchair ($n = m$) nanotubes are metallic. MWCNTs are not characterized in this way because they are almost always composed of nanotubes with varying chirality.

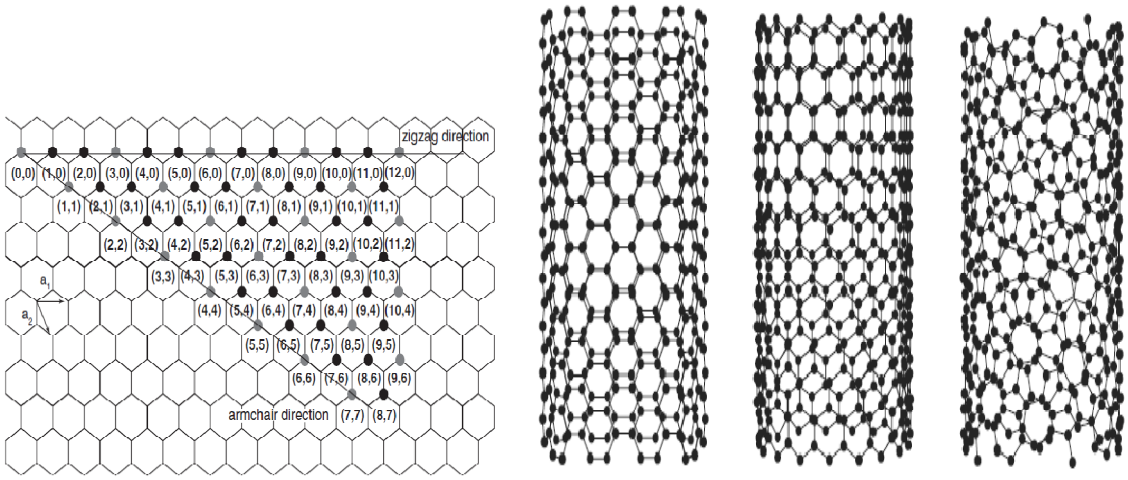



Figure 2.3 (a) Chiral vectors of SWCNTs; (b) armchair SWCNT; (c) zigzag SWCN; and (d) chiral SWCNT

### 2.1.2 Electrical Properties

While advances in silicon technology will continue for the foreseeable future, a highly scaled classical MOSFET will face significant problems in terms of reduced drive current and increased short channel effects, such as drain-induced barrier lowering (DIBL). Carbon nanomaterials have unique electrical properties that allow them to overcome these challenges and achieve strong performance at the sub-10 nm scale.

The high quality of the crystal lattice in carbon nanomaterials gives them a great mean free path, on the order of micrometers, which results in near-ballistic transport of charge carriers. More importantly, this mean free path is achieved at room temperatures, allowing for very high mobilities. Under ideal conditions, room temperature electron mobility can reach about

100,000 $cm^2$/ V·s in carbon nanotubes, and about 200,000 $cm^2$/ V·s in graphene, making them significantly more attractive than silicon at 1,400 $cm^2$/ V·s, and comparable to undoped InSb at 77,000 $cm^2$/ V·s [28].

In addition to their high mobilities, carbon nanomaterials are more robust to short channel effects. Structures such as double gated graphene and all-around gated SWCNTs offer nearly ideal control of the carbon channel electrostatics, minimizing effects such as DIBL. These properties have a direct impact on power. With larger mobilities and longer mean free paths, carbon nanomaterial channels consume less power and dissipate less heat than their silicon counterparts. Much like the switch from bipolar transistors to silicon CMOS, this would allow for a greater number of devices to be integrated for a given power density.

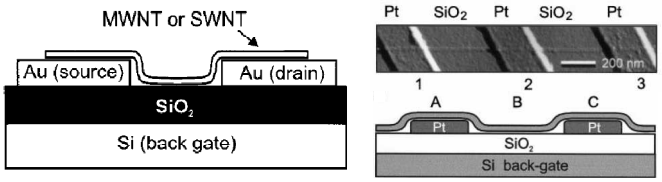
## 2.2 Carbon Nanotube FETs (CNFETs)

As mentioned before, single-walled carbon nanotube with $m-n \neq 3 \times integer$ are categorized into semiconducting nanotubes. The conductance of semiconducting nanotubes strongly depends on gate bias. More importantly, due to the nanoscale dimensions, semiconducting nanotubes demonstrate ballistic electronic conduction and insensitivity to electromigration. The aforementioned advantages make carbon nanotube transistors promising candidates for future building blocks of nano electronics. In the past decade, many works have concentrated on fabrication, modeling, and integration of carbon nanotube transistors. This section will discuss some of the most representative device structure and their modes.

### 2.2.1 Transistor Types

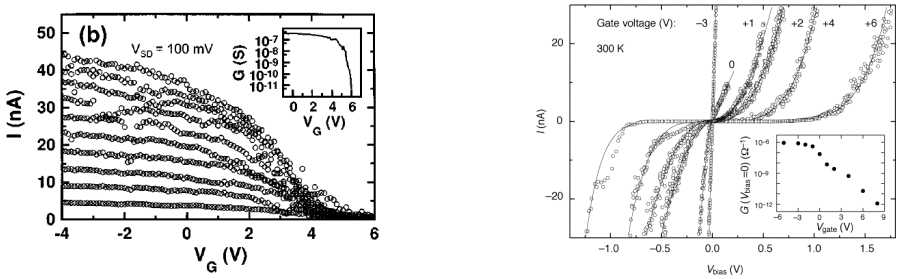
The first reported room-temperature operation of CNT field effect transistors (CNFETs), were from IBM [29] and Delft University of Technology [30] in 1998. The structures of these two CNFETs are shown in Figure 2.4 [29]-[30]. These two designs have very similar architectures: a single nanotube (either single-walled or multi-walled) behaves as the channel region and connecting source-drain electrodes. The $IV_G$ transfer characteristic of the CNFETs

developed in [29]-[30] is shown in Figure 2.5 for different source-drain voltages. As gate voltage swept from +6 V to −4 V, the source-drain current increases strongly, which indicates the device is operated as a FET. The increment of current at negative gate voltages is evidence that the holes carry most of the current. This behavior is identical to that of a p-channel MOSFET. The saturated current value corresponds to a resistance of approximately 1.1 MΩ, which is mainly contributed by metal CNT contact. A conductance difference of five orders of magnitude has been observed.




(a) (b)

Figure 2.4 (a) Schematic cross section of Si back gated CNFET with Au S/D contact (a) and Pt S/D contact (b)

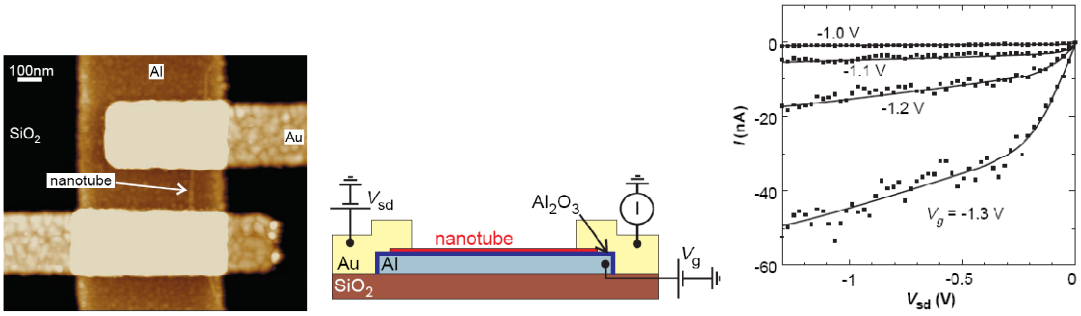



(a) (b)

Figure 2.5 IV curve of back gated CNFET: (a) $IV_G$; and (b) $IV_{Bias}$

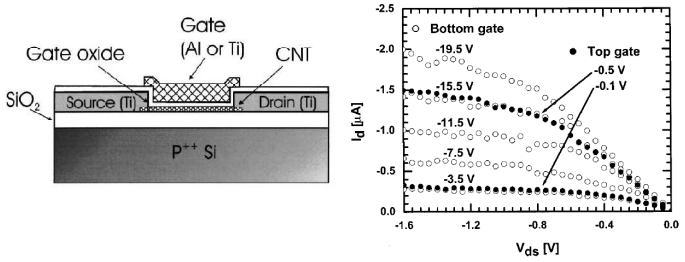
The pioneer works successfully demonstrated CNFET as a promising switch for future integrated circuit design. However, it is difficult to integrate multiple devices as a circuit based on the device layout from [29]-[30]. The Si substrate is used as a back gate, which means the same gate voltage is applied across the entire chip. In 2001, the group from Delft University of Technology enhanced their previous CNFET design by using aluminum local gates to control individual transistors as shown in Figure 2.6 (a-b) [13]. This design consists of narrow Al wire as a gate insulated by thin native $Al_2O_3$. Al gate patterns have been defined by e-beam lithography on silicon oxide, and gate insulation has been grown by exposing the Al gate into air.

Single-walled nanotubes have been deposited onto the wafer and situated on top of the predefined gates. Finally, source-drain contacts have again been created by e-beam lithography. The device transfer characteristics plotted in Figure 2.6 (c) show that this new CNFET works as an enhancement-mode p-type device.




(a) (b) (c)

Figure 2.6 (a) Atomic force microscope image of a single-nanotube transistor; (b) CNFET with individual Al back gate; and (c) device characteristics



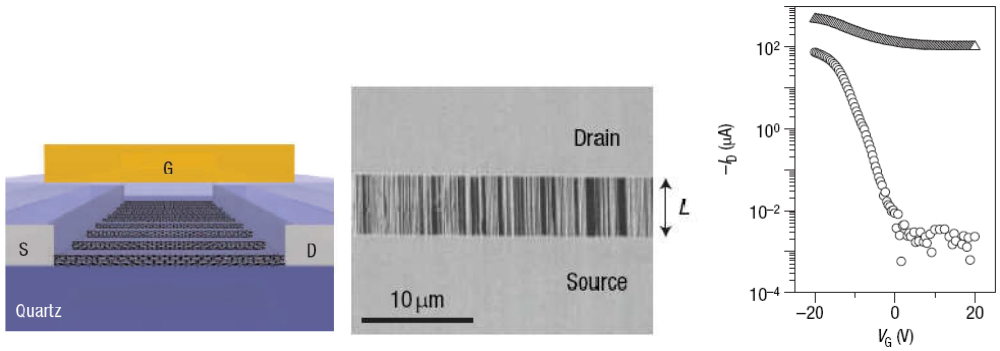

(a) (b)

Figure 2.7 (a) Top gated CNFET; and (b) IV characteristics of top and bottom gated devices

A major improvement of carbon nanotube devices was made in 2002 by creating a gate electrode on top of the nanotube channel separated by a thin layer of $SiO_2$ dielectric [15], which is the same as that of silicon MOSFET. The structure of this top gated design is shown in Figure 2.7 (a). Top gated structure has several important improvements compared to the back gated device. First of all, back gated devices normally consist of a relatively thick oxidation layer (~100 nm), which requires a high gate voltage to switch the device on. Top gated CNFETs have thin gate insulation 15 ~ 20nm and allow low operation voltage. Secondly, a top gate dramatically reduces the gate source-drain overlap capacitance, which is critical to high-frequency operation. Compared to back gated devices, which have carbon nanotubes

exposed to air, a top gated device encapsulates carbon nanotubes into the gate oxide and avoids this electrostatic instability problem, hence, improving the reliability of CNFETs.

This top gated CNFET can be fabricated on a single-crystal silicon wafer with 120 nm thermal $SiO_2$. CNTs have been deposited and titanium source-drain electrodes are patterned by e-beam lithography with spacing of 200 nm ~ 300 nm. A thin layer of gate oxide is then deposited, and finally the titanium gate electrodes are patterned by e-beam lithography. Figure 2.7 (b) compares this top gated device with previous back gated devices. The IV characteristics of the two structures have the same shape; however, the operation voltages for top gated devices are much lower (−0.5 ~ −0.1 V over threshold voltage) than bottom gated counterparts (−15.5 ~ −3.5 V over threshold voltage).




(a) (b) (c)

Figure 2.8 (a) Cross section, (b) scanning electron microscope image; and (c) IV characteristics of CNFET with multiple parallel tubes as channel region

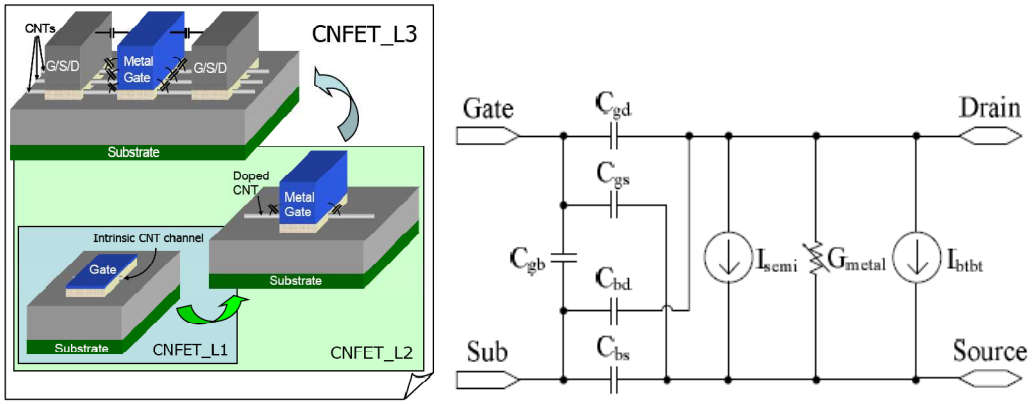
Single-nanotube devices reveal great performance improvement over existing Si-based devices. However, integration of a single tube into existing integrated circuits is still a great challenge. Due to limited fabrication control of single nanotube properties, a CNT device is susceptible to high electrical performance fluctuation. One feasible solution is to add densely packed, perfectly aligned horizontal arrays of non-overlapping SWCNTs as an effective channel, as shown in Figure 2.8 (a) [17]. This parallel conducting channel can provide a large amount of current and statistically averages device-to-device variation. Multiple carbon nanotubes in a channel region also increase the reliability.

Two major challenges have been solved in [17] to successfully fabricate and test the

aforementioned multiple carbon nanotube device. First, large-scale, high-density, perfectly aligned nanotube arrays need to be created. This is achieved by using a photolithography defined parallel pattern on quartz surface. Carbon nanotubes are grown in CVD (chemical vapor deposition) along these predefined patterns. The fabrication technique can successfully fabricate nanotube arrays with average diameter ~1 nm and length over 300 μm with 99.9% alignment (Figure 2.8 (b)). As mentioned previous, intrinsically, one third of fabricated carbon nanotubes are metallic. Those metallic carbon nanotubes cannot be controlled by gate voltage and are always conducting, which harms device on-off ratio. Metallic nanotubes can be removed by techniques such as electrical breakdown [31]. Figure 2.8 (c) demonstrates that after the electrical breakdown process, the device on-off ratio can be improved by four orders of magnitude. It is worth mentioning that the aforementioned fabrication processes can also be applied on unusual substrates such as flexible plastics.

### 2.2.2 CNFET Modeling

To maximize ease of use, models should be compatible with SPICE, the industrial-standard circuit simulator. The most comprehensive and well-known SPICE compatible CNFET model (Figure 2.9) was created by Stanford University and presented in [32]-[33].




(a) (b)

Figure 2.9 (a) Relationship between the three levels of the CNFET model; and (b) equivalent circuit of the CNFET intrinsic channel region

This CNFET ballistic model covers MOSFET-like structures, and is implemented in three levels, as shown in Figure 2.9 (a). Level 1 models the transportation in the intrinsic channel region under the metal gate. This level does not include any parasitic capacitance and resistance. The equivalent circuit for the intrinsic channel region including the trans-capacitance network is shown in Figure 2.9 (b). Like traditional silicon MOSFET SPICE models, the core part of the equivalent circuit is the voltage-controlled current sources. The three voltage-controlled current sources represent the thermionic current contributed by the semiconduting sub-bands ($I_{semi}$), the current contributed by the metallic sub-bands ($I_{metal}$), and the leakage current ($I_{btbt}$) caused by band-to-band tunneling. Note that $I_{metal}$ is equivalently modeled as a voltage-dependent conductance.

The level 2 model [33] is an extension of the level 1 model and considers the device non-idealities such as elastic scattering within the channel region, resistance and capacitance of the doped source-drain regions, and Schottky barriers formed by the metal contacts. The level 3 model [33] can be applied for a channel region containing an array of multiple nanotubes. The $n$ nanotubes in the channel can be categorized into two groups: the two carbon nanotubes on the edges and $n-2$ nanotubes in the middle. All of the CNTs within the same group are treated identically and each group considers different charge screening effects. The nanotubes in these two groups are connected in parallel for increased drive strength and reliability.

## 2.3 CNT Logic

As CNFETs have demonstrated promise as future electronic devices, the research community is making a significant effort to integrate simple CNFET devcies into complex logic circuits. The first CNFET logic gates were demonstrated in [13]. A range of digital logic operations was demonstrated, including an inverter, a logic NOR, a static random-access memory cell, and an ac ring oscillator containing one-, two-, and three-transistor circuits. This first work, however, was implemented using resistor-transistor logic, in which the CNFETs were connected to large off-chip resistors. The ring oscillator was implemented by connecting three inverters in

series (Figure 2.10 (a)) and achieved a frequency of 5 Hz. This low frequency was determined by the gigaohm resistance and a 100 pF parasitic capacitance from the wires connecting to the off-chip bias resistors.

Recently, a multi-stage top gated complementary CNFET ring oscillator has been built on a single 18 μm-long SWCNT (Figure 2.10 (b)) [34]. This ring oscillator consists of 12 individual CNFETs, 6 p-type FETs (purple) with Pd metal gates and 6 n-type FETs (blue) with Al gates. Five inverters were used for oscillation and another inverter was used for reading. A frequency response up to 52 MHz was measured. This measured frequency was still limited by the parasitics rather than by the intrinsic nanotube speed.

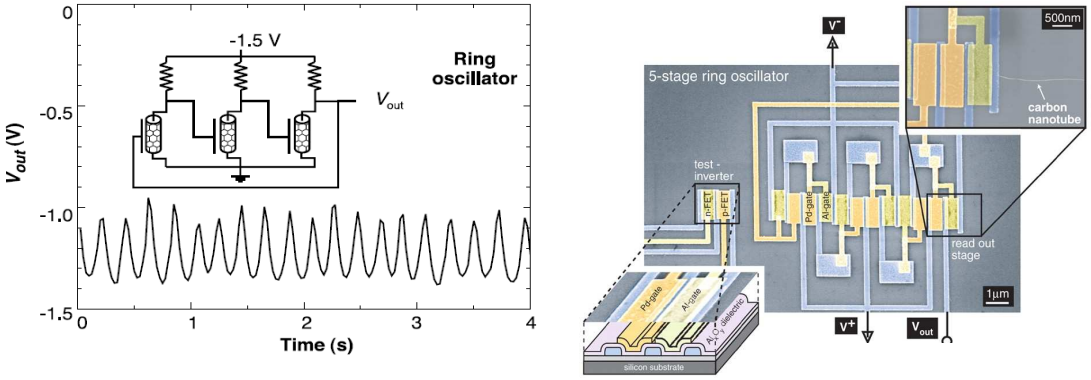



Figure 2.10 (a) Three stage ring oscillator consists of p-type CNFETs and resistors; and (b) scanning electron microscope image of a SWCNT ring oscillator

## 2.4 NRAM

NRAM is a nonvolatile NEMS memory device formed by the suspension of metallic CNTs over a trench that contains a base electrode (Figure 2.11). Bistable on-off states at the crosspoints are related to the two minimum energy points observed on the total energy curve, which is given by [35]-[36] in Equation (2.1):

$$E_T = E_{vdw} + E_{elas} + E_{elec} \tag{2.1}$$

where $E_T$ is the total energy of the memory element, $E_{vdw}$ is the van der Waals energy (vdW), $E_{elas}$ is the elastic energy, and $E_{elec}$ is the electrostatic energy.

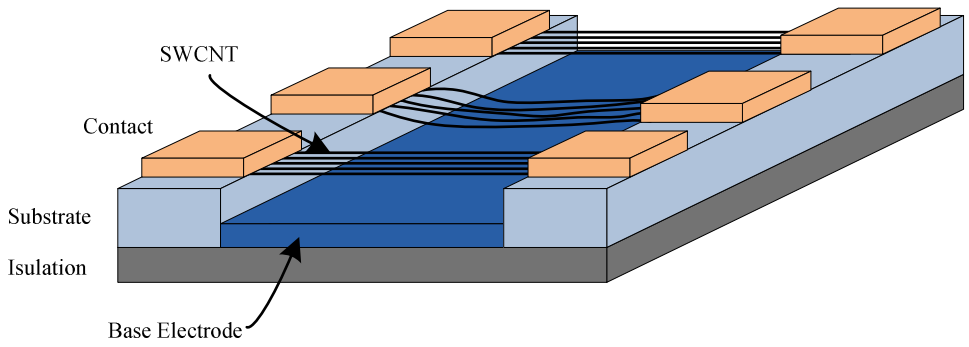



Figure 2.11 Architecture of NRAM memory cell

When the nanotubes are freely suspended (a finite separation between bottom electrode), the elastic energy is minimized, producing the first minimum total energy location. This represents the off state, when the junction resistance between separated nanotubes and electrode is very high. When the suspended nanotubes are deflected into contact with the lower base electrode, the attractive van der Waals force is maximized and a second minimum total energy location is created. This second location represents the on state, where the junction resistance will be orders of magnitude lower. Since these interactions are purely molecular, no power is consumed when the memory is at rest. Programming is accomplished by applying either attractive or repulsive voltages at the CNT and base electrode. This creates an electro-mechanically switchable, bistable memory device with well-defined off and on states [35]-[36].

## 2.5 CNT Bundle Interconnect

As integrated circuit dimensions scale down, the resistivity of copper (Cu) interconnect increases due to electron surface scattering and grain-boundary scattering, leading to a communication bottleneck. Metallic CNTs are a promising replacement because they offer superior conductivity and current carrying capabilities [37]-[39]. Since individual SWCNTs can have a large contact resistance and an intrinsic resistance that is independent of wire length, a rope or bundle of SWCNTs is used to transfer current in parallel.

The performance improvement of SWCNT bundle interconnect over copper interconnect

is shown in Figure 2.12, assuming the SWCNT bundle consists of densely packed SWCNTs with diameters of 1nm. It has been concluded in [37] that

- The best application for a SWCNT bundle is for long interconnects with small dimensions. This is because for a long SWCNT bundle ohmic resistance is dominant and the contact resistance is insignificant. In the meantime, copper suffers from increasing resistivity as it scales down. For a SWCNT bundle width of approximately 22 nm at length of 5000 μm, the improvement in resistance is 82 %.
- For long bundles with large widths, the contact resistance of the bundle is still insignificant, but copper has low resistivity close to its bulk value. The overall improvement of SWCNT bundle interconnects is therefore decreased to 61 % over copper.
- For short bundle lengths, although a SWCNT bundle has large contact resistance, it can still outperform copper because copper has exponentially increased resistivity due to scattering at narrow widths.
- SWCNT bundles are at a disadvantage for short interconnect lengths and large widths. The contact resistance is dominant compared to the ohmic resistance and the resistivity of the copper interconnect is low.

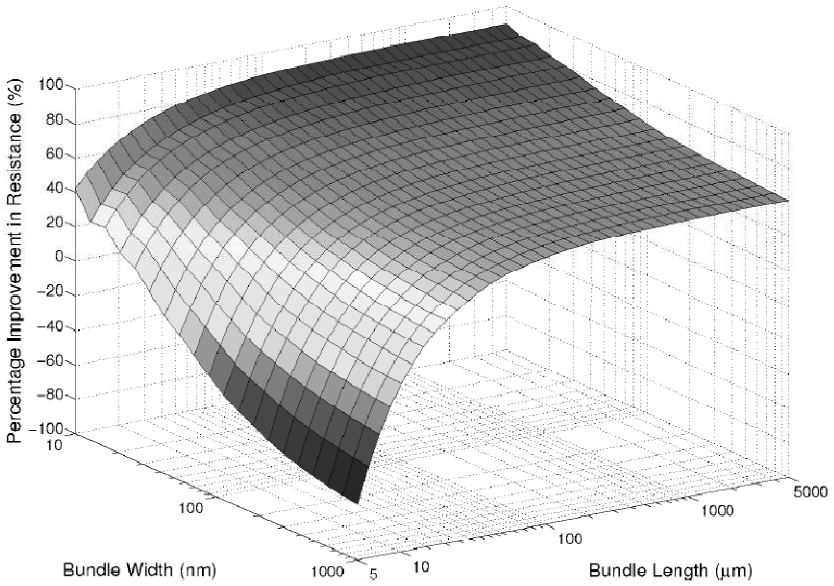
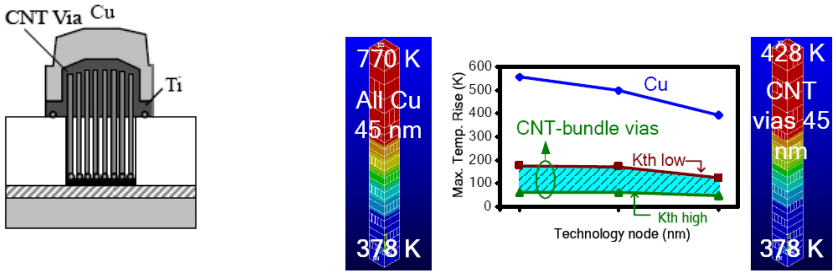



Figure 2.12 CNT bundle interconnect resistance

Besides horizontal wires, SWCNT bundle vias (Figure 2.13 (a)) also offer high performance and high thermal conductivity (more than 15 times higher than copper [40]). In nanoscale circuits, vias are prone to material deterioration, such as void formation and subsequent breakdown, caused by high current densities in small holes and current crowding effects at the edges. An SWCNT bundle would be much less susceptible to damage compared to metal due to its high current carrying capability (more than 100 times of that of copper). In addition, as shown in Figure 2.13 (b) [10], by integrating SWCNT bundle vias with copper interconnects, the temperature rise of the interconnect layers is much lower. This thermal property of SWCNT bundles is specifically useful for 3D ICs to combat thermal penalty. Large bundles of SWCNTs can be used as thermal vias to directly connect to the heat sink and efficiently dissipate the excessive heat [10]-[11].




(a) (b)

Figure 2.13 (a) Structure of SWCNT bundle vias; and (b) maximum temperature rise for Cu and SWCNT bundle vias

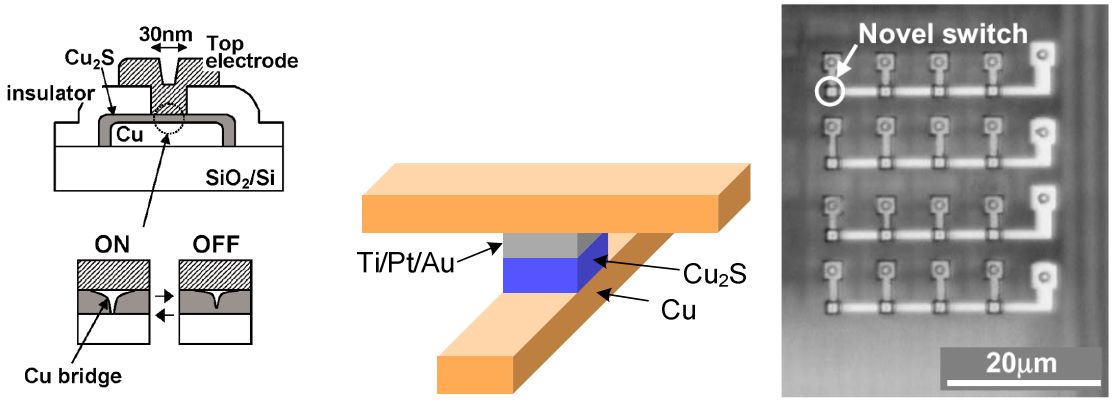
A recent advancement for CNT bundle fabrication is the integration of its fabrication into the CMOS fabrication process. In November 2006, a CMOS-compatible process was announced by Fujitzu, Japan [41]-[42]. It is essentially a two-step process consisting of a catalyst preparation step followed by the actual synthesis of the nanotube. This CMOS-compatible process will enable the practical applications of CNT bundle-based interconnects and vias into CMOS ICs.

## 2.6 Nano-Switches for Routing

Solid-electrolyte switches are a new type of nanoscale switch developed [43]. A solid-electrolyte switch is created by sandwiching a layer of $Cu_2S$ between two metals, a top

electrode (Ti, Pt, or Au) and bottom layer of Cu (Figure 2.14 (a)).

When a negative voltage is applied at the top electrode, Cu ions in the $Cu_2S$ are electrochemically neutralized by the electrons coming from that electrode, and a conductive bridge between the two electrodes is created, turning the switch on. An on-state resistance of as low as 50 Ω can be achieved by continually applying negative voltage to make the nano-bridge thicker. Similarly, the bridge can be ionized and dissolved by applying a positive voltage to the top electrode, turning the switch off.




(a) (b) (c)

Figure 2.14 Programmable solid-electrolyte switch: (a) single solid-electrolyte switch; (b) implementation in metal interconnect; and (c) SEM image of a 4 × 4 crossbar switch array

Because this design does not depend on a substrate, the switches can be manufactured between the higher layers of metal interconnect that are used for routing, as shown in Figure 2.14 (b). This figure shows how a Cu interconnect line can serve as the bottom layer in the electrolyte switch. In addition to individual devices, crossbars can be made from switch arrays. An SEM image of a prototype 4 × 4 crossbar is shown in Figure 2.14 (c), from [43].

Another radical post-silicon switch is based on nanowire crossbars which have hysteretic resistors formed at the points where two nanowire arrays cross each other (Figure 2.15). Similarly to solid-electrolyte switches, the hysteretic resistors can be configured into different resistances by applying programming voltage. Various research groups [44] have fabricated and tested crossbar memories using metal nanowires and organic molecular switches. Using nanoimprint lithography, parallel 2D nanowires of 5 nm width and 14 nm pitch have been

fabricated [45].

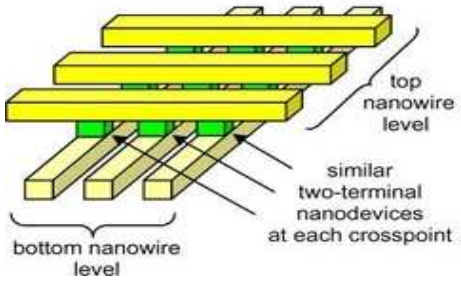



Figure 2.15 Nanowire crossbar

## 2.7 Island-Style Baseline FPGA

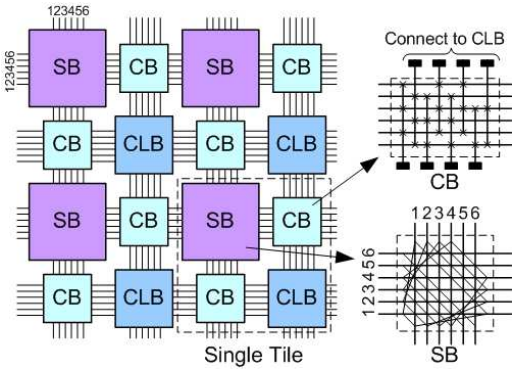



Figure 2.16 Schematic of a baseline 2D FPGA

Figure 2.16 shows a traditional 2D FPGA architecture (baseline). It consists of a number of tiles and each tile consists of one switch block, two connect blocks, and one configurable logic block (CLB). Each CLB or cluster (Figure 2.17) contains some local routing structures to route input signals to several basic logic elements (BLEs) and also connect BLEs together. In this figure, *I* represents the number of inputs the CLB has, and *N* represents the number of BLEs the CLB contains. *K* represents the size of a BLE. Each BLE consists of one *K*-input lookup table (*K* LUT) and one flip-flop. A *K* LUT can implement any logic function with up to *K* variables.

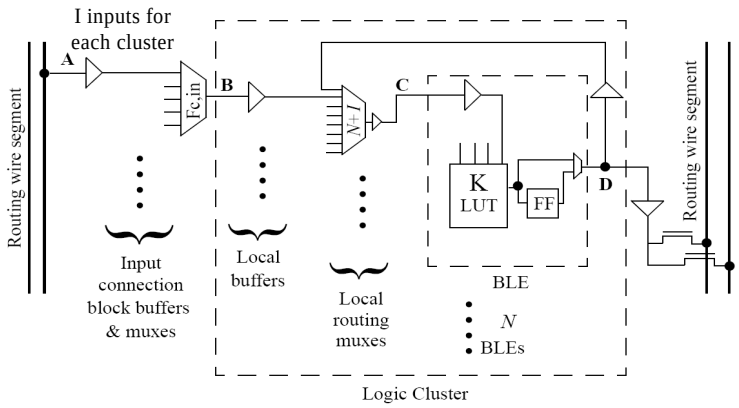
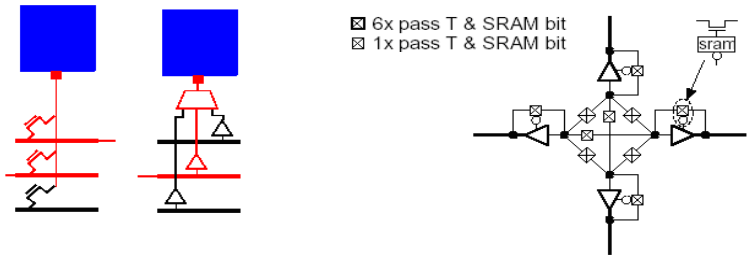



Figure 2.17 Schematic of a logic cluster or CLB




(a) (b) (c)

Figure 2.18 (a) Pass transistor-based CB design; (b) MUX-based CB design; and (c) SB connections

The CLBs connect to the routing channels through connection blocks (CB). The global routing structure consists of two-dimensional segmented interconnect channels connected by programmable switch blocks (SB). Typical designs of CB and SB are shown in Figure 2.18. There are two ways of connecting routing wires to the CLB: one is through the pass transistor (Figure 2.18 (a)) and one is through the multiplexer (MUX) (Figure 2.18 (b)). Figure 2.18 (c) shows that wires from four directions (each wire represents one track in the horizontal or vertical routing channels) are connected through bi-directional tri-state buffers. Each wire can potentially drive three other wires.

# CHAPTER 3
# 3D NFPGA: A 3D CMOS/NANO HYBRID RECONFIGURABLE ARCHITECTURE

The major performance and power bottleneck of the field programmable gate array (FPGA) is the programmable interconnects and routing elements inside the FPGA, which have been found to account for up to 80% of the total delay [46] and up to 85% of the total power consumption [47] when both local and global interconnects are considered.

Three-dimensional (3D) integration [2]-[4] increases the number of active layers and optimizes the interconnect network vertically. Both wire resistance and capacitance would drop proportionately; that is, power would drop by a factor of $(N_{layers})^{1/2}$ and wire (RC) delay would drop by a factor of $(N_{layers})$.

The application of the novel nanoelectronic materials (nanomaterials) and devices to establish FPGAs sheds new light on building future programmable devices. As mentioned in Chapter 2, carbon nanotubes (CNTs), nanowires, and other molecular electronic devices have shown strong promise in the literature.

Motivated towards integrating the two aforementioned leading technologies, a 3D FPGA structure, namely, 3D nFPGA, is presented, in this chapter. The novelty of this 3D nFPGA lies in the combination of 3D FPGA architecture design and nanotechnology, which will significantly advance future large-scale programmable devices. Furthermore, an efficient CMOS-nano hybrid method is used, so that the advantages of CMOS devices, nanotube interconnects and vias and nanowire crossbar programmable elements are utilized.

## 3.1 Existing Works

Several CMOS-based 3D FPGA structures have been proposed by stacking together a

number of 2D FPGA bare dies. The architecture in [49] implements inter-cluster routing in one layer and clusters (logic blocks or CLBs) and intra-cluster routing in another layer. The architecture in [50] spreads LUTs into different active layers and routes through 3D switch boxes. Recently, a three-layer 3D FPGA was proposed in [4], which is a monolithically stacked CMOS-based 3D FPGA. It follows the 2D FPGA architecture and efficiently divides it into three layers for configuration memory, switching, and logic. The main advantage of such an approach is that, in principle, it can achieve comparable vertical via density and scale at the same rate as the baseline CMOS technology. It shows a 1.7× performance gain on average compared to the 2D FPGA. None of aforementioned works considers nanomaterials or CMOS-nano hybrid systems.

Recently, several 2D FPGA structures built purely with nanomaterials have been proposed. An array architecture for nanoscale devices was suggested in [51]. This design is an island-style architecture in which clusters of nanoblocks and switch blocks are interconnected in an array structure. Each nanoblock is a grid of nanowires that can be configured to implement a three-bit input to three-bit output Boolean function and its complement. There are routing channels between the clusters to provide low-latency communication over longer distances. A programmable logic array (PLA)-based architecture, namely, nanoPLA, was presented in [52]. This architecture uses crossed sets of parallel semiconducting nanowires. Decoders which address each individual nanowire, can program nanowire crossbar arrays into logic-OR planes by applying a voltage differential across a pair of crossed nanowires. Nanowire field effect transistor restoring units are attached at the output of the 3D programmable logic-OR place to restore the output signals. The restoring unit is able to invert its input so that the logic-NOR plane can be provided.

There are some 2D CMOS-nano FPGA architectures. Reference [53] uses nanowires of different widths and materials as interconnects and replaces pass transistor switches with programmable molecular switches. The clusters are still implemented with CMOS. It is shown that this new architecture could reduce chip area by up to 70% compared to the traditional CMOS

FPGA architecture (scaled to 22 nm). Reference [54], in contrast to [53], presents a nanowire-cluster-based FPGA, and the inter-cluster routing remains at CMOS scale. It shows up to 75% area reduction (when LUT inputs = 7) with comparable performance to traditional FPGAs. In [24], a promising cell-based architecture called CMOL (CMOS nanodevice hybrids) was proposed. It utilizes an interface scheme by using special doped silicon pins implemented on the surface of the substrate to provide the contacts between nanowires and the CMOS layer. Therefore, logic functions are implemented by CMOS inverter arrays and nanowire-molecular-switch based OR logics. Signals are routed through nanowires and selectively configured crosspoints.

A generalized CMOL architecture, named FPNI (field programmable nanowire interconnect), was proposed in [55]. Different from the CMOL's inverter array architecture, the logics of FPNI are implemented with logic gate arrays (n-input NAND/AND together with buffers and flip-flops) in the CMOS layer, and nanowires are used for routing purpose only. This architecture allows simpler fabrication compared with CMOL because it requires less alignment accuracy between the CMOS and nanowire layers, and offers greater flexibility for creating nanodevices. Compared with traditional FPGA design, FPNI significantly reduces the chip area, but suffers from lower clock speed. Note that all these nano-FPGA structures mainly use nanowire crossbars and molecular switches. Researchers also attempted to use CNT-based memories (i.e., NRAM [56]) to be embedded into FPGAs to store bit configuration data [57].

It is noted that none of these nano-FPGA works utilizes 3D integration techniques. Only very recently, reference [58] has proposed a 3D programmable logic structure, solely based on nanowires. Compared with that work, the 3D nFPGA introduced in this chapter utilizes both CMOS and nanotube and nanowire building materials and takes advantages of both mature CMOS technology and advanced nanotechnology.

## 3.2 CMOS-Nano 3D nFPGA

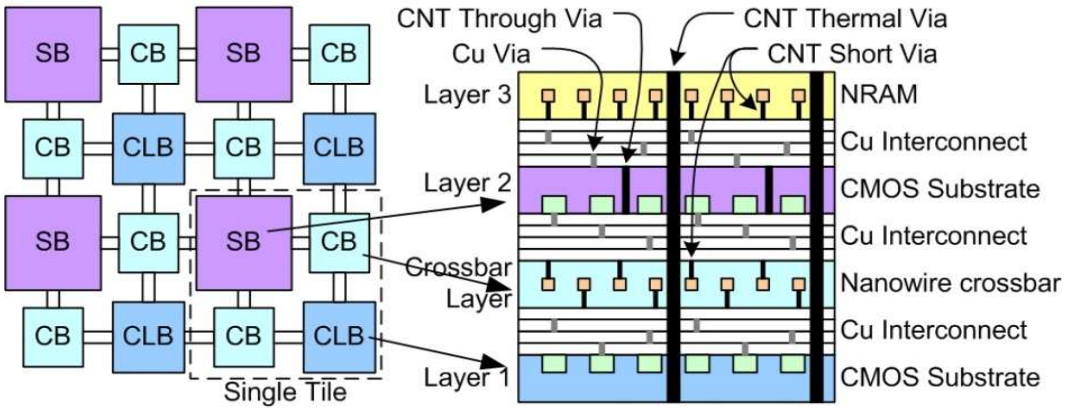
Instead of completely replacing the CMOS technology, future chips for nanotechnology

should be built as a hybrid using both CMOS (non-conventional CMOS such as strained silicon) and nanomaterials (such as CNT bundle interconnects and nanotube and nanowire crossbar memories), thus taking advantages of both mature CMOS technology and novel advances in nanotechnology.

As shown in Figure 3.1 the large 2D footprint of the FPGA is efficiently distributed into three layers in the 3D nFPGA. A 3D nFPGA consists of a 3½-layer structure, which can integrate the CMOS-based logic devices, nanowire-based memory and routing elements, post-silicon block memories, and CNT-based vias in three dimensions: (1) layer 1: the CMOS-based enhanced clusters of BLEs; (2) crossbar layer: integration of CLB local routing, connection blocks, and distributed memory blocks built by crossbars (this layer has no substrate and is considered as a half layer); (3) layer 2: CMOS-based enhanced switch blocks and local interconnects; and (4) layer 3: NRAM-based block memories and local interconnects (Figure 3.1 (a) does not show the block memories of the baseline FPGA). Layers 1 and 2 are bonded face-to-face with the crossbar layer in the middle. Layers 3 and 2 are bonded in a face-to-back manner. Communication between the layers is based on CNT bundle via networks.

The following items summarize the unique features of this architecture.

- A novel combination of logic, crossbar, and switch layer designs




(a) (b)

Figure 3.1 Components distributions of a 2D FPGA into the 3D nFPGA. (a) 2D baseline FPGA; and (b) 3½ layer 3D nFPGA

- Layers 1 and 2 are face-to-face for efficient via communication
- Crossbar layer is a novel incorporation of connection blocks, CLB local routing, and distributed memories
- Dramatic reduction of interconnects and FPGA footprint
- Vertical communication and thermal alleviation through CNT bundles
- Combination of both distributed memories and block memories to satisfy specific memory needs for control-intensive and data-intensive FPGA applications
- The 3½–layer structure or the bottom 2½–layer structure can be stacked multiple times on top of one another, enabling multi-stack 3D nFPGAs

### 3.2.1 Layer 1 – Reduced Logic Block (RLB)

A standard CLB comprises buffers, local wires, multiplexers (MUXs) and BLEs. The inputs of a CLB are routed to different BLEs through local routing elements such as MUXs. If the routing is fully connected or fully populated, that is, any BLE input can be connected to any CLB input, the local routing area is significant (for example, 65% of a CLB). This is the motivation to replace the CMOS-based routing elements with nanowire-molecular crossbars. By programming the molecular switches on or off at the crosspoints of a nanowire array, a CLB input can be routed to any BLE. This crossbar is implemented in the crossbar layer. As a result, the CLB footprint in layer 1 can be significantly reduced.

As shown in Figure 3.2, layer 1 consists of tightly packed BLEs from the original CLBs and the programming and addressing unit (PAU). The PAU is used for addressing the crossbar-based BLE routing in the crossbar layer. One layer 1 tile (named RLB) corresponds to the logic contained in the original CLB. Note that size-4 CLB (each CLB contains four BLEs) and four-input BLEs are used in this section simply for illustration purposes. This architecture can handle any reasonable CLB and BLE sizes for this transformation. Figure 3.2 shows four tiles for layer 1 as an example.

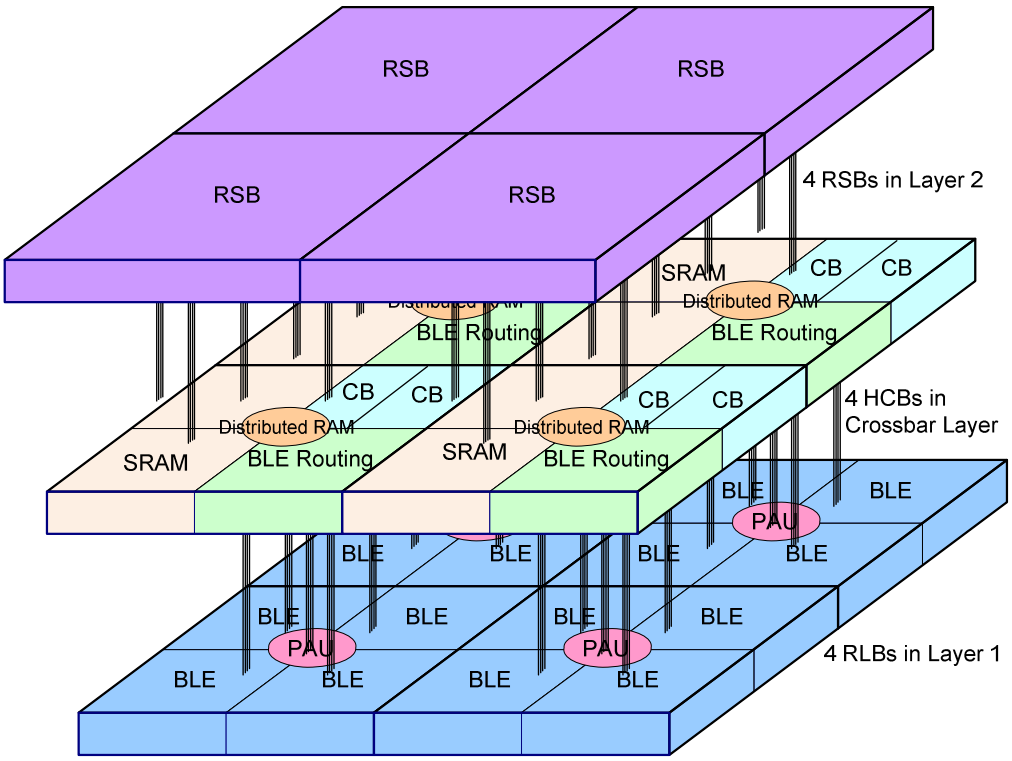



Figure 3.2 Two-and-a-half layer structure of nFPGA: layer 1 (reduced logic block), crossbar layer, and layer 2 (reduced switch block)

### 3.2.2 Layer 2 – Reduced Switch Block (RSB)

In baseline FPGAs, the global routing consists of connection blocks and switch blocks, which together take up a significant amount of the baseline FPGA footprint. For instance, if CLB size *N* (*N* BLEs per CLB) is 10 and BLE size *K* (each BLE has 4 inputs) is 4 (popular parameters for commercial FPGA products), the global routing area is 57.4%, and the total CLB area is 42.6% in the baseline FPGA [46]. Global routing area is thus very critical for FPGA footprint reduction for this 3D chip. Two techniques are applied to aggressively reduce the routing area. First, the majority of connection blocks are moved to the crossbar layer because they are multiplexer-based designs like the case in CLB local routing. Second, all the programming SRAM (static random access memory) cells of the switch blocks are moved to the crossbar layer as well and implemented by the nanowire crossbar memories. Therefore, one layer 2 tile (named RSB) is a switch block without SRAM cells plus the driving buffers which connect to the wire tracks and

drive the routing part (MUX in 2D, but replaced with nanowire crossbar in 3D nFPGA) of the connection blocks.

Taking a CLB size *N = 10* and a BLE size *K = 4* with a fixed routing channel *width = 100* as an example, the routing area of one tile can be partitioned as shown in Figure 3.3, where 47.8% of the area (SRAM cells area) of the switch block can be moved down and efficiently implemented at the crossbar layer. Only buffers driving the routing of the connection block remain in the switch layer, which takes only 17.5% of the connection block area. Combining the global routing area percentage with the detailed routing area partition, and by balancing routing resource into the switch layer and crossbar layer, a tile footprint that is only 22.4% of the 2D baseline footprint can be achieved — a more than 4× circuit area reduction.

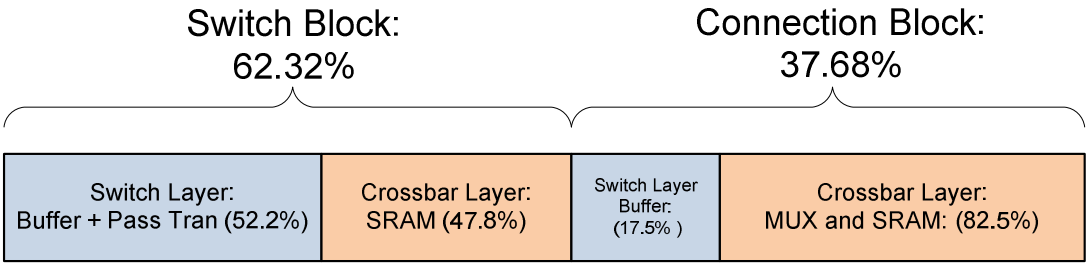



Figure 3.3 Global routing area partition

### 3.2.3 Crossbar Layer (Layer 1½) – Hybrid Communication Block (HCB)

One crossbar layer tile (named HCB) consists of one BLE routing block, two connection blocks, SRAMs for one RSB, and a distributed crossbar memory (Figure 3.2). All these functionalities can be realized because the crossbar layer is built by high-density nanowire ($10^{11}$ T/cm$^2$), much higher than the corresponding CMOS implementation ($2 \times 10^9$ T/cm$^2$ [1]). The connection blocks connect to the RSBs using up vias. They also connect to the BLE routing blocks on the same layer. The BLE routing blocks connect to the BLEs on layer 1 using the down vias.

Figure 3.4 shows how a BLE routing block works by an example. A BLE routing block receives inputs from adjacent connection blocks (Figure 3.2) and routes them to the

corresponding BLEs in layer 1 using CNT short vias. Note that these same inputs can be routed to multiple BLEs. In this example, the input signal A from CB1 (connection block 1) is routed to BLEs along a dotted line through down vias (vias is used to indicate that a group of vias connects individual inputs). The black dots at the crosspoints indicate the molecular switches that have been programmed as on state. The outputs of BLEs indicated by a dashed line can either feed back to the crossbar to connect to the inputs of other BLEs or output to adjacent connection blocks. In order to apply a programming voltage to an individual nanowire in the HCB, the PAU is required, consisting of address controllers and voltage terminals. This unit is included in layer 1 because these transistors can be efficiently implemented using CMOS. The dark blue bar in the left side of Figure 3.4 represents voltage sources for programming, which are about two times higher than the operation voltage. To control $n$ wires, $nlog_2n$ p-type transistors are required. These p-type transistors can address each nanowire and set the molecular switch at a crosspoint as either on or off state. The crossbar layer is an efficient interface between layer 1 and layer 2. The CNT short vias have metal contacts, which can establish a reliable connection to the local interconnects of layers 1 and 2.

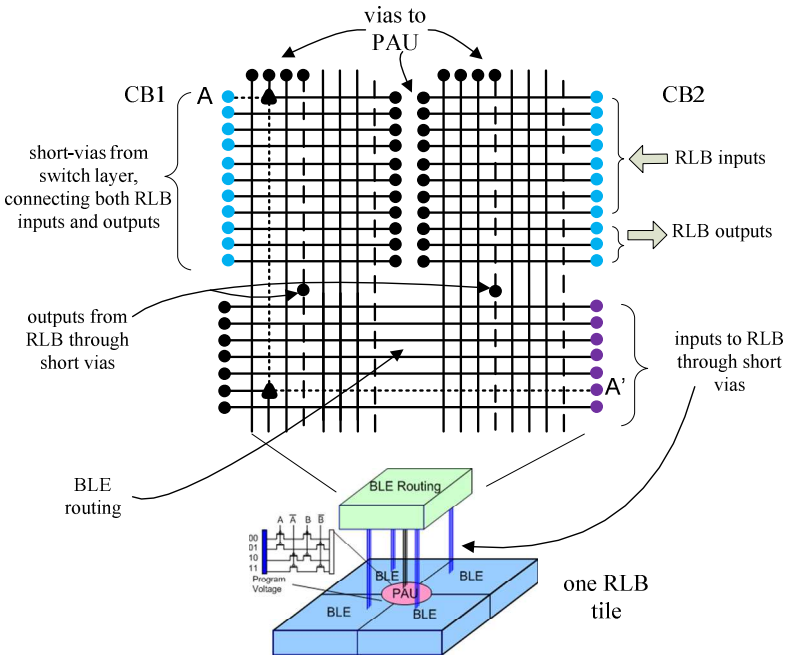



Figure 3.4 Detailed diagrams of BLE routing and PAU

### 3.2.4 Layer 3 – Block Memory Layer

I use NRAM (described in Section 2.4) in layer 3 as block memories for this architecture. They are able to store large amount of data suitable for data-intensive applications such as DSP (digital signal processor) and multimedia applications. In order to connect layer 3 (facing down) with layer 2, a face-to-back 3D IC bonding is applied and special vias called through vias are used to make the connections (Figure 3.1(b)). Because the through vias penetrate the substrate of layer 2, the density of these vias is ten times sparser than that of CNT short vias. This density is sufficient for buses and communication channels to serve the block memory. In order to obtain better via performance and thermal dissipation, the through vias are made with CNT bundles.

### 3.2.5 Hybrid Horizontal Interconnects

In the proposed structure, local horizontal interconnects are required inside layers 1, 2 and 3. CNTs are preferred over copper as interconnects. However, vertical CNT bundles are difficult to connect to horizontal CNT bundles. To overcome this difficulty, copper contacts and short copper horizontal interconnects can be used to set up the connections between vertical and horizontal CNT bundles. This hybrid approach considers both fabrication capability and performance optimization. The mixture of copper and CNT interconnects is applied for horizontal connections. For example, in layer 2, there can be short interconnects (e.g., single lines or double lines) that connect adjacent or neighboring RSBs and long interconnects (e.g., hex lines) that connect distance RSBs. This mixture of interconnects of different lengths is a common practice in modern FPGAs. Copper is used for short interconnects and CNT bundles for hex lines (or similar longer lines) to reduce interconnect delay. Note that the horizontal interconnect is much shorter than that in the baseline FPGA because of the dramatic footprint reduction in 3D nFPGA.

### 3.2.6 3D Stacks

The 3½-layer architecture or the bottom 2½-layer architecture (without the NRAM layer)

can be stacked, enabling multi-stack 3D nFPGAs. An example using 2½-layer stacking provides an excellent stacking architecture. The 2½-layer architecture is ideal for control-intensive applications. The distributed memories available on the crossbar layer can provide fine-grained register-file capabilities. As shown in Figure 3.5, two RSB layers are placed back-to-back. The RSBs on the two layers communicate using CNT through vias, which enable short and high-speed connections. In 2D FPGA, connecting distance cells can be very expensive in terms of delay and power. In 3D nFPGA, by utilizing the vertical dimension, the RSBs on the bottom stack not only can connect to other RSBs on the same layer but also can directly connect to those on the layer above. This provides a much more efficient interconnecting network and significant performance and power improvements.

The 3½-layer architecture can also be stacked. Note, for 3½-layer architecture, the RSBs of the two stacks cannot be stacked directly. Instead, longer through vias penetrating the block memory layer are required. When the stack number increases, the performance difference between multiple 2½-layer stacking and multiple 3½ stacking diminishes because multiple 2½-layer stacking will incur longer through vias as well, starting from the third stack.

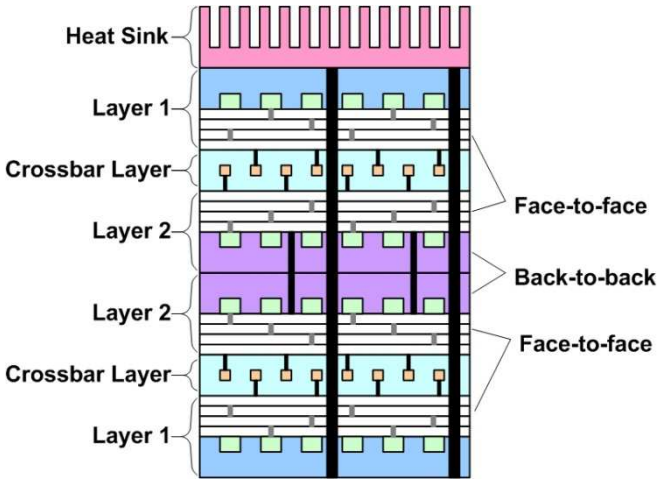



Figure 3.5 Two-stack (each stack is 2½ layers) 3D nFPGA architecture with two stacks connected back-to-back and thermal vias are inserted and linked to heat sink

### 3.2.7 Thermal Vias and Defect Tolerance

The additional features of 3D nFPGA include its emphasis on thermal optimization and defect tolerance. A major concern of the 3D IC is its thermal penalty. The 3D stacks will increase heat density, leading to degraded performance. It has been demonstrated in [59] that doubling the heat density without any improvement in cooling capacity will lead to more than 30% degradation in performance. The CNT bundles for short vias in this structure are thermal-efficient. In addition, large CNT bundles are used as thermal vias (Figure 3.5). The thermal conductivity of CNT bundles can be up to 5800 W/mK [60]. In addition, this conduction is in the direction along the length of the nanotubes because thermal conductivity in CNT bundles is anisotropic [60]. Therefore, CNT bundle vias will serve as more effective heat conductors compared to copper vias and can reduce the temperature gradient dramatically. As a result, the whole chip can cool down quickly. The size and the density of these thermal vias can be further optimized by taking into account other architectural parameters such as stack number, BLE size, short via and through via density, and so forth.

The proposed 3D nFPGA has excellent fault tolerance capabilities. The BLE and switching layers are based on CMOS technology, which offers very low defect rates. However, nanoelectronic circuits, such as the crossbar structure, always have a small percentage of defective components due to the statistical nature of the self-assembly fabrication process [51]-[52]. Errors and faults in a system could be either permanent (hard errors) or transient (soft errors). Reconfiguration, done either statistically or dynamically, is an effective solution to fix the hard errors, which is an intrinsic advantage of FPGA chips. For static reconfiguration, off-line self-test and self-diagnosis will be sufficient. To support dynamic reconfiguration, the design must have on-line self-test and self-diagnosis capabilities to detect and identify failures when a system is operating. Some existing techniques can be used to support these crucial features, such as probabilistic model checking and self-checking circuit design [53]. In addition, redundancy can be added into the crossbar layer with redundant rows and columns [61]. There will also be redundant vias and redundant molecular switches. The right amount of redundancy has to be

modeled and studied.

## 3.3 3D nFPGA Characterization and Evaluation

This dissertation evaluates performance and power of a 3D nFPGA architecture compared to the baseline 2D FPGA architecture. In order to have accurate evaluation, detailed delay and power characterization for both interconnects and devices are necessary. The interconnect characterization will be for copper wires used in the baseline FPGA and CNT bundle interconnects used in the 3D nFPGA. The device characterization is for CMOS-based MUXs used in the baseline case and nanowire-based crossbars used in the 3D nFPGA case. Also needed is a CAD flow that is able to use a set of well-accepted benchmarks and go through various design stages to report the final delay after circuit layout. The CAD flow for baseline 2D FPGAs is well studied [62]. This flow is adopted and made workable for the 3D nFPGA architecture. The following first presents the CAD flow and then introduces the delay and power characterization methods and related results.

### 3.3.1 CAD Flow

A timing-driven CAD flow shown in Figure 3.6 is used. Each benchmark circuit goes through technology independent logic optimization using SIS (system for sequential circuit synthesis) [63] and is technology-mapped to LUTs with size *K* using DAOmap [64], which is a popular performance-driven mapper working on area minimization as well. The mapped netlist then feeds into FPGA physical design tools, T-VPACK and VPR-LP, which perform timing-driven packing (i.e., clustering LUTs into the CLBs), placement, and routing [62] and further generate a BC-netlist for the power simulator fpgaEva_LP2 [47][48]. Afterwards, the critical path delay of the design and power consumption is obtained. This CAD flow is flexible: various parameters for LUT size *K*, CLB size *N*, routing architectures, and interconnect buffer sizes can be chosen. In this study, *K = 4*, *N = 10*, and *route channel width = 100*. In FPGAs, interconnects are segmented and driven by buffers. A mixture of interconnects with different

lengths provides better performance [62]. This study uses a mixture of length-4 and length-8 wire segments (wires crossing either four CLBs or eight CLBs in the baseline FPGA) in equal numbers to route the signals, which is reported as one of the best combinations [62]. All these parameters can be supplied through the architecture specification file.

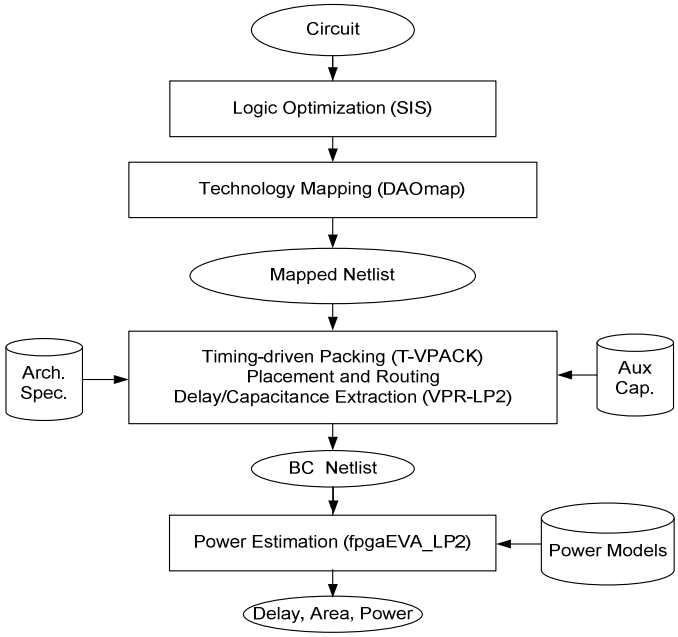



Figure 3.6 Evaluation framework for 3D nFPGA

### 3.3.2 Interconnect Characterization

The interconnect length scaling due to 3D stacking is the main reason for system performance and system dynamic power enhancement. To better understand the impact of 3D, the delay of length-4 and length-8 wire segments for both baseline FPGA and 3D nFPGA using HSPICE simulation is estimated. To obtain the actual lengths of these interconnects, the tile area based on the area model presented in Section 3.2.2 is first estimated. The baseline and the 3D cases are considered separately.

When estimating the lengths of wire segments for the baseline architecture, both the CLB area and the routing area are considered. Wire segmentation crosses a baseline tile with an area of 1561.5 $\mu m^2$. Therefore, length-1 interconnect for baseline would have a length of 39.52 μm.

Table 3.1 Interconnect delay characterization

| Wire Segments | Items | Copper Wire in Baseline | Copper Wire in 3D nFPGA | CNT Bundle Wire in 3D nFPGA |
|---|---|---|---|---|
| Length 4 | L (μm) | 158.06 | 74.859 | 74.859 |
| | R (Ω) | 1697.91 | 804.159 | 271.35 |
| | C (fF) | 11.555 | 5.472 | 8.653 |
| | D (ps) | 22.09 | 9.83 | 7.63 |
| Length 8 | L (μm) | 316.127 | 149.719 | 149.719 |
| | R (Ω) | 2863.87 | 1608.318 | 542.703 |
| | C (fF) | 19.489 | 10.945 | 17.306 |
| | D (ps) | 87.25 | 39.02 | 28.99 |

Next, I will examine the wire length for 3D nFPGA. Because 3D nFPGA distributes the switch blocks, connection blocks, and CLBs into three different layers, the situation is dramatically changed. A routing wire segment now only spans RSBs (Figure 3.2). RSB area is the area of the baseline switch block excluding SRAM cells (Section 3.2.2). The RSB area is estimated as 350.25 $\mu m^2$. Therefore, length-1 interconnect for 3D would have a length of 18.71 μm, which represents a 52.64 % length reduction compared to the baseline case. Table 3.1 shows detailed comparison data of the wire segments for both the baseline and the 3D nFPGA.

In Table 3.1, L, R, C, and D represent wire length, wire resistance, wire capacitance, and wire delay, respectively. The calculation of L, R, and C values of copper is well known. CNTs can be considered as quantum wires. Thus, CNT bundles will need to consider additional quantum resistance, quantum capacitance, and kinetic inductance [8],[10],[65]-[67]. In briefly describing the models used to derive the resistance and capacitance of CNT bundles, it is assumed that a CNT bundle interconnect is composed of hexagonally packed identical metallic single-walled CNTs [10]. The CNT bundle resistance is given by Equation (3.1)

$$R_{Bundle} = \frac{R_{Single} + R_{Contact}}{n_{CNT}} \tag{3.1}$$

where $R_{single}$ is the resistance of a single CNT wire and $n_{CNT}$ is the total number of CNTs forming the bundle. In considering the intrinsic capacitance and quantum capacitance of CNT bundles, the effective capacitance ($C_{Total}$) of a CNT bundle is a series combination of quantum and intrinsic capacitance.

Using these parameters, RC wire delay is then obtained through HSPICE. CNT bundle wire provides the best performance among the three cases examined — copper wire used in baseline 2D FPGA, copper wire used in 3D nFPGA (a fictitious case to show how copper interconnects in 3D nFPGA can help in terms of wire length and delay reduction), and CNT bundle wire used in 3D nFPGA (the architecture proposed in this dissertation). Note that this section only models interconnect delay in the routing architecture. The next section will model circuit path delay, including vias and nanowire-based devices. The capacitance of different length segmentation is also used for power estimation.

### 3.3.3 RC-Equivalent Circuits Extraction for Device Delay

Replacing the CMOS-based MUXs with nanowire crossbars not only significantly reduces the footprint of the chip but also enhances circuit performance. In this project, routing channel width $W = 100$ is set for all the benchmarks. This is often used in academic research to imitate the real FPGA routing architecture since modern FPGA chips usually provide sufficient routing resources, and a single FPGA device will have a fixed channel width. *Fc* is set at 0.5, which is also commonly used and provides connections between the CLB input and half of the routing tracks in the channel. The number of inputs *I* is 22 for the CLB [46]. For baseline architecture, this implies that thirty-two 50:1 MUXs (the MUXs marked with "*Fc,in*" in Figure 2.17) will be required in the connection block. In addition, another ten 32:1 local routing MUXs (22 CLB inputs plus 10 feedback wires from the 10 BLE outputs — the MUXs marked with "*N+I*" in Figure 2.17) are also necessary to route the cluster inputs and feedback wires to individual BLEs.

As explained earlier, MUX can be easily and efficiently implemented by nanowire crossbar. A 50:1 MUX can be constructed as 50 vertical wires crossed by 1 horizontal wire. A second MUX is simply one additional horizontal wire. A $50 \times 32$ crossbar array can serve the same functionality as the connection block in the baseline FPGA. These crossbars are especially suitable for defect tolerant designs. Considering the defects, redundant wires can be used,

requiring a larger crossbar. Even this larger crossbar is efficient due to the high-density property of the nanowires crossbar. For example, a square crossbar array with 50 × 50 nanowires only requires a 5.6 μm × 5.6 μm dimensional array at 32 nm technology.

The CAD flow shown in Figure 3.6 is ideal for the baseline FPGA. To make it work for the 3D nFPGA, various circuit models to capture the specific characteristics of 3D nFPGA architecture must be built. The architecture specification file of VPR (versatile packing, placement and routing for FPGAs) supplies delay values for various combinational circuit paths to enable accurate timing analysis. For example, in Figure 2.17, there are paths A→B, B→C, and D→C. Corresponding equivalent circuits are needed to implement these paths in 3D nFPGA. The difference now is that part of the path may go through a CNT bundle via or a nanodevice and may also go vertically instead of horizontally compared to the baseline case. These different paths are extracted for 3D nFPGA and HSPICE simulation is performed to compute their delays.

As shown in Figure 2.17, the wire track to CLB input path A→B of baseline FPGA consists of a buffer and a MUX in a connection block. For 3D nFPGA, the corresponding path consists of a CNT via between the switch layer and the crossbar layer, nanowire segments, and a programmable switch. This path is represented by resistors and capacitors in an equivalent circuit, illustrated in Figure 3.7 (a). Another example in Figure 3.7 (d) shows the equivalent circuit of local feedback path D→C in nFPGA. It can be modeled as a conducting path consists of an up via to the BLE routing box (Figure 3.4), nanowire crossbar, and a down via to the destination BLE. Other paths are illustrated in Figure 3.7 as well.

In this study, NiSi nanowire and molecular programmable switches are used. The cross section of nanowire is assumed as square; the distance between adjacent nanowires is assumed to be equal to the wire width. The insulation material around the nanowires is set to have a dielectric constant of 3.9. Applying the above configurations, provides the following equations for nanowire:

$$R_{nanowire} = \frac{\rho_{nanowire}}{Area} \times L \tag{3.2}$$

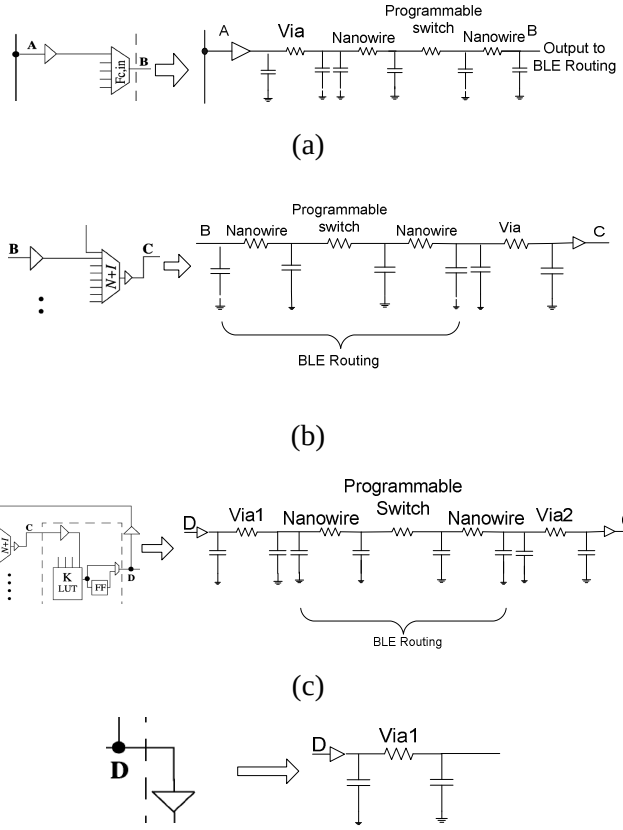



Figure 3.7 Extracted equivalent circuits of 3D nFPGA. (a) Wire track to CLB input; (b) CLB input to BLE input; (c) local feedback; and (d) CLB output

$$C_{nanowire} = \frac{\varepsilon_{ox}}{d} LW \tag{3.3}$$

where L is the nanowire length, and d is the thickness of the insulator. Resistivity ρ of nanowire is obtained based on the work of [68]. A unit resistance $R_0$ = 143 Ω/μm and a unit capacitance $C_0$ = 300 aF/μm is derived. The programmable switch has an on resistance plus a contact resistance (to nanowire) below 1 KΩ. CNT vias are extracted by using the same models of CNT interconnects assuming an interconnect length of 0.02 μm. Based on these parameters, the equivalent circuits are simulated in HSPICE. The performance comparisons are listed in Table 3.2. A 44.79% performance enhancement is achieved on average. The D→out delay in baseline FPGA is better than that in 3D nFPGA. The reason is as follows. D→out models the delay from BLE output to the output of CLB. It consists of one tri-state buffer (size 10×) to drive output

wires in the routing channel. Besides the output buffer, 3D nFPGA has an additional via delay, which occurs during the signal propagation from the BLE layer to the switch layer. This contributes extra delay for the 3D nFPGA case.

Table 3.2 Performance comparison of baseline and 3D nFPGA

| Paths | CMOS-BasedDelay (ps) | Nano-BasedDelay (ps) | Enhancement |
|---|---|---|---|
| A → B | 141.66 | 36.126 | 74.49% |
| B → C | 107.59 | 35.429 | 67.07% |
| D → C | 107.59 | 48.575 | 54.85% |
| D → Out | 28.481 | 33.367 | −17.16% |
| Ave. | | | 44.79% |

### 3.3.4 Macro Power Models

The gate-level FPGA power estimator fpgaEva_LP2 [47] requires both switch level models and macro models for power estimation. The switch level model uses extracted capacitance to model the power consumed during signal transition. A macro model predefines a circuit component using HSPICE simulation. Both dynamic and static power of size-4 LUT and various sized buffers based on the BSIM 32 nm model were studied. Randomly generated input vectors with equal occurrence probability are used to obtain the average power consumption per access to the LUT. In this chapter, only a size-4 LUT was studied. However, it is easy to extend to other LUT architectures by listing power data into a user-defined library of fpgaEva_LP2.

To correctly model the crossbar based BLE routing; a nanowire crossbar array was also simulated with HSPICE. Shown in Figure 3.4, comparing to MUX based 2D baseline design, CLB input capacitance of nFPGA now is replaced with capacitance of electrically connected nanowires (A to A` in Figure 3.4) plus crosspoint switch capacitances and necessary via capacitances. 2D intra-cluster local feedback capacitance, which was molded as length-1 wire segment capacitance plus buffer input capacitance, is replaced by nanowire capacitance and via capacitance in 3D as well. Consider $N$ = 10 and $K$ = 4; Table 3.3 lists some of the extracted capacitance values of different architectures. Leakage power of the crossbar array is captured by modeling each crosspoint as a diode with an on or off resistance. The equivalent circuit is shown

in Figure 3.8 [24]. For $N$ = 10 and $K$ = 4 architecture, crossbar of one tile has a leakage power 1.53E−06 watt.

Table 3.3 Capacitance extracted from fpgaEva_LP2 (unit: fF)

| | 2D Baseline | 3D nFPGA Copper Wire | 3D nFPGA |
|---|---|---|---|
| CLB Input | 2.84 | 3.61 | 3.61 |
| BLE Output without feedback | 1.47 | 3.61 | 3.61 |
| BLE Output with feedback | 14 | 5.60 | 5.60 |

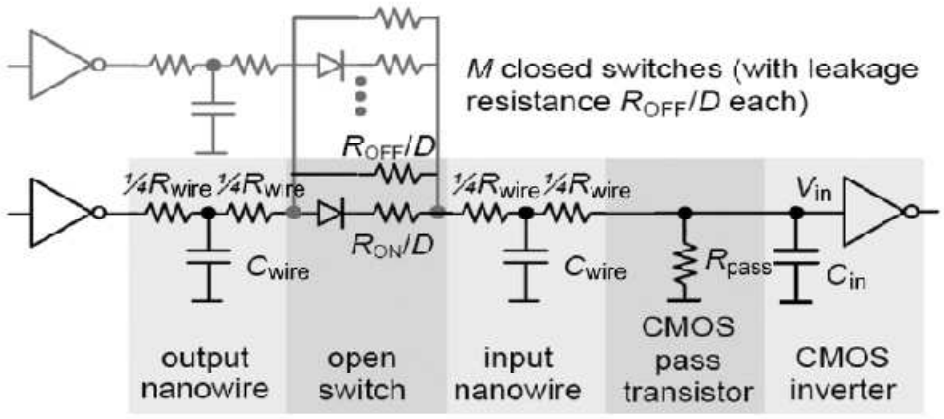



Figure 3.8 Equivalent circuit for nanowire crossbar leakage power simulation

## 3.4 Experimental Results

In this section, the overall performance improvement of the 3D nFPGA over the baseline counterpart is quantified. The performance improvement is achieved from a combination of 3D architecture, CNT bundle interconnects, and nanowire-based crossbar array. The experiment is based on a 32 nm technology platform. The 20 largest MCNC benchmarks are mapped and fit to both baseline and 3D nFPGA using the CAD flow and the detailed delay characterization data presented in the previous section.

Figure 3.9 shows the view graph of different critical path delays for each benchmark

collected for three different architectures — the baseline FPGA, 3D nFPGA with copper interconnect for routing (a fictitious case to show how copper interconnects for 3D nFPGA perform in terms of delay), and real 3D nFPGA. Table 3.4 shows the detailed delay values for the same three architectures and also the comparison results. On average, 3D nFPGA with copper interconnects provides a 2.05× performance gain (in terms of Fmax) compared to the baseline, and real 3D nFPGA provides a 2.65× gain compared to the baseline. It should be stressed that the only difference between 3D nFPGA with copper interconnects and the real 3D nFPGA is that real 3D nFPGA uses CNT bundles for the routing interconnects and vias. Overall, by using nanowire-based crossbars to shrink the MUX area and by 3D stacking, the performance gain of 3D nFPGA is very significant. Moreover, CNT bundle wires can offer an additional 0.6× for overall performance improvement.

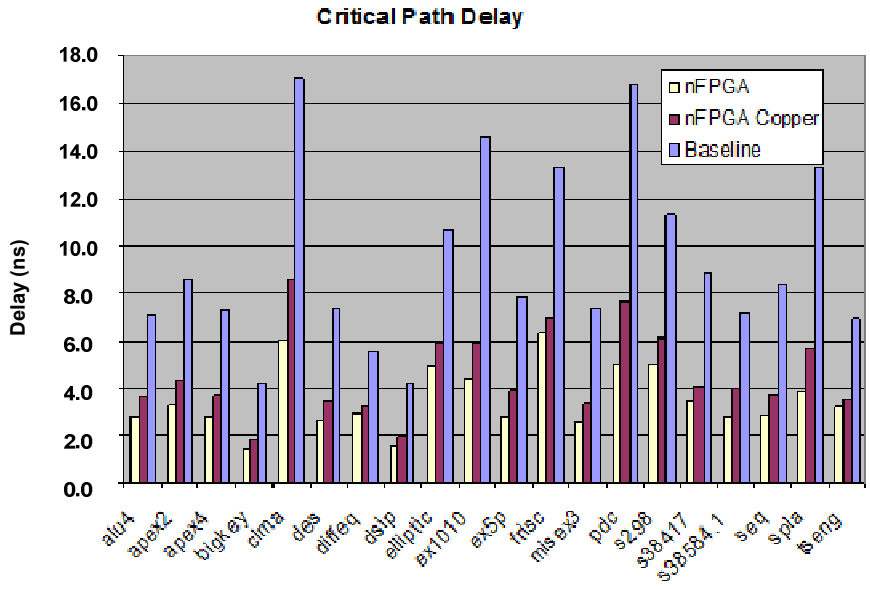



Figure 3.9 Critical path delay comparison for three architectures (1st bar in each group is critical path delays of nFPGA, 2nd bar in each group is critical path delays of nFPGA with copper global interconnect, and 3rd bar in each group is critical path delays of baseline FPGA)

Power consumptions of different architectures are shown in Figure 3.10. Table 3.5 lists and compares the detailed power consumption. At the 32 nm node, the static power is dominant and both 3D nFPGA designs have slightly higher total power consumption due to larger static

power from the crossbar array. Results in Table 3.6 show that with a smaller footprint, the dynamic power of nFPGA is reduced because of shorter wire length. However, this reduction margin is reduced by a relatively larger dynamic power from the larger CLB input and BLE output capacitance that is introduced by the crossbar array (Table 3.3). Compared with 3D nFPGA with copper interconnects, 3D nFPGA with CNT bundle interconnects can provide better performance but consumes 17.5% more dynamic power mainly because of high capacitance values of CNT bundles.

Table 3.4 Critical path delay and comparison

| | Baseline FPGA | 3D nFPGA with Copper Wire | | 3D nFPGA | |
|---|---|---|---|---|---|
| | Critical Path(s) | Critical Path (s) | Performance (Fmax) Gain of 3D stacking | Critical Path (s) | Performance (Fmax) Gain of 3D nFPGA |
| alu4 | 7.13E−09 | 3.64E−09 | 1.96 | 2.82E−09 | 2.53 |
| apex2 | 8.60E−09 | 4.38E−09 | 1.97 | 3.31E−09 | 2.60 |
| apex4 | 7.30E−09 | 3.74E−09 | 1.95 | 2.79E−09 | 2.61 |
| bigkey | 4.21E−09 | 1.82E−09 | 2.32 | 1.39E−09 | 3.04 |
| clma | 1.71E−08 | 8.62E−09 | 1.98 | 6.05E−09 | 2.82 |
| des | 7.40E−09 | 3.46E−09 | 2.14 | 2.64E−09 | 2.81 |
| diffeq | 5.56E−09 | 3.24E−09 | 1.71 | 2.99E−09 | 1.86 |
| dsip | 4.23E−09 | 1.95E−09 | 2.17 | 1.50E−09 | 2.83 |
| elliptic | 1.07E−08 | 5.95E−09 | 1.79 | 4.91E−09 | 2.18 |
| ex1010 | 1.46E−08 | 5.94E−09 | 2.46 | 4.44E−09 | 3.29 |
| ex5p | 7.83E−09 | 3.94E−09 | 1.99 | 2.85E−09 | 2.75 |
| frisc | 1.33E−08 | 6.95E−09 | 1.91 | 6.32E−09 | 2.10 |
| misex3 | 7.42E−09 | 3.37E−09 | 2.20 | 2.60E−09 | 2.85 |
| pdc | 1.68E−08 | 7.69E−09 | 2.18 | 5.00E−09 | 3.36 |
| s298 | 1.13E−08 | 6.10E−09 | 1.85 | 5.01E−09 | 2.25 |
| s38417 | 8.82E−09 | 4.10E−09 | 2.15 | 3.48E−09 | 2.54 |
| s38584.1 | 7.21E−09 | 4.04E−09 | 1.78 | 2.78E−09 | 2.60 |
| seq | 8.40E−09 | 3.74E−09 | 2.25 | 2.92E−09 | 2.88 |
| spla | 1.33E−08 | 5.67E−09 | 2.34 | 3.88E−09 | 3.41 |
| tseng | 6.96E−09 | 3.54E−09 | 1.97 | 3.24E−09 | 2.15 |
| Ave. | 9.40E−09 | 4.59E−09 | 2.05 | 3.55E−09 | 2.65 |

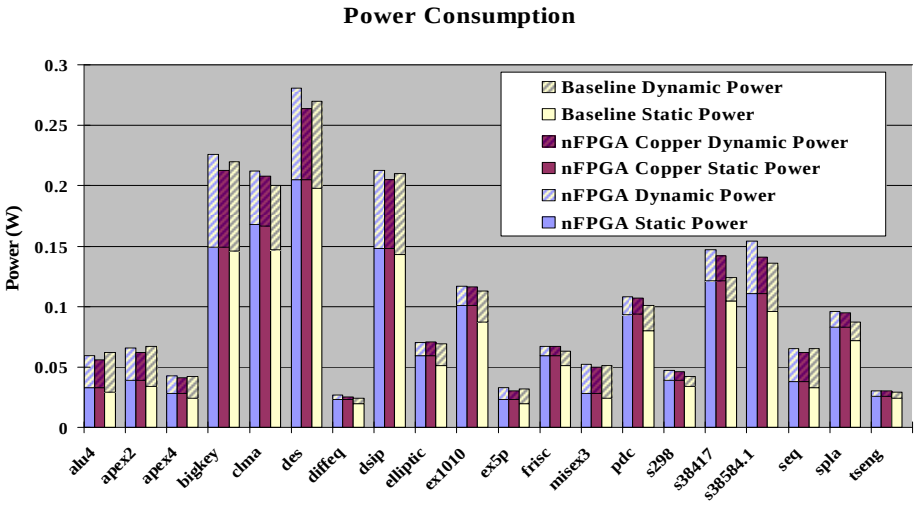



Figure 3.10 Power consumption comparison for three architectures 1st bar in each group is critical path delays of nFPGA, 2nd bar in each group is critical path delays of nFPGA with copper global interconnect, and 3rd bar in each group is critical path delays of baseline FPGA

This section concludes with a comparison of 3D nFPGA and FPNI [55]. FPNI is a 2D hybrid FPGA architecture. It is fair to compare FPNI and 3D nFPGA because both offer experimental results using the same set of benchmarks, compared to the baseline 2D FPGAs (30 nm CMOS-based FPGA for FPNI and 32 nm CMOS-based FPGA for 3D nFPGA). 3D nFPGA is 2.65× faster than the baseline architecture, and FPNI is 30% slower than the baseline. This indicates that nFPGA can outperform FPNI by 3.8× in terms of execution frequency. In terms of area, FPNI could achieve a 7.5× footprint reduction, and nFPGA on the other hand has a 4.5× reduction. The main reason behind this is that FPNI replaces all the routing elements with nanowire crossbars, which significantly reduces the routing area. However, large crossbar arrays will degrade the system performance as well. FPNI also considers power consumption, but it only reports the dynamic power consumed by nanowire arrays. The switching activity is assumed to be 0.1 for simplicity. There is no consideration of clock power and glitch power. In addition, the clock frequency considered in FPNI is 3.8× slower than 3D nFPGA. After normalization with the above factors, 3D nFPGA consumes about the same amount of dynamic power compared to FPNI on average. However, the static power of 3D nFPGA can be much less

compared to FPNI because FPNI uses a large number of crossbar arrays, which introduces a large amount of leakage power due to leaky crosspoints.

Table 3.5 Power consumption and comparison

| | 32nm Baseline | | 3D nFPGA Copper Wire | | 3D nFPGA | |
|---|---|---|---|---|---|---|
| | Total Power (W) | % Static Power | Total Power (W) | % Static Power | Total Power (W) | % Static Power |
| alu4 | 0.062 | 46.20% | 0.0562 | 58.38% | 0.0592 | 55.38% |
| apex2 | 0.067 | 50.13% | 0.0621 | 62.69% | 0.0658 | 59.19% |
| apex4 | 0.042 | 56.61% | 0.0403 | 68.96% | 0.0429 | 64.82% |
| Bigkey | 0.22 | 66.19% | 0.213 | 70.12% | 0.2262 | 66.08% |
| Clma | 0.20 | 73.52% | 0.208 | 80.38% | 0.2120 | 79.03% |
| Des | 0.27 | 73.36% | 0.264 | 77.69% | 0.281 | 73.10% |
| Diffeq | 0.024 | 83.11% | 0.0252 | 92.69% | 0.0275 | 85.00% |
| Dsip | 0.21 | 67.89% | 0.205 | 72.37% | 0.2131 | 69.58% |
| Elliptic | 0.069 | 73.96% | 0.0702 | 83.48% | 0.0696 | 84.29% |
| ex1010 | 0.113 | 77.10% | 0.116 | 86.76% | 0.1171 | 86.33% |
| ex5p | 0.0314 | 63.11% | 0.0305 | 75.66% | 0.0326 | 70.81% |
| Frisc | 0.0627 | 81.08% | 0.0672 | 88.02% | 0.0668 | 88.50% |
| misex3 | 0.0513 | 46.72% | 0.0499 | 55.91% | 0.0514 | 54.27% |
| Pdc | 0.101 | 78.72% | 0.107 | 87.59% | 0.1073 | 86.41% |
| s298 | 0.042 | 80.07% | 0.0461 | 85.39% | 0.0473 | 83.32% |
| s38417 | 0.124 | 84.45% | 0.142 | 85.03% | 0.1466 | 82.41% |
| s38584.1 | 0.136 | 70.53% | 0.141 | 79.02% | 0.1543 | 72.25% |
| Seq | 0.065 | 51.10% | 0.0620 | 61.67% | 0.0656 | 58.29% |
| Spla | 0.087 | 82.62% | 0.0954 | 87.06% | 0.0961 | 86.39% |
| Tseng | 0.029 | 83.23% | 0.0301 | 87.86% | 0.030 | 88.20% |
| Ave. | 0.100 | 69.5% | 0.102 | 77.3% | 0.106 | 74.7% |

Table 3.6 Dynamic power reduction of nFPGA architecture

| | 32nm Baseline (W) | 3D nFPGA Copper Wire (W) | 3D nFPGA (W) | Baseline / 3D nFPGA Copper Wire | Baseline / 3D nFPGA |
|---|---|---|---|---|---|
| Ave. Dynamic Power | 0.0295 | 0.0228 | 0.0268 | 1.294 | 1.10 |

# CHAPTER 4
# FPCNA：CARBON NANOTUBE-BASED PROGRAMMABLE ARCHITECTURE

This chapter proposes a new CNT-based FPGA architecture called FPCNA (field programmable carbon nanotube array). The building blocks of FPCNA have been described in detail, including the carbon nanotube lookup table, which makes up its programmable logic. Also described is a high-density routing architecture using a recently proposed nanoswitch device. Special considerations are made to mitigate the negative effects of nano-specific process variations. These components are described considering these variations, as well as circuit-level delay variations.

The performance of the proposed architecture is evaluated by adopting a typical FPGA design flow and developing variation-aware placement and routing algorithms. These algorithms are enhanced from the popular physical design tool VPR [62], and use statistical timing analysis (SSTA) to improve the performance yield. SSTA with both normal and non-Gaussian variation models is performed. The results show that FPCNA offers significant performance and density gains compared to the conventional CMOS FPGA, demonstrating potential for the use of CNT devices in next-generation FPGA circuits.

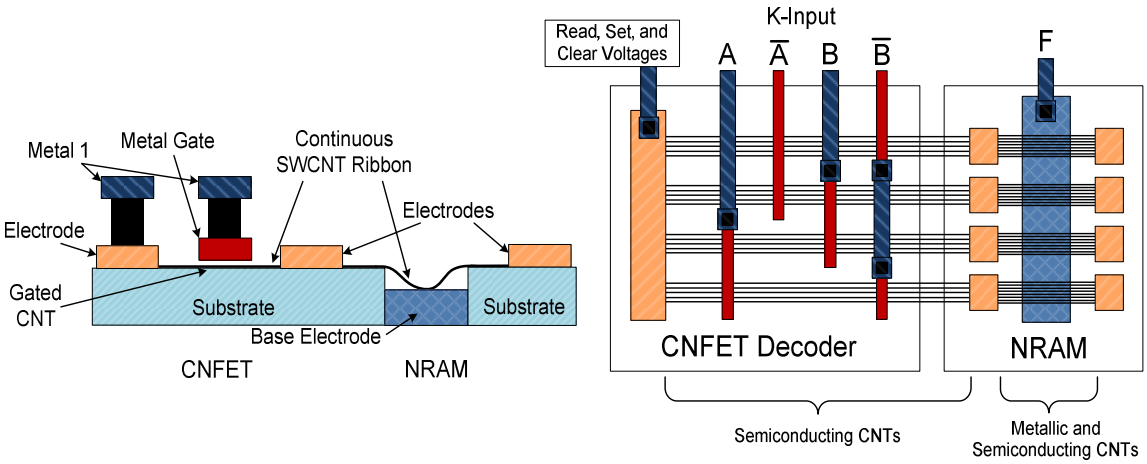
## 4.1 FPCNA Architecture

In this section the FPCNA architecture is described in detail, beginning with the introduction of the LUT design, which is based on CNT devices. Then a Basic Logic Element (BLE) that can be created using this LUT design is presented. Finally, I discuss FPCNA's high level architecture, including the design of local and global routing.

### 4.1.1 CNT-Based LUT

A *K*-input lookup table (*K* LUT) is the basic unit of programmable logic in modern FPGAs. For FPCNA, a novel *K*-LUT design is used that is based entirely on CNT devices. Profile and overhead views of this device are shown in Figure 4.1(a)-(b). This design uses parallel ribbons of SWCNTs held in place by metal electrodes and crossed by metal gates. PMOS CNFET devices are formed at the crossing points of the CNT ribbons and the metal gates, creating a CNFET decoder. At points where the CNT ribbons pass over a trench in the substrate, NRAM memory devices are formed. This CNT memory is used to store the truth table of the BLE's logic function. By applying *K* inputs to the decoder, a reading voltage will be sent to the corresponding memory bit whose output can then be read from the base electrode.

One of the key innovations of this LUT design is that it builds the decoding and memory on the same continuous CNT ribbons. This structure allows for high logic density and simplifies the manufacturing process. For comparison, the work in [69] uses an LUT memory based on individually-crossed nanotubes that is addressed by a CMOS multiplexor tree. In addition to being more costly in area, this design suffers from fabrication issues because it requires the alignment and interfacing of individual nanotubes in two dimensions.




(a) (b)

Figure 4.1 (a) Cross section of CNT-based LUT; (b) top view of CNT-based LUT

By using CNT ribbons, each device will contain multiple tubes. This adds fault tolerance from the high defect rates of nanotube fabrication, and increases the chance that a CNFET or NRAM device will contain functioning nanotubes. Thus, the design is more reliable than in [69] where a device will fail if either of the two nanotubes is defective.

### 4.1.2 BLE Design

In Figure 4.1, a 2-to-4 (2 input, 4 NRAM cell) LUT was shown for illustration purposes. In modern FPGAs, each basic logic element (BLE) typically contains a 4-to-16 LUT, as well as a flip-flop (FF) and multiplexor (MUX) to allow registered output. When scaled to *K* inputs, the LUT will contain $2^K$ CNT ribbons. The BLE design used for FPCNA is shown in Figure 4.2. In this figure, the LUT is expanded to four inputs and supporting CMOS logic for voltage control, address line inversion, and registered output are added.

In the decoder, Gray address decoding is used to minimize the number of gate-to-metal-1 transitions. Compared to binary decoding, this reduces the number of vias by 46% (from 48 to 26). Since the LUT depends on both normal and complemented inputs, inverters are added for each of the address line inputs. A buffer is used to restore the output signal before it passes to the flip-flop and MUX.

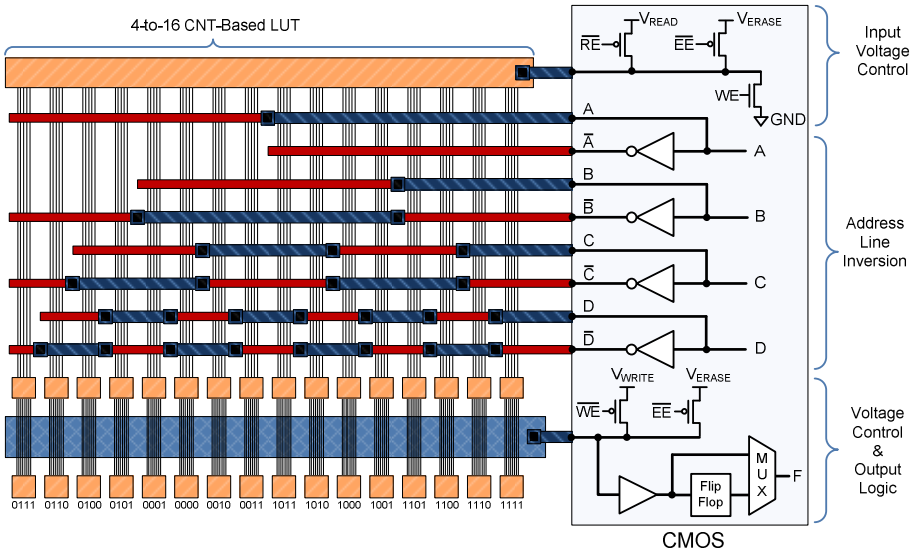



Figure 4.2 FPCNA BLE with a 4-to-16 CNT-based LUT

Table 4.1 NRAM operating modes

| ***Mode*** | ***RE*** | ***WE*** | ***EE*** | ***Ribbon Voltage*** | ***Base Electrode*** |
|---|---|---|---|---|---|
| Reading | High | Low | Low | $V_{READ}$ (1 V) | Output |
| Writing | Low | High | Low | Ground (0 V) | $V_{WRITE}$ (1.6 V) |
| Erasing | Low | Low | High | $V_{ERASE}$ (+2.5 V) | $V_{ERASE}$ (+2.5 V) |

To read and program NRAM, three different voltage configurations are needed. CMOS pass transistors are used to configure the three modes. Table 4.1 shows the pass transistor enable signals and voltages during each mode. Most often, the circuit will be in the reading mode with the RE (read enable) signal set. This allows $V_{READ}$ to pass through the decoder and select the appropriate NRAM bit. If the NRAM bit is set, the signal will pass through the relatively low resistance of the nanotubes contacting the base electrode (logical 1). If the NRAM bit is not set, a multiple GΩ resistance will prevent transmission (logical 0).

To program a value, RE is deactivated, and either WE (write enable) or EE (erase enable) is activated. When WE is set, the selected CNT ribbon is grounded, and $V_{WRITE}$ is applied to the base electrode. The difference in potential creates an attractive force, which pulls the ribbon down into the trench. In [36], Nantero measured a threshold voltage of 1.4 V +/− 0.2 V, so a $V_{WRITE}$ of 1.6 V is assumed. When erasing, the same voltage ($V_{ERASE}$) is applied to both the ribbon and the base electrode. The like voltages repel each other, releasing the ribbon from the trench floor and allowing it to return to an unbent state. $V_{ERASE}$ must be somewhat larger than $V_{WRITE}$ [35]-[36], so a value of +2.5 V is assumed. There is a risk that this voltage applied to the base electrode could attract an unselected ribbon, causing an unintentional write. This can be avoided by erasing all of the NRAM bits. As each bit is erased, its ribbon will be temporarily charged to +2.5 V, repelling it from the electrode during the erasure of the remaining bits. Then the individual bits that need to be set as logic 1 can be written to realize the new configuration.

### 4.1.3 Logic Block Design

For FPCNA, a cluster-based configurable logic block design is used. Each configurable logic block (CLB) contains *N* of the BLEs described in the previous section (where *N* is the

cluster size), as well as the local routing used to connect the BLEs together. In conventional CMOS FPGA designs, the routing is often multiplexor-based. While the same approach could be adopted, using CMOS for the MUXs and NRAM to store multiplexor configuration bits, a greater logic density can be achieved by using solid-electrolyte switch crossbars.

Figure 4.3 shows a simplified CLB design to illustrate this technique. The CLB in this figure contains four BLEs made from CNT-based LUTs. The local routing is created with solid-electrolyte switches created at the crosspoints of the vertical and horizontal routing wires. By programming the nanoswitch points, a BLE output can be routed to any BLE input. In Figure 4.3, one of the input signals to BLE 1 is identified with a dashed line labeled “Input to BLE”. The black dots at crosspoints indicate that solid-electrolyte switches at those locations are turned on. By using more switches, the same signal can be routed to multiple BLE inputs. Output from a BLE can connect to the inputs of other BLEs or be outputs from the CLB. Note that Figure 4.3 shows the local routing positioned between BLEs for clarity. In an actual implementation, the local routing and routing switches can be made above the BLEs, and the area calculations reflect this.

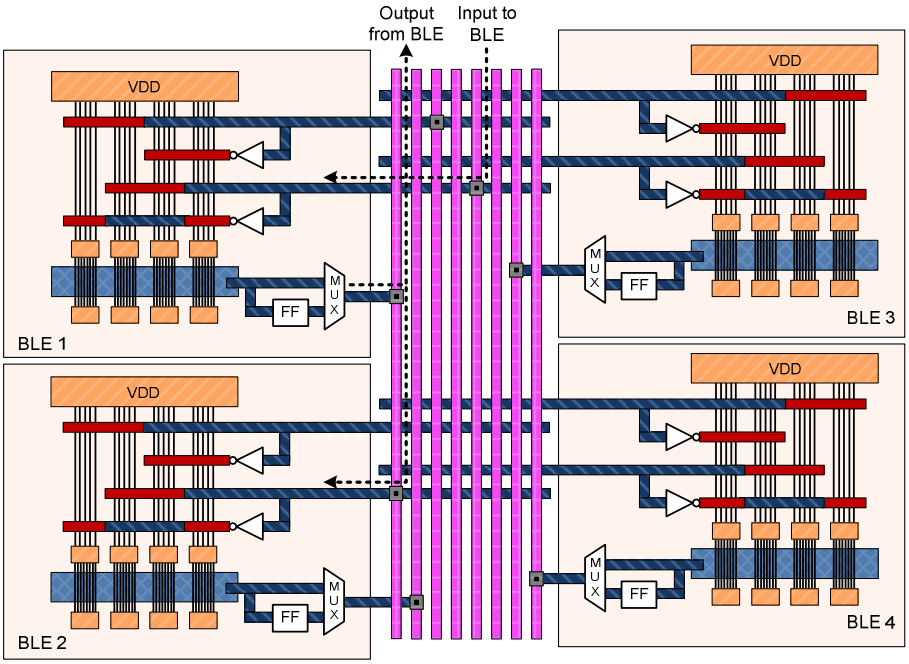



Figure 4.3 CNT-based CLB with nanoswitch local routing

### 4.1.4 High-Level Architecture and Global Routing

A conventional island-based FPGA architecture is adopted for the high-level organization of FPCNA. The basic structural unit is a tile, consisting of one programmable switch block (SB), two connection blocks (CB), and one configurable logic block (CLB). This tile is replicated to create the FPGA fabric, as shown in Figure 4.4.

The global routing structure consists of 2D segmented interconnects connected through programmable SBs and CBs. The CLBs are given access to these channels through connections in the CBs. The parameter *I* represents the number of inputs to a CLB, and *Fc* defines the number of routing tracks a CLB input can connect to. CNT bundle interconnects are used for global routing because they have been shown to be superior to copper in terms of current density and delay [37].

In a traditional CMOS-based FPGA, the SBs and CBs take up the majority of the overall area [46]. For example, if the CLB size is 10 and the BLE size is 4 (popular parameters for commercial FPGA products), the global routing takes 57.4% of the area, with the CLBs occupying the remaining 42.6% [46]. To reduce the size of the global routing in FPCNA, the traditional CB is replaced with a solid-electrolyte switch crossbar, and a new nanoswitch-based SB design is used.

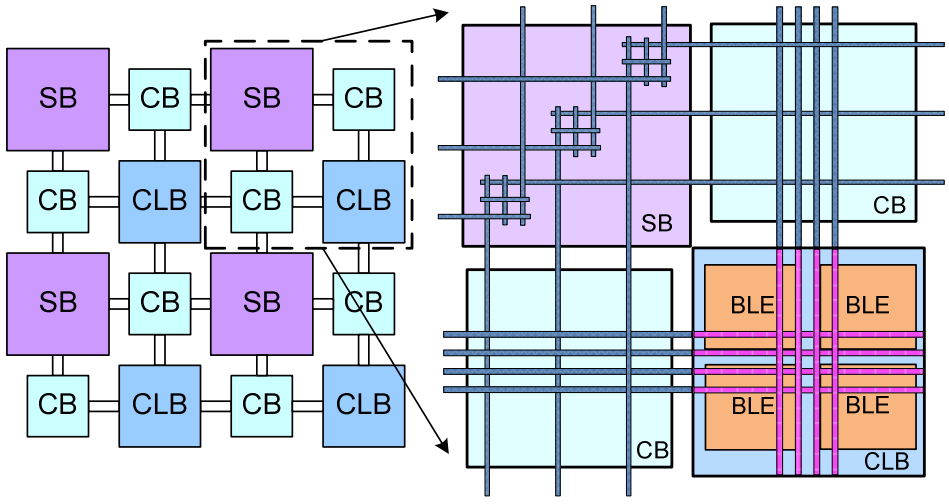
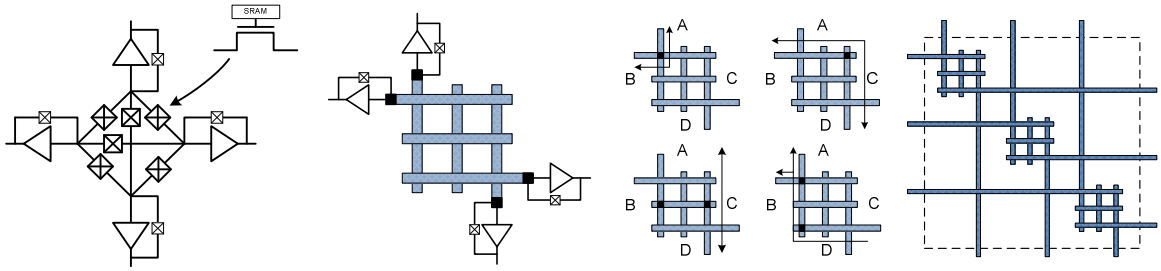



Figure 4.4 High-level layout of FPCNA

The new SB design is shown in Figure 4.5. Instead of using six SRAM-controlled pass transistors for each switch point as in conventional CMOS designs (Figure 4.5 (a) [62]), six perpendicular wire segments are used with solid-electrolyte nanoswitches at the crosspoints. In this design, the driving buffers and input control pass transistors are kept in CMOS, as shown in Figure 4.5 (b). By programming nanoswitches at the crosspoints of the wire segment array, a signal coming from one side of the block can be routed to any or all of the other three sides. To demonstrate how routing connections can be made, four switching scenarios are illustrated in Figure 4.5 (c). In the figure, arrows represent signal directions and black dots indicate the activated switches. The upper left scenario shows how signals A and B are connected using a single switch. A multipath connection is demonstrated in the lower right scenario, where a signal from C is driving both A and B. By turning on the appropriate nanoswitches, any connection of signals can be made. Using these switch points, larger switch blocks can be constructed. For example, the 3 × 3 universal-style switch block in Figure 4.5 (d) is made from three nanoswitch-based switch points. This design can be scaled to any routing channel width, and significantly reduces the SB area. In a conventional CMOS switch point (Figure 4.5 (a), center), six 10× pass transistors are controlled by six SRAM cells, which normally requires an area of 88.2*T* (where *T* is the area of a minimum-size transistor). When using nanoswitch-based switch points, the same routing function can be achieved in approximately 9*T*.




(a) (b) (c) (d)

Figure 4.5 (a) CMOS switch point; (b) nanoswitch-based switch point with CMOS driving buffers; (c) example switching scenarios; and (d) 3 × 3 switch block (driving buffers not shown)

## 4.2 Nanotube Lookup Table Fabrication

Recent progress in the fabrication of CNTs has enabled the use of CNT-based structures in FPCNA's LUT design. To demonstrate the feasibility of this design at the 32 nm technology node, some of the fabrication issues involved are addressed.

The first step in manufacturing the LUT is to define the NRAM trench in the silicon wafer using a process similar to the one described in [36]. Then the nanotubes are grown on separate quartz wafers using chemical vapor deposition. Since the desired CNT ribbons are all aligned in the same direction, an array-based CNT growth process can be used. In [17], researchers report a technique for fabricating dense, perfectly aligned arrays of CNTs using photolithographically defined catalytic seeds, which achieves an alignment of up to 99.9%. The aligned nanotubes can then be transferred to a silicon wafer using a stamping process like the one developed in [70]. These techniques create nanotubes that are suitable for the transistors and NEMS devices used in the LUT. In addition, it is possible to improve nanotube density on the silicon wafer by performing multiple consecutive transfers. This analysis assumes a multiple transfer process is used that provides a CNT pitch of 4 nm.

After the nanotubes have been transferred to the substrate, parallel ribbons are then made from the continuous nanotube array by using an etching process similar to the one used in [71]. The distance between ribbons is set to 96 nm to allow spacing for contacts, and this resolution is assumed to be achievable in the target process technology. By using etching to define the ribbons, there is an added advantage of making the ribbons immune to misalignment. This is because any nanotubes crossing the border of a ribbon will be removed during the etching process. Figure 4.6 demonstrates this concept, where (a) shows a misaligned tube, (b) shows the etched area, and (c) shows the resulting CNT ribbons.

The next major step in fabrication is to disable the metallic nanotubes inside the decoder region. Since metallic CNTs act as a short between source and drain, they need to be removed to create CNFET transistors with desirable on-off current ratios. Electrical burning [72] is an effective method to selectively disable the metallic CNTs. In this technique, a large voltage is

applied across the array which heats the conducting metallic nanotubes to a breakdown temperature of ~600ºC and causes irreversible oxidization. Because this is done when the CNTs are still exposed to air, a minimum power dissipation of 0.05 mW is needed to achieve breakdown [72].

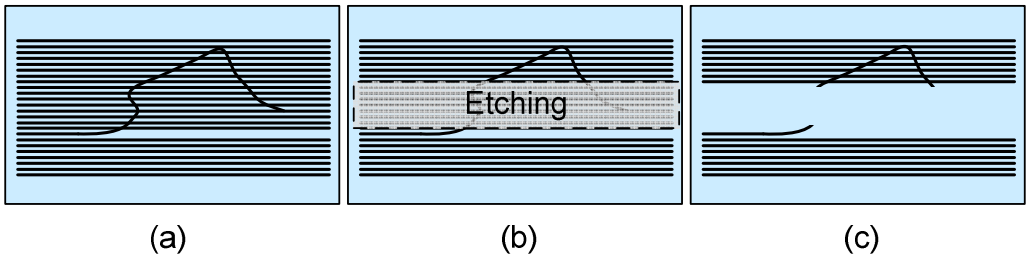



Figure 4.6 CNT ribbon etching: (a) original CNT array with misaligned CNT; (b) defined etching region; and (c) CNT array with misaligned CNT removed

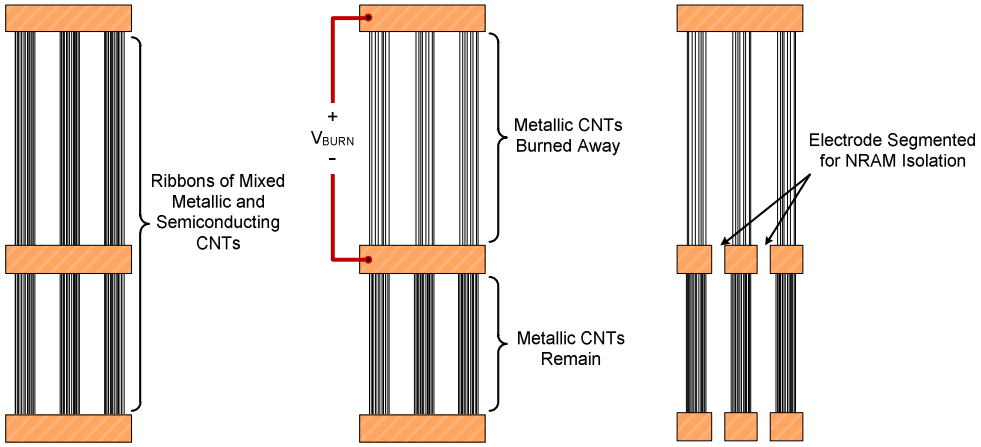



Figure 4.7 Process of metallic CNT removal: (a) CNT ribbons before processing; (b) metallic CNTs are removed through electrical burning; and (c) contacts are defined by lithography

Since metallic tubes are used for NRAM operation, the burning must only be done in the decoder region. One way to remove the metallic CNTs from the decoder but keep them for the NRAM devices is shown in Figure 4.7. In this figure, (a) shows vertical ribbons of mixed metallic and semiconducting CNTs held in place by horizontal metal electrodes. The middle and bottom electrodes are used to hold the ribbons in place during NRAM operation. In (b), a thermal

breakdown voltage, $V_{BURN}$, is applied between the top and middle electrodes. This burns away the metallic tubes in the decoder region but leaves them in the NRAM region. Because the NRAM memory devices need to be individually addressable, the electrode is segmented to provide electrical isolation (c).

After the CNT ribbons are defined and processed, the gate and source/drain formation is similar to a regular CMOS process. Based on these techniques and the existing CNT fabrication work [17], [71], [73], the proposed nanotube-based LUT design is believed to be implementable.

## 4.3 Circuit Characterization

### 4.3.1 CNFET and CNT-Based LUT Variation

As mentioned in Chapter 2, CNFETs have many properties that make them attractive for use in future electrical circuits. Ideally, the channel region of these CNFETs would consist of identical, well-aligned semiconducting CNTs with the same source/drain doping levels. However, it is difficult to synthesize nanotubes with exactly controlled chirality using known fabrication techniques. HiPco synthesis techniques yield around 50% ± 10% metallic CNTs [74]. This means the number of semiconducting CNTs per device is stochastic, causing drive current variations even after the metallic CNTs are burned away. Meanwhile, CNFETs are also susceptible to variations in diameter and source/drain region doping [21].

In a traditional MOSFET, Gaussian distributions are often assumed when modeling variation sources such as channel length and gate width. These models are then used in the delay or power characterization of the MOSFET. A similar approach can be used to characterize CNFETs. To quantify the effects of CNFET variations, a Monte Carlo simulation of CNFET devices with 2,000 runs is performed. The sources of variation that are considered are listed in Table 4.2, with two scenarios for the number of CNTs in a channel: 8 ± 3, and 6 ± 2, both normally distributed. The diameter range, doping level range, and CNFET model are suggested in [21].

Table 4.2 Sources of CNFET variation

| ***Parameter*** | ***Mean*** | ***Variation*** |
|---|---|---|
| CNTs per channel case | 8 | ± 3 |
| CNTs per channel case | 6 | ± 2 |
| CNT diameter | 1.5 nm | ± 0.3 nm |
| Doping level | 0.6 ev | ± 0.03 ev |

The results of the simulation show that the delay distribution of a CNFET device under these variations fits the Gaussian distribution. Figure 4.8 illustrates this distribution for a CNFET with 8 ± 3 semiconducting nanotubes in its channel.

Using the CNFET model, the performance of the CNT-based LUT design can also be evaluated. The LUT decoder consists of multiple stages of p-type CNFETs, simulated under the variations mentioned in Table 4.2. The contact resistance between an electrode and a single nanotube is assumed to be 20 kΩ based on [37]. In a ribbon, multiple CNTs are operating in parallel, so the ribbon contact resistance is considered to be inversely proportional to the number of semiconducting nanotubes. For NRAM devices, a contact resistance between a bending nanotube and the base electrode of 20 kΩ is assumed, based on the measurements in [75]. Since these CNTs also operate in parallel, the total ribbon NRAM contact resistance is treated as inversely proportional to the number of metallic nanotubes in the ribbon. The resulting LUT delay distribution generated by Monte Carlo simulation in HSPICE is shown in Figure 4.9.

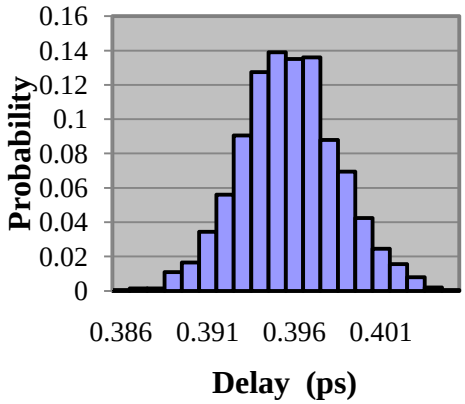



Figure 4.8 Delay distribution of a CNFET under process variation

The average delay is 60.94 ps, which is 41% faster than a traditional 32 nm CMOS LUT, which has a delay of 103.8 ps. Unlike the CMOS LUT, the delay of the nanotube-based LUT has a distribution similar to log-normal.

### 4.3.2 Crossbar Characterization

As described earlier, the routing in FPCNA is implemented using crossbars. The delay and variation of these crossbars is captured using HSPICE. CNT bundle interconnect is assumed to be 32 nm in width, with an aspect ratio of 2. The dielectric constant of the insulation material around the crossbar is set at 2.5, and a unit resistance of 10.742 Ω/μm and capacitance of 359.078 aF/μm for the carbon nanotube bundles is derived. The interconnects are evaluated for 10% geometrical variation of wire width, wire thickness, and spacing according to [76]. CNT bundle interconnect variation also considers a 40% ~ 60% range on percentage of metallic nanotubes inside a bundle. The solid-electrolyte switches between interconnect layers are considered with a 100 Ω on resistance [43] and 10% variation to capture via contact resistance.

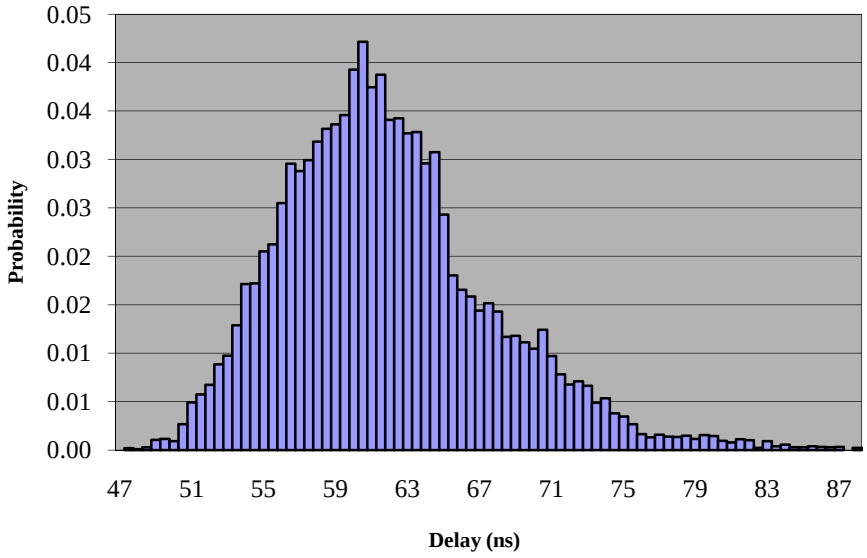



Figure 4.9 CNT-based LUT delay considering variation

Table 4.3 Delay comparison between baseline CMOS and FPCNA

| ***Paths*** | ***CMOS-Baseline*** | | ***FPCNA*** | |
|---|---|---|---|---|
| | *μ (ps)* | *σ (ps)* | *μ (ps)* | *σ (ps)* |
| A→B | 141.66 | 7.13 | 42.24 | 2.48 |
| B →C | 107.59 | 5.37 | 30.45 | 2.21 |
| D →C | 107.59 | 5.37 | 49.96 | 2.92 |
| D →Out | 28.48 | 1.22 | 29.91 | 2.28 |

### 4.3.3 Timing Block Evaluation

To support the evaluation CAD flow, various circuit models are needed to capture characteristics of the FPCNA architecture as in the previous chapter. The delay and variation of these paths in FPCNA are computed by performing a Monte Carlo simulation of 1,000 runs, varying the CNFET parameters and CNT contact resistance for each run. Figure 4.10 illustrates the resulting delay distributions of wire track to CLB inpin connections (A→B) and subblock opin to subblock inpin connections (D→C).

Based on these results, the timing blocks follow a normal distribution. Therefore, the mean (μ) and variation (σ) of each delay path can be calculated, as shown in Table 4.3. An equivalent design in CMOS is measured as a baseline for comparison, assuming 12% channel width variation, 8% gate dielectric thickness variation, and 10% doping variation (values from [1] for 32 nm CMOS), and these values are also shown in the table.

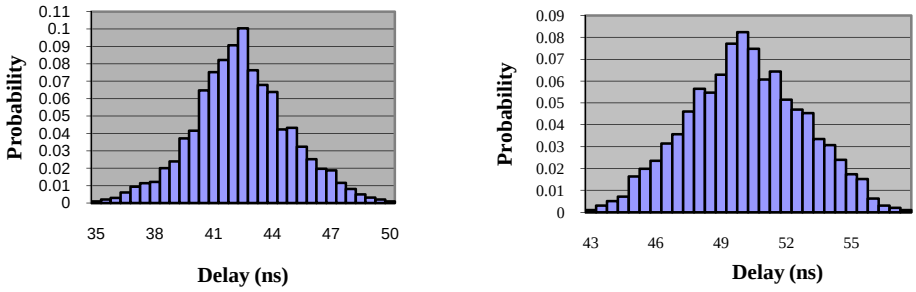



Figure 4.10 Delay distribution of wire track to CLB inpin (left) and sub-block opin to sub-block inpin (right)

## 4.4 CAD Flow

In the Previous sections showed that both deep sub-micron CMOS and nanoscale devices are susceptible to variation. In traditional static timing analysis, it is assumed that all circuit elements have deterministic delay. This approach cannot correctly capture the variability of the fabrication process. The worst-case analysis commonly used by industrial designs satisfies yield but is overly pessimistic. On the other hand, the nominal case produces low yield due to variation-based timing failures. To maximize yield without sacrificing performance, it is necessary for CAD tools to consider the statistical information of circuit elements during timing analysis.

In this work, a timing-driven, variation-aware CAD flow is used, as shown in Figure 4.11. Each benchmark circuit goes through technology-independent logic optimization using SIS [63] and is technology-mapped to LUT with size 4 using DAOmap [64]. The mapped netlist then feeds into T-VPACK and VPR [62], which perform timing-driven packing (i.e., clustering LUTs into the CLBs), placement, and routing. To take variation into consideration, the VPR tool [62] is enhanced to make it variation-aware.

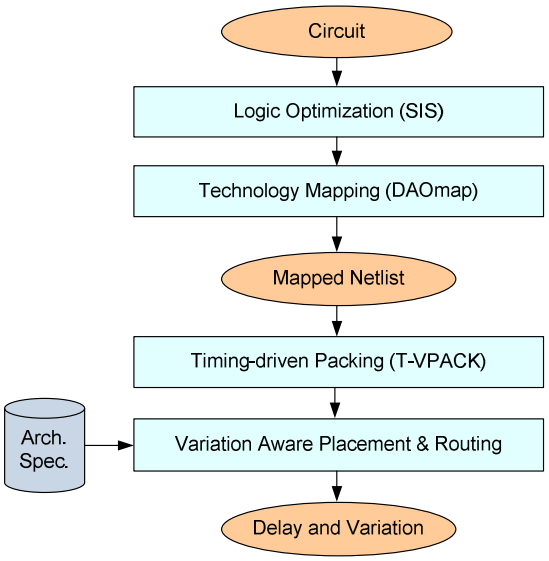



Figure 4.11 FPCNA evaluation flow

Existing works have shown that statistical optimization techniques are useful during the physical design stage. Variation aware placement is implemented in [77] and variation aware routing is developed in [78]. Based on the ideas presented in these works, a complete variation aware physical design flow is implemented. In this holistic solution, the placer calls the variation aware router to generate delay estimates for its timing cost calculations.

The Monte Carlo simulation results in Section 4.3.2 show that the CNT-based LUT delay follows a non-Gaussian distribution. Reference [37] also reports a non-Gaussian distribution for CNT bundle interconnect. However, all of the existing CAD work targeting CMOS assumes normally distributed random variables [77]-[79]. The Gaussian-based SSTA algorithms that these works use to evaluate CMOS are not suitable for modeling the non-normal variables of molecular-based architectures. Therefore, a statistical timing analyzer is utilized that can handle an arbitrary distribution, based on discretization techniques adapted from [80]-[81].

One such technique is the probabilistic event propagation developed in [80], in which discretized random variables of cell delays are used for timing analysis. As illustrated in Figure 4.12, a non-Gaussian probability density function can be represented as a set of delay-probability pairs that contain the time *t* and the probability a signal will arrive at time *t*. In [81], ADD and MIN operations are developed for propagating multiple event groups. These operations are used, and a MAX operation is defined for use in the statistical timing analyzer. Figure 4.13 shows how the discretized MAX operation is performed using an example point.

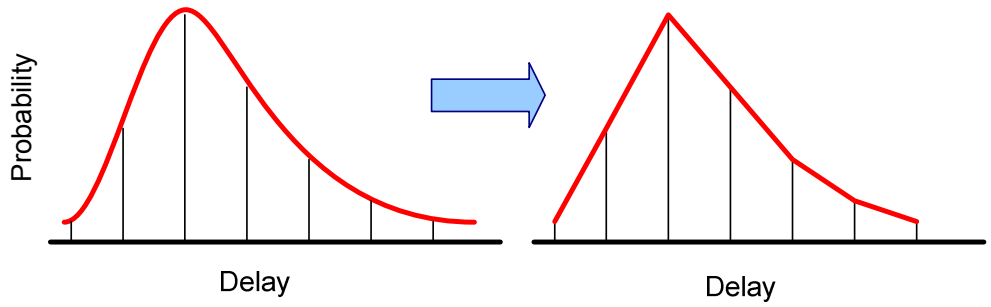



Figure 4.12 Discretization process of a log-normal probability density function

During the MAX operation, all possible timing points at the output are evaluated,

computing their probability based on the input sets of delay-probability pairs. For each timing point *t*, the probability that both inputs arrive is defined as *P(t)*. *P(t)* can be derived using conditional probability as the sum of :

1. The probability that both A and B arrive at *t*
2. The probability that A arrives at *t* and B arrived before *t*
3. The probability that B arrives at *t* and A arrived before *t*

The accuracy of this technique is dependent on the number of points used for piecewise linear approximation. It is shown in [80] that 7 points are sufficient to obtain an accuracy of less than 1% error compared to Monte Carlo. Therefore, a 7-point sampling is used throughout the discretized SSTA.

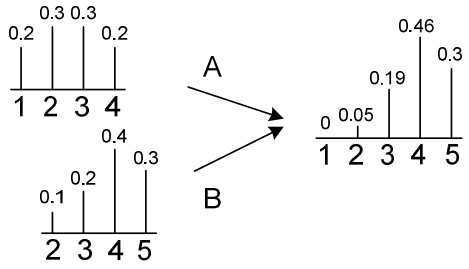



Figure 4.13 The discretized MAX operation

Figure 4.14 shows the pseudo-code of the variation aware router. The routing is iterative. During the first iteration, the criticality of each pin in every net is set to 1 (highest criticality) to minimize the delay of each pin. For the CMOS architecture, the Gaussian delay mean (μ) and standard deviation (σ) of each path are computed during the routing of each net. For FPCNA, the discretized delay distribution of each path is computed. If congestion exists, more routing iterations are performed until all of the overused routing resources are resolved. At the end of each routing iteration, criticality and congestion information are updated before the next iteration starts.

To consider variation, new formulas to capture the criticality of sink *j* of net *i* are derived. For the CMOS architecture with a Gaussian distribution, the arrival time of pin *j* in net *i* is expressed as $arr(i, j) = (t_a, \sigma_a)$ and the required time as $req(i, j) = (t_r, \sigma_r)$. The mean and

standard deviation of slack $slack(i, j) = (t_s, \sigma_s)$ can be derived as

$$t_s = t_r - t_a \tag{4.1}$$

$$\sigma_s = \sqrt{\left|\sigma_a^2 + \sigma_r^2\right|} \tag{4.2}$$

The criticality of pin *j* in net *i* can then be computed by taking both slack and slack variation into consideration:

$$Crit(i, j) = 1 - \frac{t_s - 3\sigma_s(i, j)}{t_{crit} + 3\sigma_{crit}} \tag{4.3}$$

```
Set all nets i and sinks j, Crit(i,j) =1.0;
while (overused routing resources exist) do
    for (each net, i) do
        Rip-up routing tree of net i;
        for (each sink j of net i in decreasing Crit(i,j) order) do
            Find the least cost route of sink j;
            for (all nodes in the path from i to j) do
                Update congestion;
            end
            Update delay and standard deviation of the route;
            (Update discretized delay for FPCNA;)
        end
    end
    Update historic congestion;
    Compute mean and standard deviation of delay for each net(i);
    (Compute discretized delay for each net(i) in FPCNA;)
    Update Crit(i,j);
end
```

Figure 4.14 Pseudo-code of the modified VPR router

The original VPR cost function is modified this way so that when two slacks have similar means but different variations, the $t_s - 3\sigma_s(i, j)$ term assigns a larger criticality to the path with the greater variation to weight it more heavily in the next routing iteration. This is illustrated in Figure 4.15, where the distribution with slack variation $\sigma_1$ will be assigned a higher criticality than the distribution with slack variation $\sigma_2$, even though they have the same mean. This cost function also considers the critical path variation with the $t_{crit} + 3\sigma_{crit}$ term.

In the discretized routing, the expected values of the slack and critical path discretized

points are computed and used in the following criticality function:

$$Disc\_Crit(i,j) = 1 - \frac{E[disc\_slack(i,j)]}{E[disc\_t_{crit}]} \quad (4.4)$$

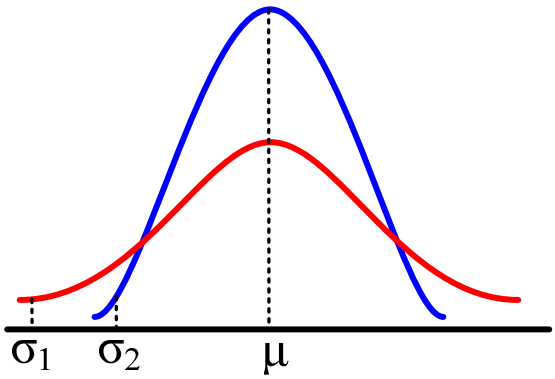



Figure 4.15 Criticality estimation

After each routing iteration, SSTA is executed by traversing the updated timing graph to calculate the new slack and critical path delay. The variation-aware placer also uses these criticality functions to calculate the timing cost of each move during simulated annealing. In the placement cost function, the critically value is raised by the exponent β. The optimal value of β is determined to be six for this design. This differs from the original VPR method of incrementing β from 1 to 8, and from [77] where a β value of 0.3 was used. As in [77], the variation is calculated during the delta array creation, and these pre-calculated values are also stored in the delta arrays for use in placement. The main difference is that a variation aware router is used to generate the delay and store sets of discretized delay-probability points for each delay value in addition to the mean and variation.

## 4.5 Experimental Results

### 4.5.1 Experimental Setup

Because this CAD flow is flexible, one can experiment with various architecture parameters and determine their impact. To evaluate FPCNA, a fixed LUT input size $K$ = 4 was used, and logic cluster sizes of $N$ = 4, 10, and 20 were explored, as well as the difference between using an average of 16 CNTs per ribbon and 12 CNTs per ribbon, to see the effect on area. The

number of CLB inputs is set based on the cluster size so that it equals $2N + 2$. *Fc* is kept at 0.5, a typical value which connects the CLB input to half of the routing tracks in the channel.

It is shown in [62] that a mixture of different length interconnects can provide improved performance. Two popular wire length mixtures are evaluated: an equal mixture of length-4 and length-8 wire segments (wires crossing either four CLBs or eight CLBs), and a mix of 30% length-1, 40% length-2, and 30% length-4 wire segments.

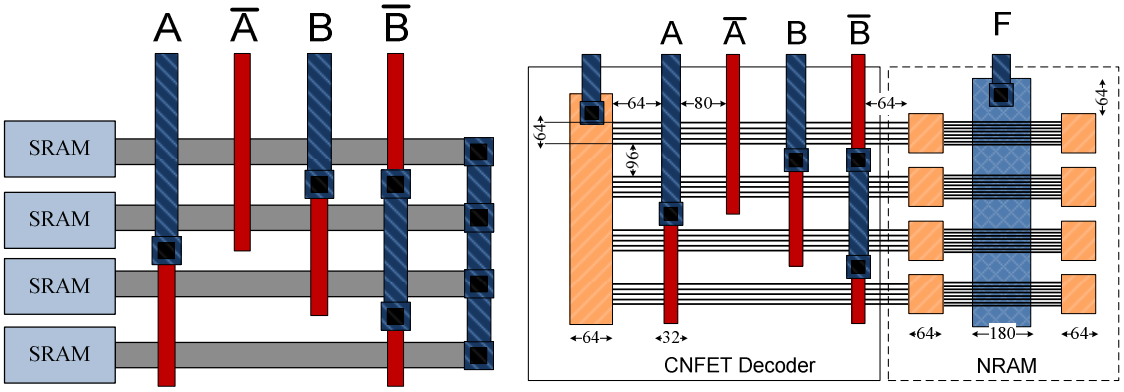
For each configuration of the above parameters, a binary search is performed to determine the routing channel width needed to successfully route the largest benchmark, and then that width is used to evaluate all of the benchmarks.

### 4.5.2 Area Reduction

Due to the high-density CNT-based logic and solid-electrolyte switch-based routing, the footprint of FPCNA is significantly smaller than the equivalent CMOS FPGA. To calculate the area, the architecture parameters defined above are used, and a transistor feature size of 32 nm (2λ) is assumed for both CNT- and CMOS-based transistors. The area of the CNT-based LUT, is determined by the size and spacing of the CNT-ribbons and addressing lines. Since an average CNT pitch of 4 nm is assumed, the CNT ribbons are 64 nm wide for the 16-tube-per-ribbon experiments and 48 nm wide for the 12-tube-per-ribbon experiments.

To accommodate vias between the gate layer and the metal-1 layer, the nanotube ribbons are spaced 96 nm (6λ) apart. LUT addressing gate metal is 32 nm (2λ), with a spacing of 80 nm (5λ) between adjacent lines. Gate-to-metal-1 via size is assumed to be 64nm (4λ) square. The nanotube memory, NRAM, offers a much smaller area than an SRAM cell. The NRAM trench is assumed to be 180 nm in width and 18 nm in height. These dimensions are conservative estimates based on fabrication results in [35]. Trench-to-electrode spacing is set to 90 nm for each side. All of the LUT electrodes are assumed to be 64 nm wide. The area of the 32 nm CMOS components in the BLE are calculated using a technique from [62] by counting minimum-width transistor area. In the FPCNA design, each BLE contains CMOS components

including four size-2 buffers, one multiplexer, and one Flip-flop, as shown in Figure 4.3. The total BLE logic area is the sum of both the CMOS logic area and CNT-based LUT area. Figure 4.16 shows design details for both CMOS and FPCNA LUT cells. For simplicity, only a 2-input LUT is shown.




(a) (b)

Figure 4.16 Design details for the LUT cells: (a) CMOS LUT; and (b) FPCNA LUT (for the 16-tube-per-ribbon case)

Local CLB interconnect crossbars are assumed to have a line thickness of 64 nm (4λ), and spacing of 64 nm (4λ). The routing crossbars are created on the metal layers above the CLB logic, so they do not add to the overall CLB area (assuming the crossbar area is smaller than the logic area, which was true in all of the experiments). Since the routing path is controlled by non-volatile solid-electrolyte switches, the SRAM cells used in the baseline CMOS FPGA can be eliminated in FPCNA. By replacing the MUX-based routing with crossbars and switching to CNT-based LUTs, a large overall area reduction is seen. For an architecture with a cluster size of 10 and wire segmentation of length 4 and 8, the footprint of a baseline CMOS FPGA tile is estimated to be 34,623 *T*. Using a minimum width transistor area of $T = 0.0451\ \mu m^2$ for a 32 nm transistor gives us a tile area of 1561.5 $\mu m^2$. When calculating the area for an equivalent FPCNA tile under the area assumptions above, only 307.99$\mu m^2$ is used. These calculations show that FPCNA can achieve an area reduction of roughly 5× over CMOS.

The area breakdowns for a single LUT and an architecture tile are shown in Table 4.4. In this table the CB area is the sum of both connection blocks. Table 4.5 shows the area breakdowns

of various FPCNA architectures. The first row gives data for global routing with 30% length-1, 40% length-2, and 30% length-4 wire segments. The second row is for 50% length-4, and 50% length-8 interconnects. As seen in the table, routing occupies the majority of FPCNA's overall area. Due to the size of the SB area, wire segmentation has a significant impact on the overall area. Shorter wire segments have better flexibility during routing, but require a larger number of switch points in each SB, which greatly increases the tile size.

Table 4.4 Area reduction of FPCNA

| | *CMOS FPGA* | *FPCNA* | *Reduction* |
|---|---|---|---|
| **Single LUT Area** | **10.88 μm$^2$** | **2.15 μm$^2$** | **5.06×** |
| LUT Addressing Area | 5.68 μm$^2$ | 1.52 μm$^2$ | 3.73× |
| LUT Memory Area | 5.20 μm$^2$ | 0.63 μm$^2$ | 8.24× |
| **Tile Area** | **1561.5 μm$^2$** | **307.99 μm$^2$** | **5.07×** |
| CLB Area | 665.2 μm$^2$ | 63.290 μm$^2$ | 10.51× |
| CB Area | 337.7 μm$^2$ | 82.5 μm$^2$ | 4.1× |
| SB Area | 558.6 μm$^2$ | 162.2 μm$^2$ | 3.4× |

Table 4.5 Area of various FPCNA architectures

| | | **Cluster 4** | | **Cluster 10** | | **Cluster 20** | |
|---|---|---|---|---|---|---|---|
| | | *16 tubes* | *12 tubes* | *16 tubes* | *12 tubes* | *16 tubes* | *12 tubes* |
| **1-2-4 Wire Segments** | *CLB Area (μm$^2$)* | 25.316 | 23.784 | 63.29 | 59.46 | 126.58 | 118.921 |
| | *CB Area (μm$^2$)* | 23.46 | 23.46 | 75.01 | 75.01 | 203.438 | 203.438 |
| | *SB Area (μm$^2$)* | 205.06 | 205.06 | 444.49 | 444.49 | 829.437 | 829.437 |
| | *Total Tile Area (μm$^2$)* | 253.84 | 252.31 | 582.79 | 578.96 | 1159.46 | 1151.8 |
| | *Tile Edge Length (μm)* | 15.932 | 15.884 | 24.141 | 24.062 | 34.051 | 33.938 |
| **4-8 Wire Segments** | *CLB Area (μm$^2$)* | 25.316 | 23.784 | 63.29 | 59.46 | 126.58 | 118.921 |
| | *CB Area (μm$^2$)* | 27.07 | 27.07 | 82.5 | 82.5 | 435.94 | 435.94 |
| | *SB Area (μm$^2$)* | 83.94 | 83.94 | 162.2 | 162.2 | 1087.86 | 1087.86 |
| | *Total Tile Area (μm$^2$)* | 136.33 | 134.8 | 307.99 | 304.16 | 1650.38 | 1642.72 |
| | *Tile Edge Length (μm)* | 11.676 | 11.61 | 17.55 | 17.44 | 40.625 | 40.531 |

### 4.5.3 Performance Gain

In this section, the experimental CAD flow presented in Section 4.4 is evaluated, quantifying the overall performance improvement of FPCNA from the baseline CMOS counterpart. When considering variation, performance evaluation becomes complicated. The

critical path delay can no longer serve as the absolute measure of performance. Due to variations, near-critical paths may actually be statistically critical. This is illustrated by PO3 (primary output 3) in Figure 4.17. In addition, setting a clock period based only on the most statistically critical path is not appropriate. Consider the case in Figure 4.17, where the target clock period is set to a 95% guard-band of PO3. This means that for 95% of chips made, PO3 will not generate a timing failure. However, at this clock period, the other POs may also fail due to variation, making the overall yield less than 95%. Because of this phenomenon, it is necessary to consider the statistical delay of every path in yield analysis. The performance yield is expressed as a delay-probability pair ($t$, $p$), so that by setting the clock period $t$, one can evaluate the system yield $p$. This allows one to compare the performance of the statistical information generated by the experiments.

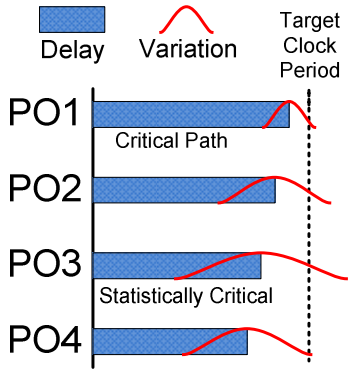



Figure 4.17 The effect of variation on critical path and yield

The performance yield for both Gaussian and non-Gaussian distributions is calculated using the flow in Figure 4.18. After selecting a target clock period *Tc*, the yield of each of the POs is computed. For a Gaussian distribution, the yield is calculated by computing the inverse cumulative distribution function (CDF) of the delay random variable. In a non-Gaussian delay distribution, the delay is represented by a group of points, so the yield is computed by converting the piecewise linear PDF into a piecewise linear CDF (Figure 4.19). The overall system yield is determined by multiplying all of the path yields. If the system yield is not satisfied, the *Tc* is increased and the process repeated until the desired yield is obtained. The final clock period is reported, which guarantees the targeted yield.

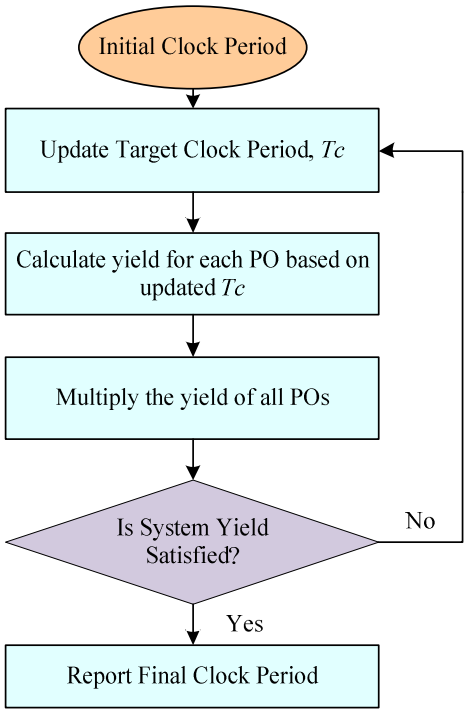



Figure 4.18 Performance yield estimation

Using the variation aware CAD flow, the achievable clock period of 20 MCNC benchmarks is evaluated, and the results are reported in Table 4.6. For a rough comparison to a deterministic solution, the CMOS design is evaluated using VPR [62], with a worst-case delay guard-band of 3σ added to each component in the VPR's architecture file. This equates to a component yield of roughly 99%.

The table shows the deterministic CMOS results, variation aware CMOS results, and two versions of variation aware FPCNA results: one with 16 nanotubes per CNT ribbon, and the other with 12 nanotubes per ribbon. For each variation aware flow, the clock period for performance yield is calculated at both 95% and 99%. Also generated is the performance gain of the FPCNA architectures over the baseline CMOS. In the table, both CMOS and FPCNA are configured with a cluster size of 10 and interconnect wire segmentation of 50% length-4 and 50% length-8. Average delays are calculated using the geometric mean. At a 95% performance yield, the FPCNA designs have an average gain of 2.75× and 2.65× over the CMOS counterpart, for 16 and 12 ribbons, respectively. This significant improvement in performance is achieved by the

synergistic combination of CNT logic, CNT interconnects, and routing crossbar in FPCNA.

As shown in Table 4.5, reducing number of tubes inside a nanotube ribbon can reduce tile footprint, which will reduce the length of global interconnects and should therefore enhance performance. However, as seen in Table 4.6, the overall performance is actually degraded. This is because with fewer tubes, each CNFET has less driving capability, which increases the LUT delay enough to overcome any global interconnect savings. To develop a better understanding of how the FPCNA architecture affects performance, different architecture combinations of wire segmentation, cluster size, and nanotube ribbon size were evaluated for the 20 benchmarks. The average results, again using the geometric mean, are plotted in Figure 4.20.

As seen in Figure 4.20 (a), for small and medium cluster sizes (4 and 10), long interconnects are preferable because they can make connections to CLBs which are distant. For the larger cluster size of 20, shorter wire segments are preferred (Figure 4.20 (b)). Note that in Figure 4.20 (a), the performance degrades rapidly at cluster size 20 because there are an increased number of connections between neighboring CLBs, and a limited number of short wire segments. The experiments also show that medium-sized clusters with longer interconnects have the best performance for FPCNA because a medium-sized cluster will take advantage of both CNT bundle interconnect and local routing.

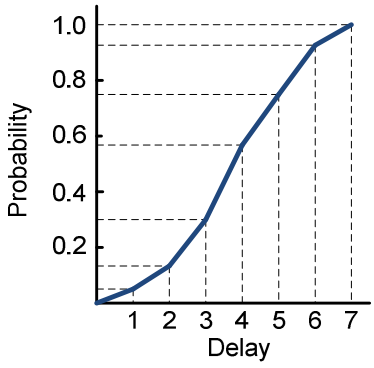
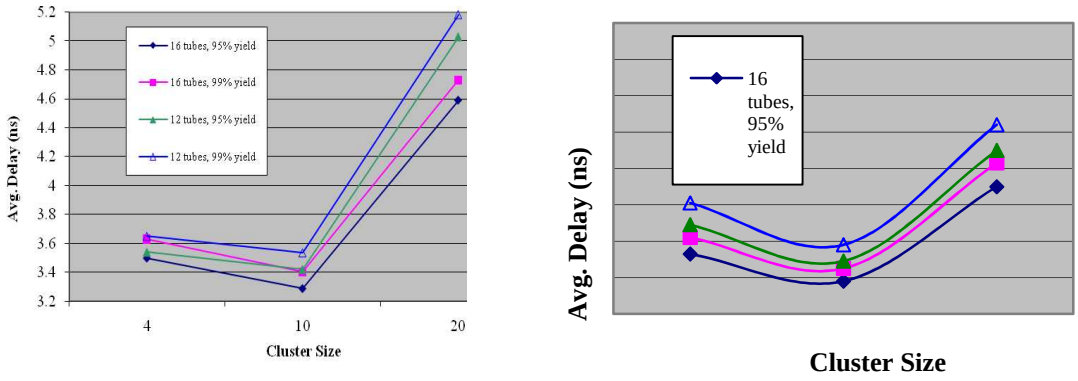



Figure 4.19 Piecewise linear CDF in discretized timing analysis




(a) (b)

Figure 4.20 Average delay for different architecture parameters at 95% and 99% yield: (a) 4-8 wire segmentation; and (b) 1-2-4 wire segmentation

Table 4.6 System clock period needed to achieve target performance yield

| **MCNC Benchmark** | **CMOS with Deterministic CAD Flow** | **CMOS with Variation-Aware CAD Flow** | | **FPCNA with Variation-Aware CAD Flow (**16 CNTs Per Ribbon) | | | **FPCNA with Variation-Aware CAD Flow (**12 CNTs Per Ribbon) | | |
|---|---|---|---|---|---|---|---|---|---|
| | *99% Component Yield (ns)* | *95% Performance Yield (ns)* | *99% Performance Yield (ns)* | *95% Performance Yield (ns)* | *99% Performance Yield (ns)* | *Perf. Gain over CMOS at 95% Yield* | *95% Performance Yield (ns)* | *99% Performance Yield (ns)* | *Perf. Gain over CMOS at 95% Yield* |
| alu4 | 9.262 | 7.338 | 7.469 | 2.559 | 2.698 | 2.87× | 2.678 | 2.812 | 2.74× |
| apex2 | 10.51 | 8.444 | 8.587 | 3.235 | 3.313 | 2.61× | 3.263 | 3.307 | 2.59× |
| apex4 | 9.796 | 7.602 | 7.726 | 3.666 | 3.706 | 2.07× | 3.460 | 3.756 | 2.2× |
| bigkey | 4.580 | 4.336 | 4.416 | 1.480 | 1.502 | 2.93× | 1.474 | 1.495 | 2.94× |
| clma | 20.55 | 18.98 | 19.18 | 5.666 | 5.720 | 3.35× | 6.790 | 6.818 | 2.8× |
| des | 8.900 | 8.853 | 8.994 | 2.884 | 2.921 | 3.07× | 3.027 | 3.058 | 2.92× |
| diffeq | 7.241 | 6.351 | 6.448 | 2.736 | 2.978 | 2.32× | 2.827 | 3.070 | 2.25× |
| dsip | 4.790 | 4.856 | 4.954 | 1.643 | 1.647 | 2.96× | 1.668 | 1.682 | 2.91× |
| elliptic | 14.87 | 11.26 | 11.39 | 3.342 | 3.483 | 3.37× | 3.810 | 3.967 | 2.96× |
| ex1010 | 16.39 | 12.99 | 13.15 | 5.215 | 5.363 | 2.49× | 4.801 | 5.046 | 2.71× |
| ex5p | 9.885 | 8.693 | 8.847 | 3.760 | 3.812 | 2.31× | 4.500 | 4.554 | 1.93× |
| frisc | 16.11 | 14.99 | 15.15 | 3.908 | 4.367 | 3.84× | 5.114 | 5.316 | 2.93× |
| misex3 | 8.284 | 6.543 | 6.649 | 3.092 | 3.284 | 2.12× | 2.709 | 2.899 | 2.42× |
| pdc | 17.25 | 16.13 | 16.32 | 4.637 | 4.863 | 3.48× | 4.770 | 4.957 | 3.38× |
| s298 | 15.14 | 14.10 | 14.25 | 3.822 | 3.857 | 3.69× | 4.029 | 4.134 | 3.5× |
| s38417 | 10.97 | 10.62 | 10.74 | 4.314 | 4.370 | 2.46× | 3.463 | 3.590 | 3.07× |
| s38584.1 | 8.456 | 7.024 | 7.140 | 2.816 | 2.894 | 2.49× | 2.884 | 3.019 | 2.44× |
| seq | 10.78 | 7.859 | 7.987 | 3.203 | 3.344 | 2.45× | 3.634 | 3.757 | 2.16× |
| spla | 15.20 | 12.04 | 12.20 | 4.643 | 4.730 | 2.59× | 4.826 | 4.864 | 2.49× |
| tseng | 8.851 | 6.700 | 6.804 | 2.692 | 2.785 | 2.49× | 2.835 | 2.917 | 2.36× |
| Average | 10.59 | 9.070 | 9.203 | 3.293 | 3.404 | 2.75× | 3.417 | 3.536 | 2.65× |

# CHAPTER 5
# VARIATION AWARE ROUTING FOR THREE-DIMENSIONAL FPGAS

In modern field programmable gate arrays (FPGAs), most of the chip area is devoted to the programmable interconnect used for the local and global routing of signals. As design complexities increase, signal paths become longer and have to connect across greater distances and through more programmable switches. The task of driving this capacitive interconnection fabric increases the critical path delay and power consumption of FPGA designs. One recognized solution to this problem is to move to a 3D architecture, where layers of logic are stacked on top of each other instead of being spread across a 2D plane. Devices on each layer connect to devices on adjacent layers through the use of vias, increasing logic density, and minimizing the average connection length.

This chapter develops a new SSTA engine designed to deal with the uncorrelated and correlated variations in 3D FPGAs. The effects of intra-die and inter-die variations are considered to develop accurate timing models. Using the 3D placement and routing framework of TPR (three-dimensional FPGA placement and routing) [82], a new 3D routing algorithm is developed which uses this engine as the basis for improving performance yield. As far as is known, this is the first physical design tool to consider variation in the routing and timing analysis of 3D FPGAs.

Variation aware routing is addressed instead of variation aware placement and/or technology mapping because the most detailed timing and variation information is available during the routing phase. For instance, correlated variation between lookup tables is only known after the placement phase, when their locations have been fixed.

## 5.1 Related Work

### 5.1.1 3D Stacking

There are three main strategies for manufacturing 3D integrated circuits: monolithic stacking, wafer-based stacking, and die-based stacking. In monolithic stacking, layers of logic are fabricated on top of existing layers of logic and interconnect. This is an ideal solution that allows layers to connect using minimally sized metal vias, but it is difficult to achieve in practice because the heat required to manufacture transistors has the potential to destroy the metal routing in the layers below.

Another solution is wafer-based stacking, in which layers of transistors and wiring are fabricated on separate wafers. These wafers are bonded together to form a multilayer wafer, and then diced into 3D ICs. The problem with this scenario is that the yield decreases as the number of layers grows, since the failure of a die on any layer will render the final IC inoperable.

In die-based stacking, wafers are diced before bonding, allowing defective dies to be discarded. By using only known good dies, the 3D device yield can be maximized. Die-based stacking is especially attractive for homogeneous FPGA architectures, where logic tiles are replicated identically in each layer, allowing a single set of masks to be used.

### 5.1.2 3D FPGAs

A number of 3D FPGA architectures and CAD tools have been proposed in the literature. One of the first works to address the 3D FPGA placement and routing problem was [83], in which Alexander et al. extended an iterated KMB (Kou, Markowsky, and Berman) algorithm into three dimensions. In another early work, Karro and Cohoon presented a simultaneous placement and global routing algorithm for 3D based on partitioning [84]. More recently, 3D routing tools were developed to characterize the performance of monolithically stacked 3D FPGAs by Lin et al. in [4].

A 3D physical design engine called TPR was presented by Ababei et al. in [82]. This

engine is an extension of the popular VPR tool [62] into three dimensions, and the source code is freely available for academic use. Three placement algorithms were considered: a global min-cut partition to assign a netlist into layers, a timing-driven, intra-layer placement based on hMetis partitioning, and a 3D simulated annealing engine. The TPR routing tool is a Pathfinder-negotiated congestion algorithm with added penalties to avoid vias [82].

Recently, Gayasen et al. addressed a number of 3D FPGA design issues in [85]. Area and critical path were compared for different combinations of layer number, bonding strategy, and via density. The routing resource utilization of a five-layer stack was compared to an equivalent 2D stack and shown to be more efficient, assuming a 3 µm via pitch. In addition, six potential 3D switchbox designs were evaluated. To analyze these designs, a timing-driven placement and routing CAD flow was developed by extending VPR and adding a vertical channel congestion parameter into the VPR cost function.

While the progression of these works demonstrates a considerable evolution in the sophistication of 3D FPGA CAD, none of these tools addressed the significant impact of process variations on circuit performance.

### 5.1.3 Variation Aware Routing

On the 2D front, the statistical optimization of FPGA design tools has been receiving increasing attention. Previous works have demonstrated the particular effectiveness of such optimization during the physical design stage.

Sivaswamy and Bazargan presented a variation aware routing algorithm in [78] that treats all sources of variation as spatially correlated, but ignores the effects of uncorrelated random variation. They conclude that while statistical optimizations are not currently as critical for FPGAs as they are for ASICs, process variations will become increasingly significant in future technologies, and statistical FPGA optimization techniques need to be explored.

In [86], Lin et al. create a complete variation aware physical synthesis flow that incorporates statistical clustering, statistical placement, and statistical routing. Since detailed

routing information is not known during clustering, interconnect uncertainty is modeled as a random variable and used to characterize performance. For placement and routing, process variation is considered. When the effects of variation are considered during all three stages of physical synthesis, average yield improvements of 9.1% at a 95% yield, and 12.6% at a 90% yield are shown.

## 5.2 3D FPGA Architecture

Like their 2D counterparts, 3D FPGAs can adopt a traditional island-style FPGA architecture. This architecture contains a fabric of repeated tiles that consist of one switch block (SB), two connection blocks (CBs), and one configurable logic block (CLB). Figure 5.1 illustrates two such tiles in a 3D stack. Connections are made between the layers using through-silicon vias (TSVs). The architectures in this study only allowed TSVs to connect in the SBs, as shown by the vertical lines in the figure.

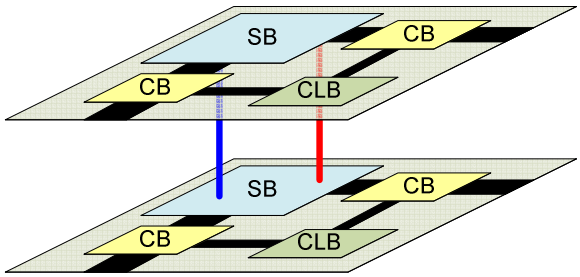



Figure 5.1 3D island-based FPGA tiles

The 3D switch box is an important component in the 3D FPGA architecture, which provides normal routing connections between the *x* and *y* horizontal routing channels, as well as vias to connect those channels vertically to additional layers in the 3D stack. Figure 5.2 shows a possible 3D switch box design that demonstrates 3D switch point connections. For simplicity, switch points that only connect to the horizontal channels are not shown. The vertical vias are segmented to achieve electrical isolation between layers. This means that each switch box must contain independent connections for vias connecting to the upper and lower layers. This allows a

signal from a horizontal wire to be directed to a specific layer instead of being sent both upwards and downwards. The pattern can be extended for any channel width by adding the appropriate number of switch points.

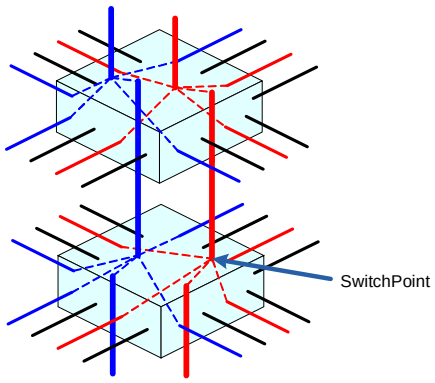



Figure 5.2 3D subset switch box layout

In 3D circuit design, the vertical interconnects require special consideration. When two logic layers are connected in a face-to-face bonding process [87], their metallization layers are joined, and the size of the connecting vias is limited by the accuracy of the layer alignment technique used. For designs with more than two layers, face-to-back bonding is needed, which requires connections through the substrate. The properties of the connecting vias are determined by the substrate thickness. In a traditional bulk silicon process with a wafer thickness of 50–300 μm, through-silicon vias (TSVs) can be made with diameters as small as 10 μm. However, if a silicon-on-insulator (SOI) process is used, the substrate is reduced to a thickness of 1–50 μm and vias can be made with diameters as low as 1 μm [88].

In addition to via diameter, the fabrication method may also affect the via density. The following calculations assume a 32 nm process where the via pitch is only constrained by the switch point area. However, manufacturing limitations such as the substrate thickness and bonding technique will not scale in the same ratio as lithography-driven feature sizes, so it is possible that these limitations will dictate the via pitch below the 32 nm technology node. This is especially true for a face-to-back bulk silicon process which requires larger vias than SOI.

To study the relationship between switch block area, vias density, and horizontal interconnection length, the parameter $Z_{frac}$ is defined according to Equation (5.1):

$$Z_{frac} = Z_{width} / \max(X_{width}, Y_{width}) \quad (5.1)$$

Table 5.1 Via density vs. interconnect length

| Via Density | Tile Size | Length 1 Wire |
|---|---|---|
| $Z_{frac}$ = 20% | 244.58 μm$^2$ | 15.64 μm |
| $Z_{frac}$ = 30% | 267.32 μm$^2$ | 16.35 μm |

$Z_{frac}$ represents the number of vertical routing vias ($Z_{width}$) compared to the maximum horizontal channel width, and can be thought of as the percentage of switch points in a switch box that have vertical connections. In this study, architectures with $Z_{frac}$ = 20% and $Z_{frac}$ = 30% are evaluated.

Based on the routing switch design in [89], each switch point takes an estimated 150.4 *T*, where *T* is the minimum-width transistor area. With a 0.0451 μm$^2$ transistor area at the 32 nm technology node [1], this gives us a switch point area of 6.783 μm$^2$. Assuming a via diameter in the range of 1.5 μm [88], the area of a via-containing switch point is estimated as 1.5× larger than a horizontal-only switch point.

By adding the CLB area to the area of the switch blocks and connection blocks, the estimated size of an FPGA tile for $Z_{frac}$ = 20% and $Z_{frac}$ = 30%. Table 5.1 presents the trade-off between via density and routing delay. The results show that while increasing the number of vias in a switch box gives the physical design tools greater routing flexibility, it increases the tile size and requires longer horizontal interconnections.

## 5.3 Variation Modeling

Developing an accurate statistical timing analysis engine requires first characterizing the variation model of 3D FPGAs, considering both correlated and uncorrelated variation sources. Uncorrelated random variations can be approximated as normally distributed random variables with mean μ and standard deviation σ. Sources of uncorrelated variation include the random concentration of doping atoms in transistor source and drain, and the irregularities in interconnect width, thickness, and spacing.

Correlated variations are used to represent the trend of certain process parameters to vary

according to their location on the chip. To characterize spatial correlation, the overall containing area can be divided into grids where every device in a grid square is assumed to be perfectly correlated. The variation relationships between the resulting $n$ grid squares can then be represented by an $n \times n$ correlation matrix. Such a matrix is needed for performing principal component analysis, as described in [90].

### 5.3.1 Interconnect Delay

Interconnection delay is random based on the geometrical variation of wire width, wire thickness, and spacing [76]. This study assumes interconnect variation is an independent random variable and models it for wires and vias based the Equation (5.2) from [91].

$$\sigma = 0.3836 \times \exp(-0.1537h) \times \mu_0 \quad (5.2)$$

In the above equation, $\sigma$ is the standard deviation, $h$ is the size of the driving buffer, and $\mu_0$ is the nominal wire delay. For the size-10 and size-5 buffers in this study's architecture, this gives the standard deviations of 8.2% and 17.8% of $\mu_0$, respectively.

### 5.3.2 Logic Delay

For logic devices, both random and spatially correlated variations and express delay are considered according to Equation (5.3):

$$D = D_{nom} + \Delta D_{intra\_spatial} + \Delta D_{intra\_rand} + \Delta D_{inter} \quad (5.3)$$

In this equation, $D_{nom}$ is the nominal delay value and is adjusted based on the variation from intra-die ($D_{intra\_spatial}$, $D_{intra\_rand}$) and inter-die ($D_{inter}$) sources.

### 5.3.3 Intra-Die Variation

Intra-die variation is the variation that causes device performance to vary across a single die. Depending on the source of the variation, intra-die variation can be correlated or uncorrelated. To model the correlated intra-die variation, both gate length and oxide thickness are considered. These contributing process parameters are considered independent and separate

correlation matrices are created for each. Additional process parameters could be modeled with additional matrices. Unlike 2D chips, which need only one correlation matrix per parameter, 3D FPGAs are made from a number of separate dies and need a correlation matrix on each layer for each parameter. This study considers intra-die variation with a 10% random component and a 10% spatially correlated component, with a correlation distance of 1 mm [91]. The size of each correlation matrix is set by dividing the dies into grids such that each grid square contains a certain number of FPGA architecture tiles.

Since published variation information from device manufactures is lacking, example data to fill the correlation matrices are generated using a method from Xiong et al. in [92]. This method ensures that the correlation matrices are positive-semi definite, a requirement for the principal component analysis. In an actual production flow, measured correlation data could be used.

### 5.3.4 Inter-Die Variation

Inter-die variation is the variation correlated between dies. The stacking process used determines if there will be any inter-die correlation. In a homogenous die-based stacking process where each die in the stack comes from the same wafer, inter-die variation can be considered. One way to do this is to create large correlation matrices for each parameter that define relationships between the grid squares on all of the layers. The size of these matrices can be calculated by Equation (5.4).

$$M_{size} = (n_{grid\_squares} \times n_{layers})^2 \tag{5.4}$$

where $n_{grid\_squares}$ is the number of grid squares in each layer and $n_{layers}$ is the number of layers in the device.

If wafer-based stacking is used instead of die-based stacking, the 3D devices will be made up of dies from different wafers, so inter-die correlation will be zero and should not be considered. However, wafer-to-wafer variation can be considered in its place to account for manufacturing processes that vary in the same way across each wafer.

The present experiments assume a homogenous die-based stacking process and a grid size of one tile. Since there are a large number of tiles in the benchmarks, this gives us a large $n_{grid\_squares}$. Considering correlated inter-die variations under these conditions would require a large $M_{size}$. Since operations performed on such matrices would be prohibitively expensive, a simpler model is used that treats inter-die variation as normally distributed with a standard deviation of 10% of the mean. This corresponds to the 11% random inter-die variation seen in [93].

## 5.4 CAD Tools

### 5.4.1 3D SSTA

Efficient timing analysis generally requires the use of independent random variables. The present 3D SSTA follows the work of Chang and Sapatnekar in [90]. This method leverages principal component analysis (PCA) to determine circuit behavior under correlated variations, and is widely used in 2D SSTA. PCA is a statistical technique that allows the transformation of a set of correlated random variables into a new set of uncorrelated random variables known as principal components. PCA is used to transform each process parameter's correlation matrix into a set of principal components (PCs) at the start of the CAD flow.

To perform 3D SSTA, each node in the timing graph must be made to store variation in addition to nominal delay. In this study's timing graphs, normally distributed random variation and $n \times p$ sets of PCs are added into each node, where $n$ is the number of layers in the stack and $p$ is the number of independently correlated process parameters. If each die is divided into $m$ grid squares, there will be $m$ PCs in each set. The maximum number of PCs is therefore $n \times p \times m$. Since many of these PCs will remain empty, memory is allocated for a set of PCs only when a non-zero value is stored.

Within a layer, statistical operations such as ADD and MAX are carried out by operating on the PCs for that layer. For example, consider two timing nodes: n1 and n2, such that each contains a set of PCs for each layer: pc1 for layer 1 and pc2 for layer 2. If both n1 and n2 belong

to layer 1, only pc1 is used during statistical operations. If n1 and n2 belong to layer 2, only the PCs in pc2 are used. When nodes n1 and n2 are located in two different layers, the PCs from both layers have to be considered.

This is illustrated by the inter-layer MAX operation shown in Figure 5.3, in which node n1 is on layer 1, and nodes n2 and n3 are on layer 2. Node n1 only has values for pc1, the spatially correlated variation within layer 1. Correspondingly, n2 only has PCs from layer 2. Therefore, the MAX operation of these two nodes must consider PCs representing intra-die correlations from both layers. To do this, each set of PCs in a node is statistically maxed with its counterpart in the other node. For example, pc1 set from n1 is maxed with the 0 from n2. Similarly, pc2 from n2 is maxed with the 0 in n1. The result then consists of PCs related to parameters on both layers. Using the same process, this result can then be maxed with additional nodes, such as n3. Note that this only accounts for intra-layer variation. Inter-layer variation ia modeled as an additional 10% random variation when performing inter-layer operations.

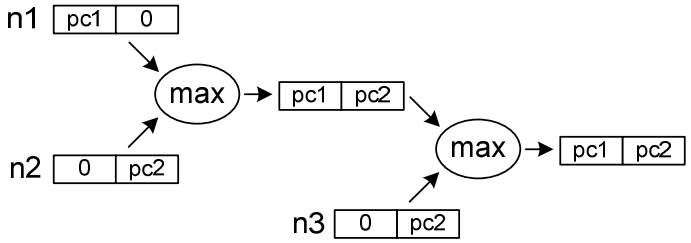



Figure 5.3 Inter-layer MAX operation

In a 3D architecture, nets can connect across multiple layers, spanning multiple spatially correlated variation domains. Using the techniques outlined above, the delay and slack calculations of the present SSTA tool account for this by accumulating and combining sets of PCs for each of the correlated variation sources.

### 5.4.2 Variation Aware 3D Router

To perform variation aware routing, the 3D FPGA placement and routing tool TPR [82] is modified. TPR uses a routability-driven cost function focused on minimizing congestion. A routability-driven router is useful because the vertical connections of a 3D architecture increase

the complexity and difficulty of achieving a legal routing solution. However, to achieve competitive performance, timing information must also be considered. The TPR router only uses static timing analysis to calculate the final critical path delay. This baseline is improved upon by adding a variation aware SSTA engine and incorporating timing information into the router's cost function. The pseudo-code for this new router is shown in Figure 5.4.

```
Compute correlation matrices;
Compute principle components and  generate routing, timing graphs.

Routing starts: set all nets i and sinks j, Crit(i,j) =1.0;
while (overused routing resources exist) do
    for (each net, i) do
        Rip-up routing tree of net i;
        for (each sink j of net i in decreasing Crit(i,j) order) do
            breadth first routing of sink j;
            for (all nodes in the path from i to j) do
                Update congestion;
            end
        end
    end
    Update historic congestion;
    Compute routing tree delay and variation.
    Update timing graph.
    SSTA compute mean and variation of arrival time and required time
                for each net(i);
    Update net Crit(i,j);
end
```

Figure 5.4 Pseudo-code of the modified TPR router

The first step in this algorithm is to capture spatial correlations in the form of correlation matrices. No correlation between layers is assumed, so each layer has its own correlation matrix for each correlated process parameter. PCA is then used to generate the uncorrelated PCs for each node.

The rest of the routing is iterative. During the first iteration, the criticality of each pin in every net is set to 1 (the highest criticality) to minimize the wire length of each pin. After all of the nets are routed, SSTA is performed by traversing the updated timing graph to calculate the new slack and critical path delays. The criticality of pin *j* in net *i* is then computed, considering variation in the slack and critical path according to Equation (5.5).

$$Crit(i,j) = 1 - \frac{slack(i,j) - 3\sigma_s(i,j)}{t_{crit} + 3\sigma_{crit}} \tag{5.5}$$

The criticality function is derived in this way so that when two slacks have a similar mean but different variations, the $slack(i, j) - 3\sigma_s(i, j)$ term assigns larger criticality to the path with the greatest variation, weighting it more heavily in the next iteration. It is possible that some nets have positive slack but a large enough variation that the $slack - 3\sigma s$ term becomes zero or even negative. All of these nets are considered critical and will be assigned the maximum criticality.

Using this new criticality, the cost function used in TPR is updated during maze routing. This function is used to select between multiple paths during breadth-first wave-front expansion. The function TPR used to calculate the total cost, $C_{total}$, is shown in Equation (5.6).

$$C_{total} = C_{path} + b(c)h(c)p(c) \tag{5.6}$$

In this equation, $C_{path}$ is the path cost at the current routing node in the maze expansion wave front and *b(c)h(c)p(c)* is the congestion cost of the routing resource. The congestion cost consists of a base cost *b(c)*, which is 1, 0.95, or 0, depending on the routing resource type, a historic congestion cost *h(c)*, reflecting the overuse of this routing resource in the past, and a present congestion cost *p(c)*, representing its current use.

The new criticality function is used to update this cost function by replacing the congestion cost by a timing and variation aware routing resource cost, as shown in Equation (5.7).

$$C_{total} = C_{path} + b(c) + \sqrt{(1 - crit_i)}\,h(c)\,p(c) \tag{5.7}$$

This differs from the original TPR cost function in that the base cost is separated from the congestion cost, and the congestion cost is scaled by $\sqrt{(1 - crit_i)}$. Adding the net criticality, $crit_i$ into Equation (5.7) allows nets with higher criticality to be less affected by congestion during maze expansion. The square root is used to preserve more of the congestion cost for lower criticality nets so that timing closure can also be achieved quickly. In this way, the impact of congestion on the critical path is greatly reduced, but the base cost remains the same.

In variation-aware routing, several nets are likely to be critical under statistical analysis.

If each of these nets completely ignores congestion with a $crit_i$ = 1, overused resources could persist indefinitely, resulting in an unroutable situation. Therefore, the maximum net criticality value is set to be slightly less than 1 instead of using 1. This allows the congestion component to resolve such situations.

To further enhance the routing results, *b(c)* is set according to the statistical delay of the routing resources instead of using the base cost values suggested in [62] that do not consider timing information. This is shown in following equation, where the delay of the routing resource node in question is $rr\_node_{mean}$ and $rr\_node_{var}$.

$$b(c) = rr\_node_{mean} + 3 \times rr\_node_{var} \quad (5.8)$$

By considering timing in the base cost, *b(c)*, and considering the base cost separately from the historic costs, a routing resource with a larger propagation delay will now have an appropriately larger cost. This differentiation of routing resources allows the new routability-driven router to select routing resources with lower propagation delay during maze expansion. As a result, the router achieves a large performance gain over the baseline routability-driven router used in TPR.

## 5.5 Experimental Results

### 5.5.1 Experiment Setup

This experiment runs simulations for 2-layer and 3-layer FPGA designs, using a standard set of 12 MCNC benchmarks. For each configuration, via densities of $Z_{frac}$ = 20% and $Z_{frac}$ = 30% are calculated to compare the impact of via density on routing results. Note that these techniques could also be scaled to a larger number of layers.

Traditional island-style FPGA architecture is used, with size 4 lookup tables and a cluster size of 1. Delays of components such as LUTs and buffers are characterized in HSPICE using PTM 32 nm models. A fixed routing channel width is set at 30 with buffered drivers. A mixture of length 1-, length 2-, and length-6 wires are used (wires spanning 1, 2, or 6 CLBs). Vias of both length 1 and 2 (crossing 1 or 2 layers) are used in the 3-layer case.

Before place and route, T-VPACK [62] is used for timing-driven packing of these benchmarks. The packed netlists are placed using the partition-based placement algorithm in TPR [82]. The same placement files are used to evaluate both variation aware and baseline TPR routing. TPR baseline results are generated by the original TPR deterministic router. An SSTA is then performed on the routed circuits to estimate the resulting mean and variation. Variation aware routing is based on the algorithm described in Section 5.4 and nets are routed using dynamically updated criticality.

### 5.5.2 Results and Discussion

Table 5.2 details the performance of the baseline TPR and variation aware routing results for 2- and 3-layer FPGAs. Due to space limitations, only the average values for the 3-layer experiments are shown. This table shows that for both 30% and 20% via density, the new variation aware router improves the guard-banded $\mu+3\sigma$ performance of a 2-layer FPGA by over 22%, and a 3-layer FPGA by over 27%. Note that using a higher via density (30%) will increase the routability of the architecture, but comes at a performance penalty. This is due to the corresponding increase in horizontal wire length and variation, as described in Section 5.2.

If only nominal values are considered in the baseline TPR routing, increasing the 3D stacking to 3 layers shows a 13% and 9.1% reduction in the critical path delay for 20% and 30% via densities, respectively. However, once the variation of the resulting 3D architectures is considered, this advantage is degraded to 7.0% and 4.4%, respectively. This is because the stacking of an additional layer increases the amount of inter-die variation.

Table 5.2 Performance of two- and three-layer FPGA routing (ns)

| Benchmark Circuit (2 Layer) | Via Density 20% | | | | | | | Via Density 30% | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TPR Baseline | | | Variation Aware | | | μ+3σ Reduction | TPR Baseline | | | Variation Aware | | | μ+3σ Reduction |
| | μ | σ | μ+3σ | μ | σ | μ+3σ | | μ | σ | μ+3σ | μ | σ | μ+3σ | |
| tseng | 10.1 | 1.47 | 14.5 | 7.21 | 0.68 | 9.24 | 36.3% | 10.5 | 1.52 | 15.1 | 7.13 | 0.71 | 9.25 | 38.8% |
| ex5p | 7.09 | 0.99 | 10.1 | 5.75 | 0.47 | 7.15 | 29.0% | 7.88 | 1.10 | 11.2 | 6.11 | 0.56 | 7.79 | 30.3% |
| diffeq | 7.43 | 0.91 | 10.2 | 6.36 | 0.76 | 8.65 | 14.8% | 6.66 | 1.07 | 9.86 | 6.26 | 0.75 | 8.51 | 13.7% |
| misex3 | 8.48 | 0.73 | 10.7 | 6.60 | 0.20 | 7.21 | 32.4% | 8.51 | 0.84 | 11.0 | 7.38 | 0.42 | 8.65 | 21.6% |
| apex4 | 8.02 | 0.71 | 10.2 | 7.37 | 0.31 | 8.29 | 18.4% | 8.23 | 0.71 | 10.3 | 7.25 | 0.43 | 8.55 | 17.4% |
| alu4 | 7.93 | 0.67 | 9.93 | 7.17 | 0.35 | 8.21 | 17.3% | 8.75 | 0.77 | 11.1 | 7.11 | 0.40 | 8.32 | 24.8% |
| seq | 7.96 | 0.77 | 10.3 | 7.31 | 0.40 | 8.51 | 17.1% | 7.40 | 0.70 | 9.50 | 7.18 | 0.53 | 8.76 | 7.72% |
| apex2 | 9.48 | 1.28 | 13.3 | 6.94 | 0.24 | 7.66 | 42.5% | 10.3 | 1.41 | 14.5 | 7.79 | 0.34 | 8.82 | 39.2% |
| dsip | 4.65 | 0.51 | 6.17 | 4.55 | 0.47 | 5.97 | 3.29% | 4.44 | 0.51 | 5.97 | 4.22 | 0.43 | 5.50 | 7.91% |
| des | 7.22 | 0.82 | 9.67 | 6.14 | 0.66 | 8.12 | 16.0% | 6.78 | 0.97 | 9.69 | 5.47 | 0.61 | 7.31 | 24.6% |
| s298 | 22.0 | 2.28 | 28.8 | 15.9 | 2.09 | 22.1 | 23.3% | 24.3 | 2.74 | 32.5 | 16.3 | 2.01 | 22.3 | 31.4% |
| bigkey | 4.77 | 0.45 | 6.12 | 3.45 | 0.44 | 4.76 | 22.2% | 4.83 | 0.51 | 6.35 | 4.59 | 0.44 | 5.91 | 7.01% |
| 2-Layer Average | 8.07 | 0.87 | 10.7 | 6.62 | 0.48 | 8.21 | 22.7% | 8.19 | 0.95 | 11.1 | 6.81 | 0.56 | 8.56 | 22.0% |
| 3-Layer Average | 7.02 | 0.96 | 9.97 | 5.45 | 0.48 | 6.92 | 28.5% | 7.44 | 1.02 | 10.6 | 5.75 | 0.55 | 7.45 | 27.4% |

# CHAPTER 6
# 3D NANOELECTROMECHANICAL RELAY-BASED RECONFIGURABLE ARCHITECTURE

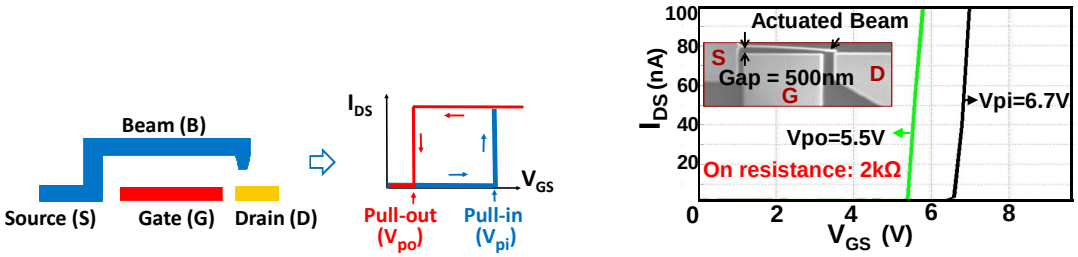
Nanoelectromechanical relay (NEM) [94] devices, which are electrostatically-actuated switches with zero leakage at off state and low resistance at on state, show promising electrical characteristics comparing to CMOS pass transistors. Another advantage of NEM relays is that it is possible to encapsulate them into metal layers and, therefore, to integrate them on top of CMOS. Motivated by this leading technology, this chapter presents a 3D hybrid CMOS-NEM FPGA architecture, namely, 3D CMOS-NEM FPGA. The novelty of this 3D CMOS-NEM FPGA lies in the combination of 3D FPGA architecture design and NEM technology, which will significantly advance future large-scale programmable devices.

To maximize the benefit of this new architecture, a 3D placement and routing flow has been developed based on the state-of-art FPGA placement routing tool VPR5.0 [95]. This 3D flow is flexible; it takes 3D architecture file as input and dynamically generates the 3D architecture to be evaluated. The placement and routing algorithms in VPR are tuned and enhanced for 3D purposes.

This chapter is organized as follows: Section 6.1 introduces the principle of operation and advantages of NEM devices. NEM-based LUTs and routing switch designs are provided in Section 6.2. In Section 6.3, the overall 3D CMOS-NEM FPGA architecture is presented. Section 6.4 describes in detail the 3D CAD flow. Experimental results showing the advantages of 3D NEM FPGA over a conventional CMOS and 2D counterpart are presented in Section 6.5, and Section 6.6 concludes this chapter.

## 6.1 NEM Devices

Nanoelectromechanical relays are electrostatically-actuated switches that have zero leakage at off state and are promising to achieve relatively low on-resistance compared to CMOS pass transistors. Figure 6.1 (a) shows the structure of a three-terminal (3T) NEM relay, which consists of 1) a deflecting beam (connected to the source electrode), which forms the channel for current flow; 2) a gate electrode with a gap to the beam, which can control the state of the switch through electrostatic force; and 3) a drain electrode, which connects to the beam when the NEM relay is in its on state. When gate voltage ($V_{GS}$) is applied, electrostatic force attracts the beam towards the gate, while the elastic force in the beam resists the beam from deflecting. Beyond a certain $V_{GS}$, defined as pull-in voltage (*Vpi*), the elastic force can no longer balance the electrostatic force, and the beam collapses toward the gate until contact is made at the drain. Since pull-in is achieved through electromechanical instability, the voltage at which the beam disconnects from the drain (pull-out voltage, *Vpo*) is smaller than *Vpi*. This leads to hysteresis in the current-voltage characteristics of NEM relays (Figure 6.1 (a)). Figure 6.1 (b) shows the IV characteristics of a fabricated 3T NEM relay, where zero leakage in the off state is confirmed, and an on-resistance of 2 kΩ is demonstrated [96].




(a) (b)

Figure 6.1 (a) Structure of a three-terminal (3T) NEM relay and its IDS-VGS curve; and (b) measured IV characteristics of a fabricated NEM relay with 2 kΩ on-resistance

All structural materials to fabricate NEM relays could be typical materials in standard CMOS back-end-of-line (BEOL) processes [97]. Due to the low processing temperatures of these materials, it is promising that the fabrication of NEM relays could be BEOL compatible. In

addition, encapsulating NEM relays between metal layers after fabrication [97]-[98] could enable monolithic 3D integration of NEM relays on top of CMOS to reduce area, as indicated in Figure 6.2 .

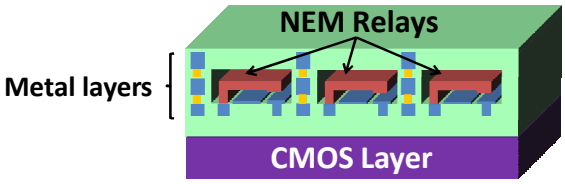
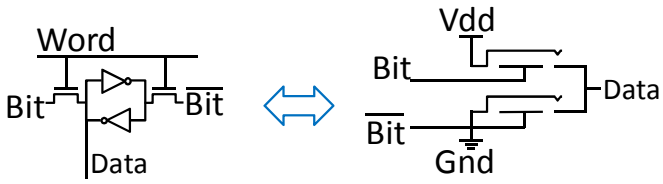



Figure 6.2 Encapsulated NEM relays between metal layers to enable monolithic 3D integration

## 6.2 NEM-Based FPGA Tiles

### 6.2.1 NEM Relays as LUT Memory Element

In CMOS SRAM-based FPGA, the major component is the lookup tables (LUT). Consisting of CMOS SRAM cells (Figure 6.3 (a)) and an NMOS pass-transistor-based multiplexer, they are used to provide programmable logic functions. Inside each LUT, pre-programmed SRAM cells provide corresponding values to the output, which could be either logic high (*Vdd*) or logic low (*Gnd*).

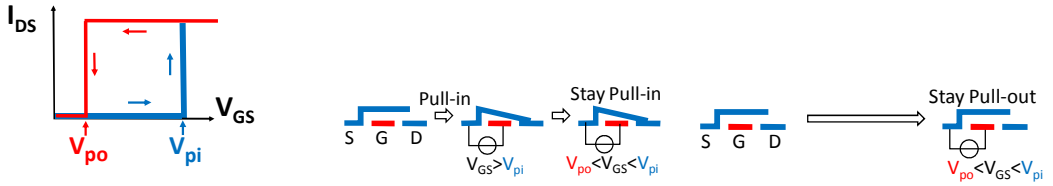



(a) (b)

Figure 6.3 (a) CMOS 6-transistor SRAM cell; and (b) NEM Memory cell

Hysteresis characteristics of NEM relays enable the use of NEM relays as memory components, which makes it possible to replace CMOS SRAM cells inside CMOS LUTs. As shown in Figure 6.4 (a), after being pulled in by applying a $V_{GS}$ greater than *Vpi*, applying $V_{GS}$ inside the hysteresis window ($Vpo<V_{GS}<Vpi$) will keep the NEM relay in the pull-in (closed) state (Figure 6.4 (b)). However, if the NEM relay is in the pull-out (open) state, applying $V_{GS}$ inside ($Vpo<V_{GS}<Vpi$), the relay will remain in the pull-out (open) state (Figure 6.4(c)). As NEM

relays have zero leakage in their off state and can be placed on top of CMOS, replacing CMOS SRAM cells with NEM relays will help reduce LUT leakage and reduce LUT layout area. Figure 6.5 shows the reduction in area, delay, and leakage power comparing CMOS-NEM 4-input LUT with traditional CMOS-only LUT. As shown in Figure 6.5, stacking NEM relays on top of CMOS can lead to 53.12% reduction in LUT layout area. In the meantime, 55% leakage reduction, and 9.3% delay reduction can be achieved due to zero leakage and low on-resistance of the NEM relay.




(a) (b) (c)

Figure 6.4 Different states of NEM relay based on its hysteresis property: (a) hysteresis ring; (b) pull-in process; and (c) pull-out process

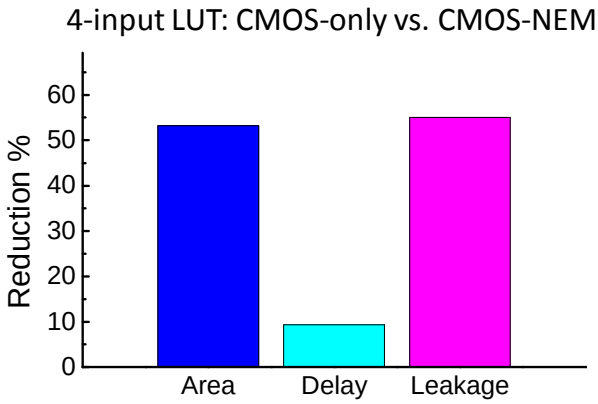
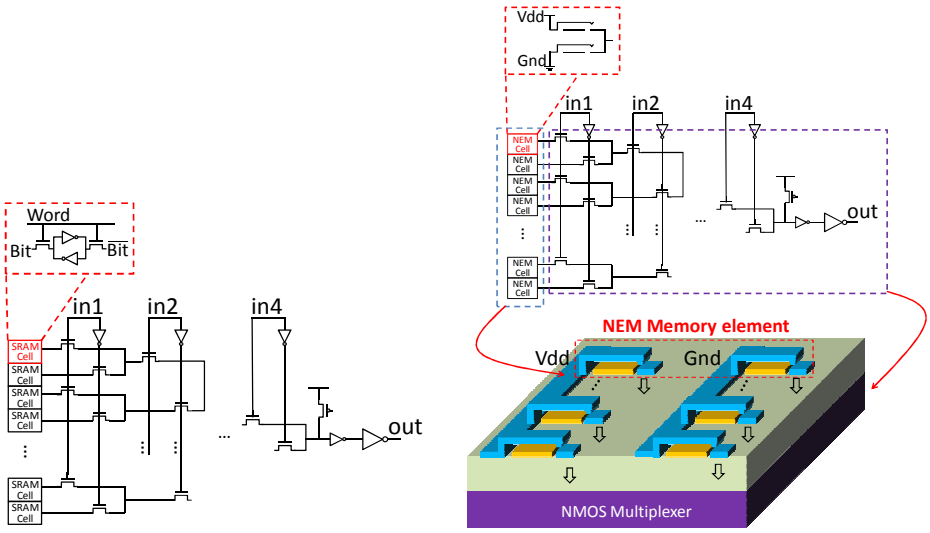



Figure 6.5 Reduction of area, delay, and leakage comparing CMOS-NEM 4-input LUT with traditional CMOS-only 4-input LUT

In CMOS SRAM-based LUT, each CMOS SRAM can be programmed to have its output voltage to be either *Vdd* or *Gnd*, driving the LUT output to either *Vdd* or *Gnd*. Although each NEM relay has two stable states, i.e., open or closed, an NEM relay in open state cannot generate a specific output voltage. In order to provide both *Vdd* and *Gnd* output, two NEM relays are needed to replace one CMOS SRAM cell, as shown in Figure 6.3 (b). For convenience, this

design is called an NEM memory cell. In each NEM memory cell, only one NEM relay will be programmed to the closed state, connecting either *Vdd* or *Gnd* to the output (Data). Each NEM relay can be programmed individually through a half-select programming scheme, as described in [94]. Figure 6.6 shows the idea of replacing CMOS SRAM cells in CMOS-LUT with NEM memory cells. For convenience, the hybrid LUT is called a CMOS-NEM LUT. In this new type of LUT, pre-configured NEM memory cells are used to store corresponding logic values; an NMOS pass-transistor-based multiplexer is used to select the desired output based on input values. As described in Section 6.2, it is possible to stack NEM relays on top of CMOS layers. To achieve this, two fabrication processes are needed: 1) a back-end-of-line (BEOL) process is needed for NEM relays; 2) encapsulation of NEM among metal interconnect layers [97].




(a) (b)

Figure 6.6 (a) Traditional CMOS SRAM-based 4-input LUT; and (b) CMOS-NEM 4-LUT, where NEM memory elements are stacked on top of CMOS

### 6.2.2 NEM Relay as FPGA Routing Switch

Traditional CMOS SRAM-based FPGA uses SRAM controlled NMOS pass-transistors to implement programmable routing switches. As described in [94], both the controlling SRAM cell and the NMOS pass-transistor can be replaced at the same time using just a single NEM relay,

as shown in Figure 6.6, Figure 6.7, and Figure 6.8. This is because the hysteresis properties enable each NEM relay to be used as one memory element. Unlike replacing SRAM cells in LUT, only one NEM relay is needed to replace one NMOS pass-transistor and the corresponding controlling SRAM cell. The NEM relay will be programmed using a half-select programming scheme.

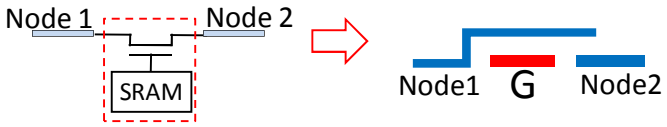



Figure 6.7 CMOS SRAM and corresponding NEM switch

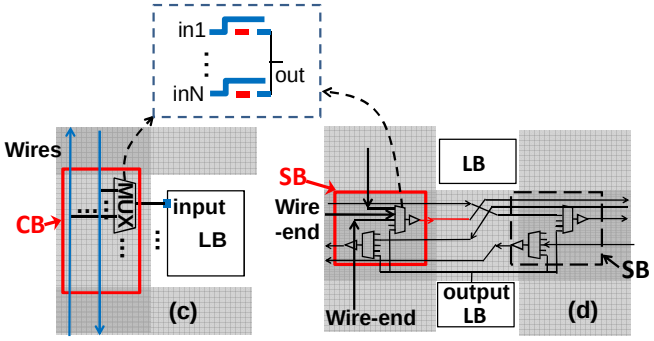



Figure 6.8 NEM relay based FPGA connection block (CB) and switch block (SB)

### 6.2.3 Area Estimation

CMOS baseline FPGA tile area is estimated using the minimum-transistor-width area model [62]. For NEM-CMOS FPGA, tile area is estimated using a similar method. For the 3T NEM relay layout, the same dimension is used as described in [94], which will lead to a pull-in voltage around 0.8 V at the 22 nm technology node ($\lambda$ = 11nm) (see Figure 6.9). Based on the 3T NEM relay layout, the minimum NEM relay layout area can be estimated. Using the minimum NEM relay layout area model and the minimum CMOS transistor area model, the area for the required NEM relays on top of CMOS and the area for the remaining CMOS circuitry were estimated separately. Since NEM relays are stacked on top of CMOS, the final layout area will be determined by the larger area between the CMOS layer and the NEM layer.


Figure 6.9 Layout for a 3T NEM relay (22 nm technology λ = 11nm)

## 6.3 3D NEM FPGA Architecture

Similarly to its 2D counterparts, 3D NEM FPGA adopts the traditional island-style FPGA architecture. Each layer contains a fabric of repeated tiles where each tile consists of one switch block (SB), two connection blocks (CB), and one configurable logic block (CLB).

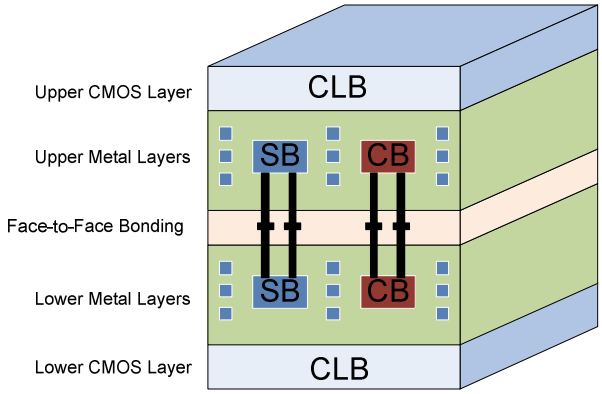



Figure 6.10 Two-layer face-to-face stacking

In this study, the face-to-face bonding process [87] has been adopted to study the benefit of the 3D NEM FPGA. During the face-to-face bonding, metallization layers are joined, and the size of the connecting vias is limited by the accuracy of the layer alignment technique used. Compared to other bonding solutions which use TSVs (through silicon vias) to provide connections between layers, a face-to-face bonding solution can provide relatively high via density and is also relatively easier to fabricate [59], [99]. Figure 6.10 demonstrates the concept of such a face-to-face bonding solution used in this study. As described in Section 6.2, the top and bottom CMOS consists of an addressing circuit, flip-flop, and buffers in the LUT. SRAM cells, SBs, and CBs are implemented by NEM switches and encapsulated within the metal layers as shown in Figure 6.2 . The NEM device does not require a substrate, and therefore, does not occupy the footprint. Vertical connections have been added among SBs as well as among CBs

between the two layers and bonded face-to-face. Details will be described in following sections.

### 6.3.1 Face-to-Face Stacking and Via Density

Face-to-face bonding is a process where two active device layers have been fabricated individually and then aligned and connected in a face-to-face fashion through mechanical and electrical techniques. Compared to TSVs used in face-to-back bonding and multilayer stacking [99]-[100], face-to-face bonding enables high via density. Vias used as vertical interconnects in the face-to-face stacking (named 3D vias) have dimensions similar to the regular vias in the top metal layers in a 2D chip. This high 3D via density enables great layer-to-layer communication bandwidth in the 3D design with the benefits of easier fabrication and less thermal stress compared to the case of 3D stacking of multiple layers [99]-[100]. Therefore, this study is limited to a two-layer face-to-face stacking 3D architecture design with a novel combination of NEM relay and CMOS for higher logic density and performance.

Table 6.1 3D via dimensions and electrical parameters

| | Face-to-Face (projected) |
|---|---|
| Size ($\mu m$) | 1.7 × 1.7 (0.75 × 0.75) |
| Minimum Pitch ($\mu m$) | 2.4 (1.46) |
| Feed-Through Capacitance (*fF*) | 0.74 |
| Series Resistance (Ω) | 116 |

Table 6.1 collected from [1], [101] illustrates various 3D via dimensions and electrical properties. A 3D via in the face-to-face integration can be projected as small as 0.75 μm × 0.75 μm with a pitch of 1.5 μm and unit RC value as shown in Table 6.1.

As described in [94], the layout of a CMOS based FPGA tile occupies an area of 3300 λ × 2200 λ, which is equal to 36.3 μm × 24.2 μm at 22 nm technology. Within each tile of the CMOS-based FPGA, the SBs and CBs take up most of the overall area [46]. For example, if the CLB size is 10 and the BLE size is 4, the global routing (CB + SB) takes 57.4% of the tile area, with the CLB occupying the remaining 42.6% [46]. Therefore, the CLB area can be estimated as 374.2 $\mu m^2$, corresponding to a 19.3 μm dimension. Note that, differently from

using TSV, the 3D via in face-to-face bonding does not go through the silicon layer. Therefore, no area overhead has been added in the CMOS layers compared to the result in [102]. Connection between any devices to the 3D via can be made through regular interconnections. The density of the 3D vias being inserted through the bonding layer is determined by the bonding layer area (equal to the CLB area) and 3D via pitch. Comparing the CLB dimension with the 3D via dimension, the upper bound of via density accommodated within each CLB for vertical communication can be estimated. Figure 6.11 shows an example which has 25 3D vias in each tile. The $5 \times 5$ via array takes an area of $6.75\ \mu m \times 6.75\ \mu m$. Figure 6.11 also shows 10 extra 3D vias used for direct links for faster and dedicated layer-to-layer communication, which will be discussed in the next section.

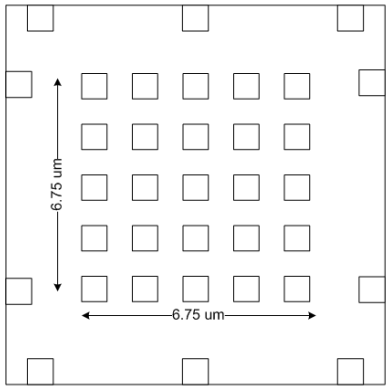



Figure 6.11 CLB area and 3D via density

### 6.3.2 3D Switch Block

The 3D switch block is a critical component in the 3D FPGA architecture, which provides not only normal routing connections between the horizontal routing channels but also connections between the two device layers vertically.

Figure 6.12 shows two vertically stacked tiles and the SB and CB designs sandwiched in between. Each CMOS layer has its own metal layers (upper metal layers and lower metal layers in Figure 6.10). The top metal layers of the two face-to-face stacks are connected through NEM 3D switch blocks incorporating 3D vias. The 3D switch block is an MUX-based design, which is widely used in modern FPGA architectures. Each wire in the routing channel is unidirectional

and driven by an MUX. Inputs of a driver MUX come from different channels of different directions. In the 3D case, the MUX also contains inputs from the vertical direction.

Figure 6.12 shows the path from an output of CLB 3 to a CLB 2 input, assuming the single-driver architecture as used in VPR 5.0 [95]. The output of CLB 3 is connected to a switch point underneath. By configuring the MUX accordingly, the signal can be routed through the MUX *a* to the connection block of CLB 2, then to the CLB input MUX. Routing on the same layer can be carried out in the same way by configuring MUX connections.

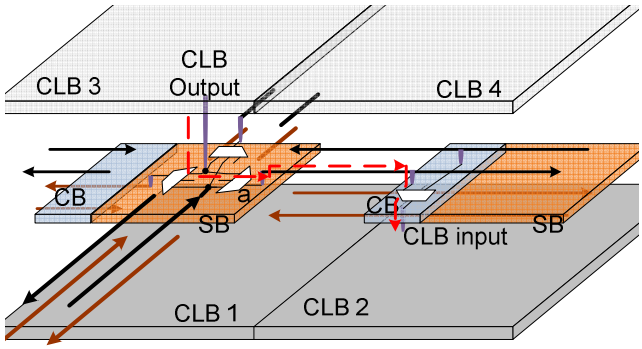



Figure 6.12 3D Stacking with switch block and connection block

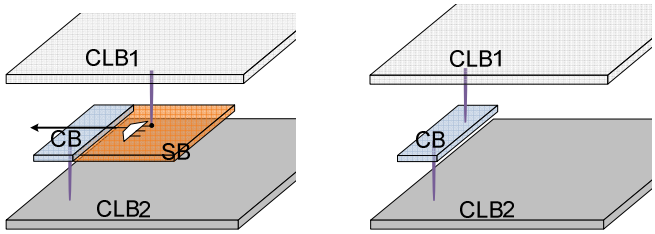
Note that in Figure 6.12, only one possible 3D connection is demonstrated. In reality, each outgoing wire in a switch block will be driven by an MUX and each incoming wire will be connected to the inputs of several MUXes.

### 6.3.3 Direct Links

As observed in Figure 6.13 (a), if two vertically stacked CLBs need to communicate with each other, a routing path would go through switch block MUXes and connection block MUXes. Given the face-to-face bonding with short layer-to-layer distance, going through several MUXes is costly. This is the motivation to provide another architectural enhancement by including direct connections between two layers.

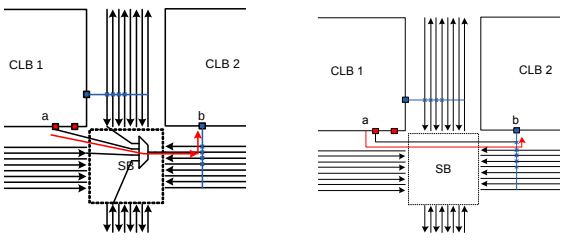
As shown in Figure 6.13 (b), a direct connection between an output of CLB 1 and the CB of CLB 2 is created. This connection bypasses the switch block and saves an MUX delay as well

as the wire RC load from the routing track. Figure 6.14 (a) shows the equivalent topology in 2D FPGA. In this example, the MUX in Figure 6.14 (a) has four inputs: one from CLB 1 output and the other three from routing channels respectively. Figure 6.14 (b), on the other hand, shows the proposed direct connection scenario between the two CLBs. The output pin of CLB 1 is connected to the connection block of CLB 2 directly. To have a better utilization rate of these direct links, the idea is extended so that each CLB can talk to five neighbors in the other layer as illustrated in Figure 6.15. The direct links are inserted in a balanced way on four sides of each CLB. Figure 6.15 shows the case when the cluster size is 10. Two extra links are inserted in between the CLB pair, where one is directly on top of the other. The overhead of direct link is the slight increase of the size of the CLB input MUX slightly. For example, if an architecture with channel width 100 and *Fc* = 0.5 (50% of wires in wire channel are connected to a CB input), a 50-to-1 MUX is required at each CLB input pin. By adding 10 direct links as shown in Figure 6.15, 2 or 3 (1 from right above or below and 2 from the CLB with 1 grid offset in different layer) more inputs need to be added on each side of the CLB, which increases the MUX size to 52 or 53, respectively.




(a) (b)

Figure 6.13 Connection of two vertically stacked CLBs: (a) without direct link; and (b) with direct link




(a) (b)

Figure 6.14 (a) Regular length-1 connection; and (b) direct link

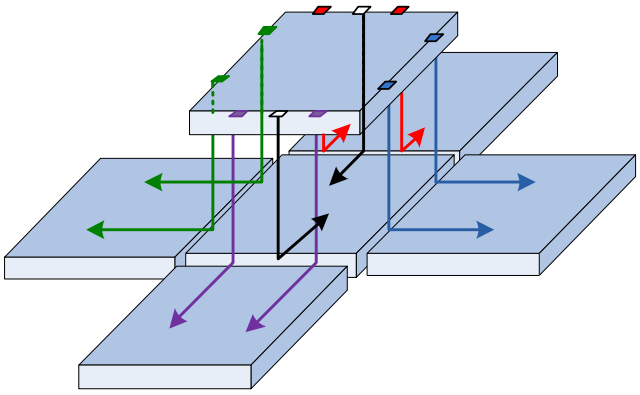


Figure 6.15 Direct links between neighboring CLBs

## 6.4 CAD Flow

In this work, a timing-driven CAD flow has been developed (Figure 6.16). Each benchmark circuit goes through technology independent logic optimization using SIS [63] and is technology-mapped to *K*-LUTs using DAOmap [64], which is a popular performance-driven mapper working on area minimization as well. The mapped netlist then feeds into T-VPACK, which performs timing-driven packing (i.e., clustering LUTs into CLBs). The major contribution in this work is the final step, which performs placement and routing for the design targeting this 3D architecture. The new placement and routing engine is adopted from and developed in VPR 5.0 [95].

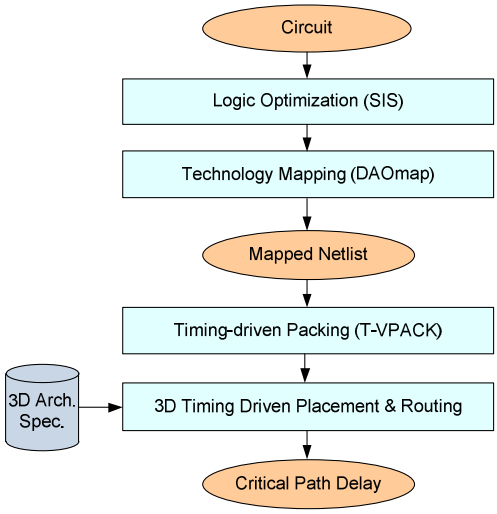



Figure 6.16 CAD evaluation flow

### 6.4.1 3D Architecture Generation

VPR is a famous FPGA placement and routing tool and has been widely used within the research community. One of VPR's advantages is that it supports flexible FPGA architecture exploration, and users can easily redefine the architecture in the architecture file. This study enhanced the existing architecture by introducing additional 3D-related options to guide the 3D FPGA architecture generation. Four new options have been added:

- *max_3d_vias_ per_ tile*

  This parameter sets an upper limit of the number of the 3D vias that could be inserted within each tile. A 3D via has a relatively large pitch. This value needs to be extracted based on a detailed area model to make sure that there would be enough space to accommodate all 3D vias in a tile.

- *3d_via_percentage*

  This parameter defines the number of wires in a wire channel that are 3D capable. For example, considering the architecture with a channel width 100, setting *3d_via_percentage* to 0.25 will create 25 3D vias within each tile. The detailed process of 3D via creation will be discussed below. Note that this value will be overwritten by *max_3d_vias_per_tile* if it exceeds the max value.

- *3d_via_parameter*

  This option defines the resistance and capacitance value of a 3D via. These values should be derived from the via RC model and the 3D FPGA architecture information, i.e., the distance between two layers and the bonding process of 3D stacking.

- *direct_link*

  This Boolean option indicates whether direct links will be inserted or not.

Figure 6.17 is an example showing how 3D connections have been made. In the VPR 5.0 single-driver architecture, each outgoing wire in SB is driven by an MUX and each incoming wire will be connected to a set of MUXes based on the SB model. For example, for regular VPR, input in_1 will connect to three other MUXes on the other three sides, respectively. In this 3D

architecture, in_1 can also connect to all the four sides on the top layer. Similarly, the upper layer wire in_2 can also connect to four outgoing wires on the bottom layer.

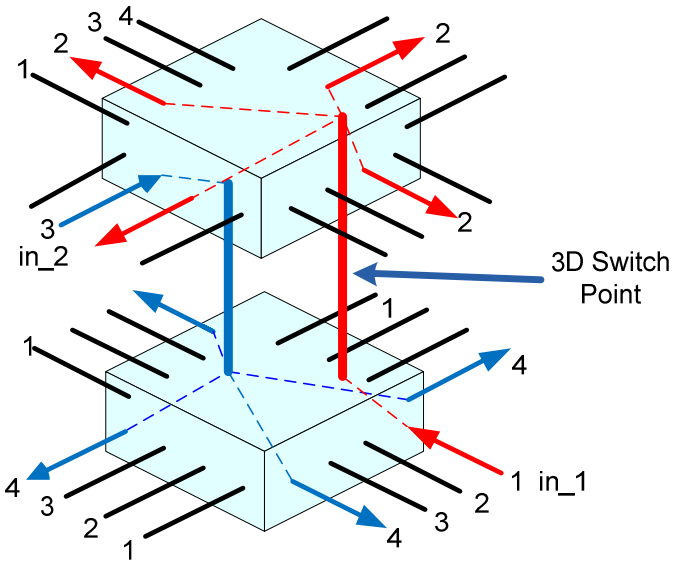



Figure 6.17 3D via creation

The wires which have 3D capability are evenly distributed across the wire channel. If one takes channel width 100 and *3d_via_percentage* 0.25 as an example again, 25 3D vias in total will be generated: 12 out of the 25 vias have the direction from the bottom to the top layer and the other 13 have the direction from the top to the bottom layer. The 12 or 13 vertical connections will be evenly assigned into wire channels. For example, if wires with odd wire ID (e.g. 1, 3, 5, 7…49) are incoming wires to a SB, then the 12 3D vias will be added to wire 1, 5, 9…49, respectively. Figure 6.17 demonstrates a simple example with four wires in the channel numbered from 1 to 4 clockwise. Incoming wire in_1 with wire ID 1 in the bottom layer is connected to outgoing wires with wire ID 2 on each side on the top layer. Similarly, incoming wire in_2 with wire ID 3 in the top layer is connected to outgoing wires with wire ID 4 on each side on the bottom layer. The percentage of switch points that have 3D capability is an architecture input defined in the architecture file. In the meantime, the maximum number of 3D vias which is determined by the bonding layer area has to be considered as well. Therefore, in most situations, the percentage 3D switch point has an upper limit.

If the option direct link is enabled, extra 3D vias as direck links are added from CLB output to CLB input as shown in Figure 6.13.

### 6.4.2 3D Placement and Routing

To carry out 3D placement and routing, the first step is the construction of the 3D routing graph. In VPR, each component is represented as a routing node, and possible connections between components are represented as routing edges. 3D routing graph construction is the process of linking appropriate routing nodes in different layers and change values, such as outgoing edge array, resistance and capacitance. The detailed algorithm is shown in Figure 6.18.

```
Follow VPR's original process, create planar routing graph for each layer
        separately.
Compute number and locations that 3D vias need to be inserted.
for (each 3D via location, i) do
    Find wires segs that need to be connected in another layer;
    for (each wires seg j that needs to be connected to i) do
        Find routing node index of seg j;
        Add outgoing routing edge (i, j) on node i;
        Update R, C values on routing node j;
    end
end

If (direct link enabled)
for (each CLB, i) do
    Find neighboring CLBs on corresponding layer;
    for (each CLB input j that needs to be connected to i) do
        Find routing node index of  j;
        Add outgoing routing edge (i, j) on node i;
        Update R, C values on routing node j;
    end
end
```

Figure 6.18 Process of 3D routing graph construction

A 3D routing graph is generated based on two individual 2D routing graphs, which represent two stacking layers, respectively. However, each routing node in these two planar graphs has a unique node ID. The amount and location of 3D vias are then calculated based on the flow described in previous sections. Since each wire segment has a unique routing node ID,

routing edges then can be added to represent 3D vias. The resistance and capacitance values of the destination routing node can then be updated to incorporate 3D via resistance and capacitance values from Table 6.1 for accurate timing analysis.

VPR placement is based on simulated annealing algorithm. During simulated annealing, random swaps of logic blocks are accepted or rejected based on a cost function and an annealing temperature.

3D placement takes a similar approach, but the random swaps are carried out both within a layer and between layers. To speed up the process of placement, VPR pre-calculates a delay matrix for net delay lookup.

$$\text{NetDelay} = \text{DelayMatrix}[\Delta X, \Delta Y] \tag{6.1}$$

where $\Delta X$ and $\Delta Y$ are the Manhattan distances between two pins of the net.

In the 3D case, the pre-calculated delay matrix is expanded into three dimensions.

$$\text{NetDelay}_{3D} = \text{DelayMatrix}[\Delta X, \Delta Y, \Delta Z] \tag{6.2}$$

If $[\Delta X, \Delta Y]$ is [0, 0], [1, 0] or [0, 1] and $\Delta Z$ is not 0, it means these two pins can be connected by a direct link as shown in Figure 6.15. When a direct link is used, $\text{DelayMatrix}[\Delta X, \Delta Y, \Delta Z]$ is computed based on the RC delay of the direct link via. Otherwise $\text{DelayMatrix}[\Delta X, \Delta Y, \Delta Z]$ is computed through the 3D switch block routing.

During placement, VPR uses the value $\Delta$net_delay to evaluate each swap,

$$\Delta\text{net\_delay} = \sum_i \text{net\_delay}_i \tag{6.3}$$

where *i* is the nets being affected by this swap and $\Delta\text{net\_delay}_i$ is computed based on $\Delta X$ and $\Delta Y$ before and after swap.

In 3D placement with direct links, $\Delta\text{net\_delay}_i$ is looked up in 3D $\text{DelayMatrix}[\Delta X, \Delta Y, \Delta Z]$. If two locations are directly linked, the smaller net delay will be loaded. For example, the case $[\Delta X, \Delta Y, \Delta Z] = [1, 0, 0]$ before swap and [0, 0, 1] after swap indicates a placement where two connected CLB are placed side by side in the same layer before swap, and moved and stacked vertically after swap. Directly linked [0, 0, 1] placement will have a smaller delay value. Therefore, solution [0, 0, 1] will be preferred and this swap will be

accepted.

VPR has a very good annealing schedule, where the window of random swapping is dynamically adjusted based on success rate. At high temperature, random swap can be made across the chip area, and at lower temperature, swap will be made in a small region only. The overall range of swap is guided by the Equation (6.4) [62]:

$$\text{rlim} = \text{rlim} * (1. - 0.44 + \text{success\_rat}) \quad (6.4)$$

In this experiment, It was found that for 3D placement the optimal value of *rlim* is changed as follows:

$$\text{rlim} = \text{rlim} * (1. - 0.25 + \text{success\_rat}) \quad (6.5)$$

This means 3D placement achieves better results at a lower rate of shrinking the window where two blocks are picked and swapped as compared to 2D placement.

VPR routing is based on the Pathfinder negotiated congestion algorithm [62]. The routing process is iterative. During the first iteration, the criticality of each pin in every net is set to 1 (highest criticality) to minimize the delay of each pin. If congestion exists, more routing iterations are performed until all of the overused routing resources are resolved. At the end of each routing iteration, criticality and congestion information are updated before the next iteration starts. 3D routing takes the same approach but operates on a completely new 3D routing graph generated from the 3D architecture file, as described in Section 6.4.1.

## 6.5 Experimental Results

### 6.5.1 Experiment Setup

To evaluate the 3D NEM FPGA, a fixed LUT input size $K$ = 4 and a logic cluster size of $N$ = 10 were used. It is shown in [62] that a mixture of interconnects with different lengths can provide improved performance. This study evaluated the architecture with the following wire segment mixture: 30% length-1 wires, 40% length-2 wires, and 30% length-4 wires. The CAD flow shown in Figure 6.16 was run for different FPGA architectures using the standard set of 20

MCNC benchmarks as well as 5 big benchmarks from VPR 5.0. Note that the flow developed in this study is flexible and capable of supporting different architecture settings.

### 6.5.2 Results and Discussions

This section quantifies the overall performance improvements of the 3D NEM FPGA over the baseline 2D CMOS FPGA and the 2D NEM FPGA. Table 6.2 details the performance comparison results. The performance improvement of 3D NEM FPGA is achieved from a combination of NEM-based LUT, NEM-based routing design, and the 3D architecture.

On average, 2D NEM FPGA provides a 19.51% delay reduction comparing to the baseline. This delay reduction is achieved by the reduced tile area using the NEM design, which reduces the global wire length. Replacing the SRAM-based LUT with the NEM-based LUT also contributes to delay reduction.

3D NEM FPGA provides a 37.63% delay reduction comparing to the baseline. The performance gain comes from the 3D stacking, which dramatically reduces the FPGA footprint. By adding direct link into the scope, an additional 9% delay reduction can be achieved (a 46.34% reduction comparing to the baseline).

Overall, using NEM devices and 3D stacking produces very significant performance gains for 3D NEM FPGA. In addition, vertical direct links can offer an additional performance improvement.

Table 6.2 Performance comparison of CMOS and NEM FPGA (unit: ns)

| | CMOS | 2D NEM | | 3D NEM without direct link | | 3D NEM with direct link | |
|---|---|---|---|---|---|---|---|
| | Crit.Path | Crit.Path | % reduction | Crit.Path | % reduction | Crit.Path | % reduction |
| alu4 | 2.81E−09 | 2.09E−09 | 25.49% | 1.64E−09 | 41.52% | 1.40E−09 | 50.15% |
| apex2 | 3.16E−09 | 2.49E−09 | 21.34% | 1.98E−09 | 37.39% | 1.71E−09 | 45.86% |
| apex4 | 3.15E−09 | 2.70E−09 | 14.51% | 1.88E−09 | 40.31% | 1.61E−09 | 49.10% |
| bigkey | 1.59E−09 | 1.24E−09 | 21.70% | 9.16E−10 | 42.27% | 8.01E−10 | 49.54% |
| clma | 5.85E−09 | 5.06E−09 | 13.40% | 3.64E−09 | 37.73% | 3.34E−09 | 42.91% |
| des | 2.84E−09 | 2.28E−09 | 19.82% | 1.75E−09 | 38.54% | 1.58E−09 | 44.47% |
| diffeq | 3.97E−09 | 3.25E−09 | 18.02% | 2.31E−09 | 41.71% | 2.02E−09 | 49.04% |
| dsip | 1.42E−09 | 1.27E−09 | 10.85% | 8.88E−10 | 37.51% | 8.00E−10 | 43.69% |
| elliptic | 5.95E−09 | 4.54E−09 | 23.78% | 3.60E−09 | 39.53% | 3.13E−09 | 47.48% |
| ex1010 | 4.13E−09 | 3.44E−09 | 16.54% | 2.69E−09 | 34.71% | 2.33E−09 | 43.57% |
| ex5p | 3.44E−09 | 2.83E−09 | 17.70% | 2.41E−09 | 30.11% | 2.01E−09 | 41.49% |
| frisk | 7.17E−09 | 6.21E−09 | 13.43% | 5.08E−09 | 29.09% | 3.70E−09 | 48.38% |
| misex3 | 2.65E−09 | 2.07E−09 | 21.85% | 1.59E−09 | 39.87% | 1.36E−09 | 48.66% |
| pdc | 5.61E−09 | 4.45E−09 | 20.65% | 3.59E−09 | 36.11% | 3.19E−09 | 43.20% |
| s298 | 5.90E−09 | 4.76E−09 | 19.41% | 3.34E−09 | 43.48% | 3.03E−09 | 48.67% |
| s38417 | 4.15E−09 | 3.25E−09 | 21.57% | 2.74E−09 | 34.06% | 2.33E−09 | 43.79% |
| s38584.1 | 3.35E−09 | 2.39E−09 | 28.74% | 2.03E−09 | 39.36% | 1.64E−09 | 51.03% |
| seq | 2.97E−09 | 2.54E−09 | 14.79% | 1.97E−09 | 33.64% | 1.68E−09 | 43.38% |
| spla | 3.91E−09 | 3.01E−09 | 22.88% | 2.35E−09 | 39.93% | 2.11E−09 | 45.90% |
| tseng | 3.92E−09 | 3.30E−09 | 15.82% | 2.64E−09 | 32.60% | 2.03E−09 | 48.26% |
| rs | 3.71E−09 | 2.83E−09 | 23.94% | 2.14E−09 | 42.38% | 1.84E−09 | 50.57% |
| paj_top_hierarchy_no_mem | 3.07E−08 | 2.45E−08 | 20.23% | 1.97E−08 | 35.87% | 1.75E−08 | 42.96% |
| mac2 | 1.55E−08 | 1.21E−08 | 21.94% | 9.44E−09 | 38.99% | 8.11E−09 | 47.64% |
| cf_cordic_v_18_18_18 | 2.74E−09 | 2.16E−09 | 21.11% | 1.70E−09 | 38.03% | 1.50E−09 | 45.15% |
| des_perf | 1.88E−09 | 1.54E−09 | 18.23% | 1.21E−09 | 35.88% | 1.06E−09 | 43.72% |
| Ave. | 5.30E−09 | 4.25E−09 | 19.51% | 3.33E−09 | 37.63% | 2.87E−09 | 46.34% |

# CHAPTER 7
# CONCLUSION

This dissertation has introduced and discussed three novel reconfigurable architectures, 3D nFPGA, FPCNA, and NEM FPGA. 3D nFPGA architecture utilizes 3D integration techniques and new nanoscale materials. The combination of these two leading technologies shows a great potential for innovation and technology breakthroughs. The evaluation result demonstrates that the proposed 3D nFPGA is able to provide a 2.65× Fmax advantage over the traditional CMOS baseline 2D FPGA with a small total power overhead.

FPCNA is a CNT-based design including novel LUTs and switching boxes. An effective variation aware CAD flow was developed, which handles arbitrary delay distributions using variation aware placement and routing. Experimental results show that FPCNA offers a 5× footprint reduction and a 2.75× performance gain (targeting a 95% yield) compared to a baseline CMOS FPGA at the same technology node. These first results of nano 3D reconfigurable architectures are very encouraging and provide motivation for further study, including thermal behavior and architectural reliability.

NEM FPGA architecture is a hybrid architecture of nanoelectromechanical relays and CMOS devices. Taking advantage of NEM relay, which can be encapsulated into metal layers, face-to-face stacking is applied to this architecture to pursue high performance. In addition, a new concept called direct link has been evaluated to further enhance the benefits of this new architecture. Compared to the CMOS baseline, 2D NEM FPGA provides a 19.51% performance enhancement due to the new NEM LUT design. 3D NEM FPGA is able to achieve a 37.63% delay reduction compared to the baseline. This performance gain comes from the NEM device as well as the 3D architecture, which dramatically reduces the FPGA footprint. Direct link is able to provide an additional 9% delay reduction, which is a 46.34% total reduction compared to the baseline.

Customized design automation flows—including a comprehensive SSTA engine, variation aware placement and routing, and 3D placement and routing—have been developed to evaluate different architectures. The SSTA engine is designed to consider both intra-die and inter-die variation, 2D and 3D spatial correlated and random variation, and variation with Gaussian and non-Gaussian distribution. Using this SSTA engine, 2D and 3D SSTA aware placement and routing algorithms have been developed for improving performance yield.

In summary, this dissertation presents research on nano FPGA architecture and CAD. The results offer insights on FPGA architecture exploration including CMOS nano hybrids and 3D stacking. Key architectural parameters have been discovered to improve overall chip performance in terms of delay, power, yield, and reliability. CAD tools are developed to support and validate different concepts. A concrete step in nano FPGA research, this dissertation provides guidance for the development of emerging nanotechnologies.

# REFERENCES


[1] International technology roadmap for semiconductors, 2007. [Online]. Available: http://public.itrs.net

[2] C. Ababei, P. Maidee, and K. Bazargan, “Exploring potential benefits of 3D FPGA integration,” in *Field Programmable Logic and Application*, vol. 3203, S. Vernalde, Ed. Heidelberg, Germany: Springer, 2004, pp. 874-880.

[3] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, “3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration,” in *Proceedings of the IEEE*, vol. 89, no. 5, pp. 602-633, 2001.

[4] M. Lin, A. El Gamal, Y. C. Lu, and S. Wong, “Performance benefits of monolithically stacked 3D-FPGA,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Array*, 2006, pp. 113-122.

[5] A. Bachtold et al. “Scanned probe microscopy of electronic transport in carbon nanotubes,” *Physical Review Letters*, vol. 84, pp. 6082–6085, Jun. 2000.

[6] J. Kong, E. Yenilmez, T. W. Tombler, W. Kim, and H. Dai, “Quantum interference and ballistic transmission in nanotube electron wave-guides,” *Physical Review Letters*, vol. 87, no. 10, pp. 106801-1-106801-4, Aug. 2001.

[7] W. Liang, M. Bockrath, D. Bozovic, J. H. Hafner, M. Tinkham, and H. Park, “Fabry-Perot interference in a nanotube electron waveguide,” *Nature*, vol. 441, pp. 665–669, Jun. 2001.

[8] A. Naeemi and J. D. Meindl, “Monolayer metallic nanotube interconnects: Promising candidates for short local interconnects,” *IEEE Electron Device Letters*, vol. 26, pp. 544-546, Aug. 2005.

[9] A. Naeemi, R. Sarvari, and J. D. Meindl, “Performance comparison between carbon nanotube and copper interconnects for gigascale integration (GSI),” *IEEE Electron Device Letters*, vol. 26, pp. 84-86, Feb. 2005.

[10] N. Srivastava, R. V. Joshi, and K. Banerjee, “Carbon nanotube interconnects: implications for performance, power dissipation and thermal management,” in *Proceedings of IEEE International Electron Devices Meeting*, 2005, pp. 249-252.

[11] B. Kaustav, L. Sheng-Chih, and S. Navin, “Electrothermal engineering in the nanometer era: from devices and interconnects to circuits and systems,” in *Proceedings of Asia South Pacific Design Automation Conference*, Jan. 2006, pp 8-14.

[12] M. Dresselhaus, G. Dresselhaus, and P. Avouris, Eds. *Carbon Nanotubes: Synthesis, Structure Properties and Applications*. Berlin, Germany: Springer-Verlag, 2001.

[13] A. Bachtold, P. Hadley, T. Nakanishi, and C. Dekker, “Logic circuits with carbon nanotube transistors,” *Science*, vol. 294, no. 5545, pp. 1317-1320, Nov. 2001.

[14] V. Derycke, R. Martel, J. Appenzeller, and P. Avouris, “Carbon Nanotube Inter- and Intramolecular Logic Gates,” *Nano Letters*, vol. 1, no. 9, pp. 453–456, 2001.

[15] S. J. Wind, J. Appenzeller, R. Martel, V. Derycke, and P. Avouris, “Vertical scaling of carbon nanotube field-effect transistors using top gate electrodes,” *Applied Physics Letters*, vol. 80, no. 20, pp. 3817-3819, May 2002.

[16] A. Javey et al. “High-k dielectrics for advanced carbon nanotube transistors and logic gates,” *Nature Materials*, vol. 1, no. 4, pp. 241-246, Dec. 2002.

[17] S. J. Kang et al. “High-performance electronics using dense, perfectly aligned arrays of single-walled carbon nanotubes,” *Nature Nanotechnology*, vol. 2, no. 4, pp. 230-236, Apr. 2007.

[18] M. Fuhrer, B. M. Kim, T. Dürkop, and T. Brintlinger, “High-mobility nanotube transistor memory,” *Nano Letters*, vol. 2, no. 7, pp. 755–759, May 2002.

[19] S. Rosenblatt, Y. Yaish, J. Park, J. Gore, V. Sazonova, and P. L. McEuen, “High performance electrolyte gated carbon nanotube transistors,” *Nano Letters*, vol. 2, no. 8, pp. 869–872, Jul. 2002.

[20] C. Zhou, J. Kong, and H. Dai, “Electrical measurements of individual semiconducting single-walled nanotubes of various diameters,” *Applied Physics Letters*, vol. 76, no. 12, pp. 1597–1599, Jan. 2000.

[21] J. Deng et al. “Carbon nanotube transistor circuits: Circuit-level performance benchmarking and design options for living with imperfections,” in *Proceedings of International Solid-State Circuits Conference*, Jun. 2007, pp. 70-71.

[22] A. DeHon and K. K. Likharev, “Hybrid CMOS/Nanoelectronic digital circuits: Devices, architectures, and design automation,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2005, pp. 375-382.

[23] D. B. Strukov and K. K. Likharev, “A reconfigurable architecture for hybrid CMOS/Nanodevice circuits,” in *Proceedings of the International Symposium on Field Programmable Gate Arrays*, 2006, pp. 131-140.

[24] D. B. Strukov and K. K. Likharev, “CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices,” *Nanotechnology*, vol. 16, no. 6, pp 888-900, Jun. 2005.

[25] A. K. Geim, and K. S. Novoselov, “The rise of graphene,” *Nature Materials*, vol. 6, pp. 183-191, 2007.

[26] S. Iijima, “Carbon nanotubes: Past, present, and future,” *Physica B: Physics of Condensed Matter*, vol. 323, no. 1-4, pp. 1-5, Oct. 2002.

[27] J. P. Bourgoin, “Carbon nanotubes”, in *Nanostructures: Fabrication and analysis*. N. Hitoshi, Ed. Berlin, Germany: Springer, 2007.

[28] J. H. Chen, C. Jang, S. Xiao, M. Ishigami, and M. S. Fuhrer, “Intrinsic and extrinsic performance limits of graphene devices on $SiO_2$,” *Nature Nanotechnology*, vol. 3, no. 4, pp. 206-209, Apr. 2008.

[29] R. Martel, T. Schmidt, H. R. Shea, T. Hertel, and P. Avouris, “Single- and multi-wall carbon nanotube field-effect transistors,” *Applied Physics Letters*, vol. 73, no. 17, pp. 2447-2449, Oct. 1998.

[30] S. J. Tans, A. R. M. Verschueren, and C. Dekker, “Room-temperature transistor based on a single carbon nanotube,” *Nature*, vol. 393, no. 6680, pp. 49-52, May 1998.

[31] P. G. Collins, M. S. Arnold, and P. Avouris, “Engineering carbon nanotubes and nanotube circuit using electrical breakdown,” *Science*, vol. 292, no. 5517, pp. 706-709, Apr. 2001.

[32] J. Deng and H. S. P. Wong, “A compact SPICE model for carbon nanotube field effect transistors including non-idealities and its application - part I: Model of the intrinsic channel region,” *IEEE Trans. Electron Devices*, vol. 54, no. 12, pp. 3186-3194, Dec. 2007.

[33] J. Deng and H. S. P. Wong, “A compact SPICE model for carbon nanotube field effect transistors including non-idealities and its application - part II: Full device model and circuits performance benchmarking,” *IEEE Trans. Electron Devices*, vol. 54, no. 12, pp. 3195-3205, Dec. 2007.

[34] Z. Chen et al. “An integrated logic circuit assembled on a single carbon nanotube,” *Science*, vol. 311, no. 5768, p. 1735, Mar. 2006.

[35] T. Rueckes, K. Kim, E. Joselevich, G. Y. Tseng, C. Cheung, and C. M. Lieber, “Carbon nanotube-based nonvolatile random access memory for molecular computing,” *Science*, vol. 289, no. 5476, pp. 94-97, Jul. 2000.

[36] J. W. Ward et al. “A nonvolatile nanoelectromechanical memory element utilizing a fabric of carbon nanotubes,” in *Proceedings of Non-Volatile Memory Technology Symposium*, Nov. 2004, pp. 34-38.

[37] Y. Massoud and A. Nieuwoudt, “Modeling and design challenges and solutions for carbon nanotube-based interconnect in future high performance integrated circuits,” *ACM Journal on Emerging Technologies in Computing Systems*, vol. 2, no. 3, pp. 155-196, Jul. 2006.

[38] B. Q. Wei, R. Vajtai, and P. M. Ajayan, “Reliability and current carrying capacity of carbon nanotubes,” *Applied Physics Letters*, vol. 79, no. 8, pp. 1172-1174, Jul. 2001.

[39] N. Srivastava and K. Banerjee, “Performance analysis of carbon nanotube interconnects for VLSI applications,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2005, pp. 383-390.

[40] M. Nihei, M. Horibe, A. Kawabata, and Y. Awano, “Carbon nanotube vias for future LSI interconnects,” in *Proceedings of IEEE International Interconnect Technology Conference*, Jun. 2004, pp. 251-253.

[41] Fujitsu Corp., “Fujitsu reports progress towards carbon nanotube interconnects for 32nm,” *Solid State Technology*, vol. 49, no. 12, pp. 15-16, Nov. 2006.

[42] L. Zhu, Y. Xiul, D. W. Hess, and C. P. Wong, “Growth of aligned carbon nanotube arrays for electrical interconnect,” in *Proceedings of Electronics Packaging Technology Conference*, Dec. 2005, pp. 646-651.

[43] S. Kaeriyama et al. “A nonvolatile programmable solid-electrolyte nanometer switch,” *IEEE Journal of Solid-State Circuits*, vol.40, no.1, pp. 168-176, Jan. 2005.

[44] Y. Chen et al. “Nanoscale molecular-switch crossbar circuits,” *Nanotechnology*, vol. 14, no. 4, pp. 462-468, Apr. 2003.

[45] M. D. Austin et al. “Fabrication of 5 nm linewidth and 14 nm pitch features by nanoimprint lithography,” *Applied Physic Letters*, vol. 84, no. 26, pp. 5299-5301, Jun. 2004.

[46] E. Ahmed and J. Rose, “The effect of LUT and cluster size on deep-submicron FPGA performance and density,” *IEEE Transactions on VLSI*, vol. 12, no. 3, pp. 288-298, Mar. 2004.

[47] F. Li, D. Chen, L. He, and J. Cong, “Architecture evaluation for power-efficient FPGAs,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Array*, Nov. 2003, pp. 175-184.

[48] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, “Power modeling and characteristics of field programmable gate arrays,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 11, pp. 1712-1724, Nov. 2005.

[49] S. Chiricescu, M. Leeser, and M. M. Vai, “Design and analysis of a dynamically reconfigurable three-dimensional FPGA,” *IEEE Transactions on VLSI*, vol. 9, no. 1, pp. 186-196, Feb. 2001.

[50] A. Rahman, S. Das, A. P. Chandrakasan, and R. Reif, “Wiring requirement and three-dimensional integration technology for field programmable gate arrays,” *IEEE Transactions on VLSI*, vol. 11, no. 1, pp. 44-54, Apr. 2003.

[51] S. C. Goldstein and M. Budiu, “NanoFabric: Spatial computing using molecular electronics,” in *Proceedings of International Symposium on Computer Architecture*, Jul. 2001, pp. 178-189.

[52] A. DeHon, “Nanowire-based programmable architectures,” *ACM Journal on Emerging Technologies in Computing Systems*, vol. 1, no. 2, pp. 109-162, Jul. 2005.

[53] A. Gayasen, N. Vijaykrishana, and M. J. Irwin, “Exploring technology alternatives for nano-scale FPGA interconnects,” in *Proceedings of Design Automation Conference*, Jun. 2005, pp. 921-926.

[54] R. M. P. Rad and M. Tehranipoor, “A New Hybrid FPGA with Nanoscale Clusters and CMOS Routing,” in *Proceedings of Design Automation Conference*, Jun. 2006, pp. 727-730.

[55] G. Snider and S. Williams, “Nano/CMOS architecture using a field-programmable nanowire interconnect,” *Nanotechnology*, vol. 18, no. 3, pp. 11-24, Jan. 2007.

[56] Nantero Corp. NRAMTM. [Online]. Available: http://www.nantero.com/tech.html

[57] W. Zhang, N. Jha, and L. Shang, “NATURE: A hybrid nanotube/CMOS dynamically reconfigurable architecture,” in *Proceedings of Design Automation Conference*, Jun. 2006, pp. 711-716.

[58] B. Gojman, R. Rubin, C. Pilotto, and A. Dehon, “3D nanowire-based programmable logic,” in *Proceedings of Nanonet Conference*, Sep. 2006, pp. 54-61.

[59] W. R. Davis et al. “Demystifying 3D ICs: The pros and cons of going vertical,” *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 498-510, Nov. 2005.

[60] J. Hone et al. “Electrical and thermal transport properties of magnetically aligned single wall carbon nanotube films,” *Applied Physics Letters*, vol. 77, no. 5, pp. 666-668, Jul. 2000.

[61] G. Snider, P. Kuekes, and R. S. Williams, “CMOS-like logic in defective nanoscale crossbars,” *Nanotechnology*, vol. 15, no. 8, Aug. 2004.

[62] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs.* Boston, MA, Kluwer Academic Publishers, 1999.

[63] E. M. Sentovich et al. (1992). SIS: A System for Sequential Circuit Synthesis. Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/2010.html

[64] D. Chen and J. Cong, "DAOmap: A depth-optimal area optimization mapping algorithm for FPGA designs," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2004, pp. 752-759.

[65] A. Nieuwoodt and Y. Massoud, "Evaluating the impact of resistance in carbon nanotube bundles for VLSI interconnect using diameter-dependent modeling techniques," *IEEE Transactions on Electron Devices*, vol. 53, no. 10, pp. 2460-2466, Oct. 2006.

[66] A. Raychowdhury and K. Roy, "Circuit modeling of carbon nanotube interconnects and their performance estimation in VLSI design", in *Proceedings of the International Workshop on Computational Electronics*, Nov. 2004, pp. 24-27.

[67] A. Raychowdhury and K. Roy, "Modeling of metallic carbon-nanotube interconnects for circuit simulations and a comparison with Cu interconnects for scaled technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 1, pp. 58-65, Jan. 2006.

[68] C. Dong and W. Wang, "Exploring carbon nanotubes and NiSi nanowires as on-chip interconnections," in *Proceedings of the International Symposium on Circuits and Systems*, May 2006, pp. 3510-3513.

[69] Y. Zhou, S. Thekkel, and S. Bhunia, "Low power FPGA design using hybrid CMOS-NEMS approach," in *Proceedings of the International Symposium on Low Power Electronics and Design*, Aug. 2007, pp. 14-19.

[70] S. J. Kang et al. "Printed multilayer superstructures of aligned single-walled carbon nanotubes for electronic applications," *Nano Letters*, vol. 7, no. 11, pp. 3343-3348, Nov. 2007.

[71] N. Patil, A. Lin, E. Myers, H.S.-P. Wong, and S. Mitra, "Integrated wafer-scale growth and transfer of directional carbon nanotubes and misaligned carbon nanotube immune logic structures," in *Proceedings of the Symposium on VLSI Technology*, Jun. 2008, pp. 17-19.

[72] E. Pop, "The role of electrical and thermal contact resistance for Joule breakdown of single-wall carbon nanotube," *Nanotechnology*, vol. 19, no. 29, pp. 273-274, Jun. 2008.

[73] W. Zhou, C. Rutherglen, and P. Burke, "Wafer scale synthesis of dense aligned arrays of single-walled carbon nanotubes," *Nano Research*, vol. 1, pp. 158-165, Aug. 2008.

[74] Y. Li et al. "Preferential growth of semiconducting single-walled carbon nanotubes by a plasma enhanced CVD method," *Nano Letters*, vol. 4, no. 2, pp. 317-321, Jan. 2004.

[75] R. F Smith, T. Rueckes, S. Konsek, J. W. Ward, D. K. Brock, and B. M. Segal, "Carbon nanotube based memory development and testing," in *Proceedings of the IEEE Aerospace Conference*, Mar. 2007, pp. 1-5.

[76] D. Boning and S. Nassif, "Models of process variations in device and interconnect," in *Design of High-Performance Microprocessor Circuits*, A. Chandrakasan, W. J. Bowhill, and F. Fox, Eds. New York, NY; Wiley-IEEE Press, 2000.

[77] Y. Lin, M. Hutton, and L. He, “Placement and timing for FPGAs considering variations,” in *Proceedings of the International Conference on Field Programmable Logic and Applications*, Aug. 2006, pp.1-7.

[78] S. Sivaswamy and K. Bazargan, “Variation-aware routing for FPGAs,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Array*, Feb. 2007, pp. 71-79.

[79] C. Visweswariah et al. “First-order incremental block-based statistical timing analysis,” in *Proceedings of Design Automation Conference*, Jun. 2004, pp 331-336.

[80] A. Devgan and C. Kashyap, “Block-based static timing analysis with uncertainty,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2003, pp. 607-614.

[81] J. Liou, K. Cheng, S. Kundu, and A. Krstic, “Fast statistical timing analysis by probabilistic event propagation,” in *Proceedings of Design Automation Conference*, Jun. 2001, pp. 661-666.

[82] C. Ababei, H. Mogal, and K. Bazargan, “Three-dimensional Place and Route for FPGAs,” *IEEE Transactions on Computer-Aided Design*, vol. 25, no. 6, pp. 1132-1140, Jun. 2006.

[83] M. Alexander, J. Cohoon, J. Colflesh, J. Karro, E. L. Peters, and G. Robins, “Placement and routing for three-dimensional FPGAs,” in *Proceedings of the Canadian Workshop on Field-Programmable Devices*, 1996, pp. 11-18.

[84] J. Karro and J. Cohoon, “A spiffy tool for the simultaneous placement and global routing for three-dimensional field-programmable gate arrays,” in *Proceedings of Great Lakes Symposium on VLSI*, Mar. 1999, pp. 226-227.

[85] A. Gayasen, N. Vijaykrishnan, M. Kandemir, and A. Rahman, “Designing a 3-D FPGA: Switch box architecture and thermal issues,” *IEEE Transactions on VLSI Systems*, vol. 16, no.7, pp. 882-893, Jul. 2008.

[86] Y. Lin, L. He, and M. Hutton, “Stochastic physical synthesis considering prerouting interconnect uncertainty and process variation for FPGAs,” *IEEE Transactions on VLSI Systems*, vol. 16, no.2, pp. 124-133, Feb. 2008.

[87] S. J. Koester et al. “Wafer-Level 3D Integration Technology,” *IBM Journal of Research and Development*, vol. 52, no. 6, pp. 583-597, Nov. 2008.

[88] J. U. Knickerbocker et al., “Three-dimensional silicon integration,” *IBM Journal of Research and Development*, vol. 52, no. 6, pp. 553-569, Nov. 2008.

[89] G. Lemieux and D. Lewis, “Circuit design of FPGA routing switches,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Array*, Feb. 2002, pp. 19-28.

[90] H. Chang and S. S. Sapatnekar, “Statistical timing analysis considering spatial correlations using a single PERT-like traversal,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, Nov. 2003, pp. 621-625.

[91] G. Lucas, S. Cromar, and D. Chen, “FastYield: Variation-aware, layout-driven simultaneous binding and module selection for performance yield optimization,” in *Proceedings of Asia South Pacific Design Automation Conference*, Jan. 2009, pp. 61-66.

[92] J. Xiong, V. Zolotov, and L. He, “Robust extraction of spatial correlation,” in *Proceedings of International Symposium on Physical Design*, May. 2007, pp. 2–9.

[93] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester, “Statistical estimation of leakage current considering inter- and intra-die process variation,” in *Proceedings of International Symposium on Low Power Design*, Aug. 2003, pp. 84-89.

[94] C. Chen et al. “Efficient FPGAs using nanoelectromechanical relays,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Array*, Feb. 2010, pp. 273-282.

[95] J. Luu et al. “VPR 5.0: FPGA cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling,” in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Array*, Feb. 2009, pp. 133-142.

[96] R. Parsa et al. “Composite polysilicon-platinum lateral nanoelectromechanical relays,” in *Proceedings of Hilton Head Workshop: A Solid-State Sensors, Actuators and Microsystems Workshop*, Jun. 2010, pp. 7-10.

[97] H. J. De Los Santos, G. Fischer, H. A. C. Tilmans, and J. T. M. van Beek, “RF MEMS for ubiquitous wireless connectivity: Part 1-fabrication,” *IEEE Microwave Magazine*, vol. 5, no. 4, pp. 36-49, Dec. 2004.

[98] Cavendish Kinetics Corp., (2010). Cavendish Ushers in Next Generation of MEMS and IC Integration. [Online]. Available: http://www.cavendish-kinetics.com

[99] P. Morrow et al. “Wafer-level 3D interconnects via Cu bonding,” in *Proceedings of the Advanced Metallization Conference*, Oct. 2004, pp. 125-130.

[100] P. Lindner, V. Dragoi, T. Glinsner, C. Schaefer, and R. Islam, “3D interconnect through aligned wafer level bonding,” in *Proceedings of the Electronic Components and Technology Conference*, May 2002, pp. 1439-1443.

[101] Tezzaron Semiconductor, (2009, Jul.). Tezzaron’s Patented Technologies. [Online]. Available: http://www.tezzaron.com/

[102] C. Dong, S. Chilstedt, and D. Chen, “Variation aware routing for three-dimensional FPGAs,” in *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, May 2009, pp. 298-303.