

FPCNA: A Field Programmable Carbon Nanotube Array

Chen Dong, Scott Chilstedt, and Deming Chen
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{cdong3, chilste1, dchen}@illinois.edu

ABSTRACT

Carbon nanotubes (CNTs), with their unique electronic properties, are promising materials for building nanoscale circuits. In this paper, we present a new CNT-based FPGA architecture known as FPCNA. We define novel CNT and nanoswitch based components and characterize these components considering nano-specific process variations, including the variation caused by the random mixture of metallic and semiconducting CNTs. To evaluate the architecture, we develop a variation-aware physical-design flow which can handle both Gaussian and non-Gaussian random variables using variation-aware placement and routing. When FPCNA is evaluated with this CAD flow, we see a $2.67\times$ performance gain over a baseline CMOS FPGA at the same technology node (at a 95% performance yield). In addition, FPCNA offers a $4.5\times$ footprint reduction compared to the baseline FPGA. These results demonstrate the potential of using CNTs and nanoswitches to build high performance FPGA circuits.

Categories and Subject Descriptors

B.7.1 [Integrated Circuits]: Types and Design Styles – Gate arrays, Advanced technologies; B.7.2 [Integrated Circuits]: Design Aids – Placement and routing

General Terms

Algorithms, Design, Experimentation, Performance

1. INTRODUCTION

At 22nm and below, the conventional top-down manufacturing process faces serious challenges due to physical device limitations and fundamental economic constraints. Shifting to a bottom-up approach, in which self-assembled nanoscale building blocks such as nanowires and carbon nanotubes (CNTs) are combined to create integrated functional devices, offers the potential to overcome these challenges and revolutionize electronic system fabrication.

The nature of the chemical synthesis process allows smaller feature sizes to be defined, but offers less control over individual device location. This requires a very regular structure, making it ideal for creating the repetitive structures found in Field Programmable Gate Arrays (FPGAs). In addition, the programmability of FPGAs allows reconfiguration around the large number of fabrication defects inherent in nanoscale processes. This provides a high level of fault tolerance which is critical for correct nanocircuit operation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

FPGA '09, February 22–24, 2009, Monterey, California, USA.

Copyright 2009 ACM 978-1-60558-410-2/09/02...\$5.00.

Several nanomaterial-based programmable architectures have been proposed in the literature. In [8], Goldstein and Budiu presented an island-style fabric in which clusters of nanoblocks and switch blocks are interconnected in an array structure. A PLA-based architecture called NanoPLA was presented by DeHon in [4]. This architecture builds logic from crossed sets of parallel semiconducting nanowires and uses nanowire field-effect gating to provide signal restoration and inversion. Snider et al. proposed a CMOS-like logic structure based on nanoscale FETs in [20], where nanowire crossbars were used to make arrays of both n-type and p-type semiconductors.

CMOS-Nano hybrid FPGAs have also been explored. In [7], an architecture was presented by Gayasen et al. that combines CMOS logic blocks/clusters with nanowire routing elements. On the contrary, Rad et al. presented a nanowire-cluster based FPGA in [16] where the inter-cluster routing remains in CMOS. In [22], a promising cell-based architecture called CMOL was proposed by Strukov and Likharev. CMOL uses specially doped silicon pins on the substrate surface to provide the contacts between nanowires and the CMOS layer. A generalized CMOL architecture named FPNI was proposed by Snider et al. in [19]. Unlike CMOL's inverter array architecture, the logic in FPNI is implemented with gate arrays in the CMOS layer, and nanowires are only used for routing. In reference [6], a 3D nano-FPGA architecture is proposed that distributes the components of a 2D FPGA into vertically stacked CMOS and nanomaterial layers. Researchers have also proposed using carbon nanotube based memory (i.e., NRAM [26][27]) as both block storage for FPGAs configuration data [24], and FPGA LUT memory [29].

While many of the aforementioned studies demonstrate the use of nanowire crossbars for logic and interconnect, none of them explore the possibility of using carbon-based nanoelectronics to implement FPGA logic. In addition, none of the previous designs were evaluated using a variation-aware CAD flow to predict the performance yield under the large variations in nanoscale fabrication.

In this paper, we propose a new carbon nanotube based FPGA architecture called FPCNA. We detail the building blocks of FPCNA, including the carbon nanotube lookup table which makes up its programmable logic. We also describe a high-density routing architecture using a recently proposed nanoswitch device. In our design, special considerations are made to mitigate the negative effects of nano-specific process variations. We characterize our components considering these variations, as well as circuit-level delay variations.

To evaluate the performance of our architecture, we adopt a typical FPGA design flow and develop variation-aware placement and routing algorithms. These algorithms are enhanced from the popular physical design tool VPR [2] and use statistical timing analysis (SSTA) to improve the performance yield. We perform

SSTA with both normal and non-Gaussian variation models. Our 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.

This paper is organized as follows: In Section 2 we introduce the molecular devices used in FPCNA. In Section 3 we present the FPCNA architecture, and describe how these devices are used to build components. Section 4 discusses the fabrication of our CNT-based lookup table. Circuit-level characterization results are presented in Section 5, which provide meaningful guidance for our CAD flow presented in Section 6. In Section 7 we present experimental results showing the advantages of FPCNA over a conventional CMOS FPGA. Section 8 concludes this paper.

2. MOLECULAR DEVICES

2.1 Carbon Nanotubes

Single-walled carbon nanotubes (SWCNTs) are molecular devices with a diameter of roughly 1nm. They are composed of hexagonal carbon rings which form a seamless cylinder, and depending on the chirality of the lattice can be either metallic or semiconducting [15], as shown in Figure 1 [30].

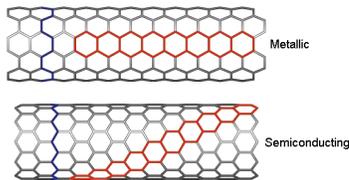


Figure 1. Two types of CNTs

Metallic SWCNTs have high electron mobility and robustness, making them attractive for uses in nanoscale interconnect and nano-electromechanical systems (NEMS). Semiconducting SWCNTs, on the other hand, have ideal characteristics for use in field-effect transistors [11].

In the past, it was difficult to mass produce CNT-based circuits because of the inability to accurately position the nanotubes. However, recent research has demonstrated the fabrication of dense, perfectly aligned arrays of linear SWCNTs [11], and wafer scale CNT-based logic devices [28].

2.2 CNT Field Effect Transistors

The structure and operation of a CNT-based field effect transistor (CNFET) is analogous to that of a silicon-based device. A semiconducting CNT forms the conducting channel between the source and drain contacts, and is controlled by a gate electrode. In the past decade, several groups have reported that CNFETs are a promising post-silicon technology [12][13]. Based on intrinsic CV/I gate delay, CNFET devices can be up to 13× and 6× faster than pMOS and nMOS devices of the same gate length, when local interconnect capacitances and CNT imperfections are not considered [5].

2.3 NRAM

NRAM is a nonvolatile NEMS memory device formed by suspending a metallic CNT over a trench which contains a base electrode. The off state is characterized by the CNT lying flat across the trench where elastic energy keeps the tubes in place. The on state occurs when the CNT is bent into the trench and

makes contact with the base electrode. In the on state, the van der Waals force between the CNT and the trench floor creates a strong molecular attraction, overpowering the elastic energy. 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, bi-stable memory device with well-defined off and on states [26][27].

2.4 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 [14][23]. Since individual SWCNTs have a large contact resistance, a rope or bundle of SWCNTs is used to transfer current in parallel. SWCNT bundle interconnect can outperform copper interconnect for propagation delay, especially for intermediate and long interconnects [21].

2.5 Solid-Electrolyte Nanoswitches

Solid-Electrolyte switches are a new type of nano-scale switch developed by [10]. A Solid-Electrolyte switch is created by sandwiching a layer of Cu_2S between two metals, Ti and Cu (Figure 2(a)).

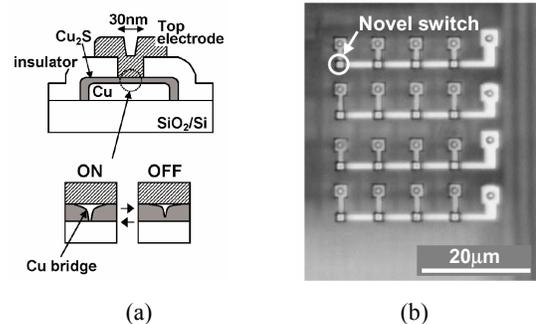


Figure 2. (a) Solid-Electrolyte switch (b) SEM image of a 4x4 crossbar switch array

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 ohms 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. The contact holes are formed by electron beam lithography and the upper electrode is obtained by the lift-off process. To explore the potential of Solid-Electrolyte switches, prototypes with a contact diameter of 30nm were created in [10]. In addition, a 4×4 crossbar switch array (Figure 2(b)) was fabricated and tested [10].

3. FPCNA ARCHITECTURE

In this section we describe the FPCNA architecture in detail. We begin with a review of the generic island-based FPGA architecture, and then introduce the new molecular device based LUT design, local and global routing, and high level architecture.

3.1 Island-Style FPGA Architecture

We adopt a conventional island-based FPGA architecture as the basis for FPCNA. The basic structural unit is a tile consisting of one switch block (SB), two connection blocks (CB), and one configurable logic block (CLB) (Figure 3). The global routing structure is composed of segmented interconnect channels connected by programmable switch blocks. CLBs are connected to the routing channels through connection blocks. Each CLB contains several basic logic elements (BLEs) and the local routing used to connect them, as shown in Figure 4 from [2]. The parameter I represents the number of inputs to a CLB, F_c defines the number of routing tracks a CLB input can connect to, and N represents the number of BLEs that a CLB contains. Each BLE consists of one K -input lookup table (K -LUT) and one flip-flop.

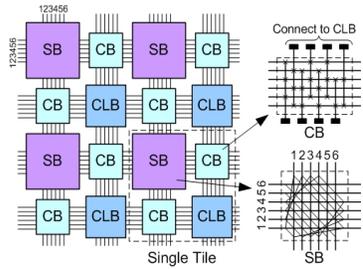


Figure 3. Island-Style FPGA Architecture

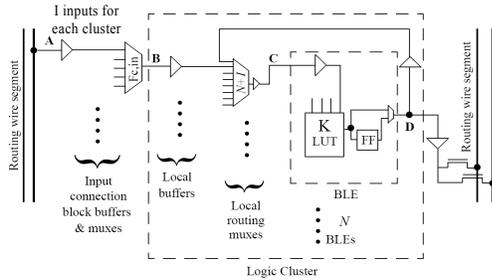


Figure 4. Schematic of a CLB (Logic Cluster)

3.2 FPCNA LUT Design

A K -LUT is the basic unit of programmable logic in an FPGA. For FPCNA, we propose a novel LUT design based entirely on carbon nanotube devices (Figure 5). This design uses parallel ribbons of SWCNTs in conjunction with metal gates and metal electrodes. CNFET devices are formed at the crossing points of the CNT ribbons and the metal gates, creating a K -input 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. Gray address decoding is used to reduce the number of gate-to-metal1 vias.

One of the key innovations of this LUT design is that it builds the decoding and memory on the same continuous CNT ribbons. This holistic structure allows for high logic density and simplifies the manufacturing process. For comparison, [29] 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 serious fabrication issues because it requires the alignment and interfacing of individual nanotubes in two dimensions.

By using CNT ribbons, we have the advantage that each device will contain multiple tubes. This adds fault tolerance from the high defect rates of nanotube fabrication, and increases the chance that the CNT memory will contain functioning metallic tubes. Thus, the design is more reliable than in [29] where a device will fail if either of the two nanotubes is defective.

We discuss the fabrication of this device in Section 4 and characterize its delay in Section 5.

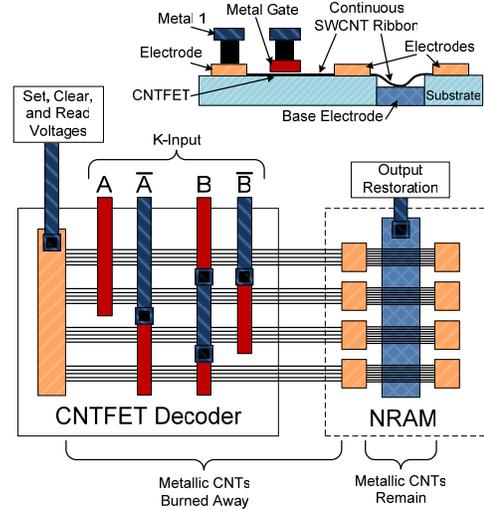


Figure 5. CNT-based LUT

3.3 FPCNA Local Routing

Another key architectural component is the structure of the routing. In conventional CMOS FPGA designs, the routing is often multiplexor-based. While we could adopt the same approach and use NRAM to store multiplexor configuration bits, we believe a greater logic density can be achieved by using solid-electrolyte switch crossbars.

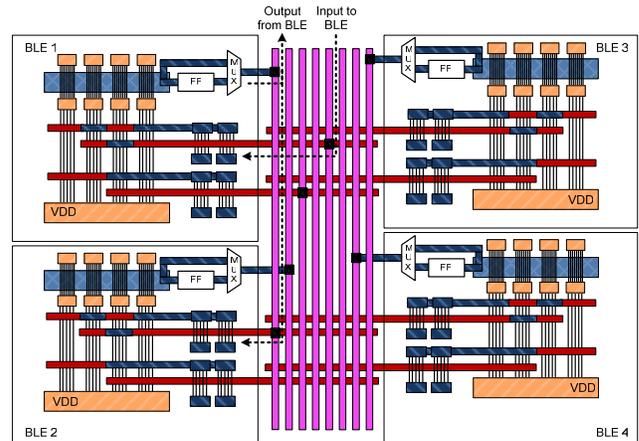


Figure 6. CNT-based CLB with nanoswitch local routing

Figure 6 shows a simplified local routing design to illustrate this technique. This CLB structure contains four BLEs made from CNT-based LUTs. The local routing is created with Solid-electrolyte switches 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 6, 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 those solid-electrolyte switches are turned on. By turning on multiple 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 output from the CLB to connect to the adjacent connection blocks.

For simplicity, only 2-input LUTs are shown in Figures 5 and 6. When scaled to K inputs, LUTs will contain 2^K CNT ribbons. The BLE flip-flops and muxes could also be implemented using CNT-ribbons, but for this study we assume CMOS size and delay. Area calculations assume a more compact layout than shown in Figure 6.

3.4 FPCNA Global Routing

The global routing structure consists of two-dimensional segmented interconnects connected through programmable switch blocks (SBs) and connection blocks (CBs). The connection blocks connect signals between the CLBs and wires in the global routing channels (Figure 3). We use CNT-bundle interconnects for global routing because they have been shown to be superior to copper in terms of current density and delay [14].

In a traditional CMOS-based FPGA, the SBs and CBs take up the majority of the overall area [1]. 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, while the CLBs occupy the remaining 42.6% [1]. To reduce the size of the global routing in FPCNA, we replace the traditional CB with a solid-electrolyte switch crossbar, and propose a new nanoswitch-based SB design.

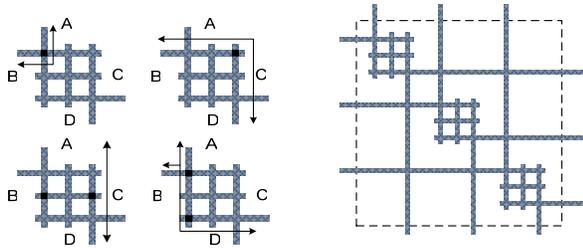


Figure 7. Nanoswitch-based switch block design. (a) switching scenarios, (b) switch block pattern

The new SB design is shown in Figure 7. Instead of using six SRAM-controlled pass transistors for each switch point as in conventional CMOS [2], we use six perpendicular wire segments. By programming nanoswitches at the crosspoints of the resultant array, a signal coming from one side of the block can be routed to any or all of the other three sides. This significantly reduces the SB area because an SRAM cell normally requires 72T (where T is the minimum size transistor area), and the nanoswitch design can perform the same function in approximately 9T.

Figure 7(a) demonstrates how routing connections can be made. The figure uses arrows to show connection paths and black dots

for switches in the ON state. The lower right scenario shows a multipath connection between sides A, B, and C. Other patterns can be constructed by tuning on the appropriate nanoswitches. Figure 7(b) illustrates an example 3x3 universal-style switch block using wire-based switch points.

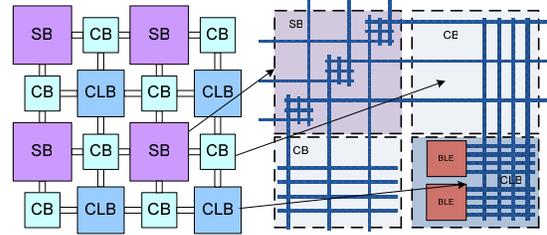


Figure 8. High-level layout of FPCNA

3.5 FPCNA Architecture

The conceptual high level layout of FPCNA is shown in Figure 8, demonstrating how the new blocks fit into the traditional tile-based structure. The CLBs connect to the routing channels through the CBs, and signals between CBs connect through the SBs. This design leads to a significant area reduction, as shown in Section 5.

4. NANOTUBE LUT FABRICATION

Recent progress in the fabrication of CNTs has enabled us to explore the use of CNT-based structures in FPCNA’s LUT design. To demonstrate the feasibility of this design at the 32nm technology node, we address some of the fabrication issues involved.

The first step in manufacturing the LUT is the creation of the nanotubes themselves. Since all of the CNT ribbons are aligned in the same direction, an array-based CNT growth process can be used. In [11], researchers report a technique for fabricating dense, perfectly aligned arrays of CNTs. Using photolithographically-defined catalytic seeds, nanotubes are grown on quartz wafers using chemical vapor deposition, and can achieve up to 99.9% alignment. The aligned nanotubes can then be transferred to a silicon wafer using the stamping process developed in [35]. These techniques create nanotubes that are suitable for the transistors and NEMS devices used in our LUT. In addition, it is possible to improve nanotube density on the silicon wafer by performing multiple transfers. In this study, we assume that the distance between ribbons is 64nm wide and that this resolution can be achieved by the target photolithographic process.

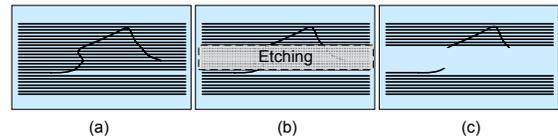


Figure 9. CNT ribbon etching

The next step is to define ribbons from the continuous nanotube array. Figure 9(a) illustrates an array containing a misaligned tube from the transfer process. As shown in Figure 9(b), ribbons are created by an etching process such as [9]. This has the added advantage of making the ribbons misalignment immune, because any nanotubes crossing the border of a ribbon are removed during etching (Fig 9(c)), which enhances the reliability of the LUT.

The third major step is to remove the metallic nanotubes inside the decoder region. Since metallic CNTs act as a short between the source and drain, they need to be removed to create CNFET logic with desirable on/off current ratios. Electrical burning [36] is an effective technique to selectively burn the metallic nanotubes. The mechanism for burning is applying a voltage large enough to reach the breakdown temperature of the conducting metallic nanotubes and cause irreversible oxidization. Burning is only done in the decoder region because the metallic tubes are needed for NRAM operation.

After these CNT ribbons are defined and processed, the gate and source/drain formation is similar to a regular CMOS process. Based on the techniques outlined above and some existing CNT fabrication work [11][27], we believe the proposed nanotube-based LUT design is implementable.

5. CIRCUIT CHARACTERIZATION

5.1 CNFET and CNT-based LUT Variation

As mentioned in Section 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 perfectly aligned nanotubes with exactly controlled chirality using known fabrication techniques. HiPco synthesis techniques yield around $50\% \pm 10\%$ metallic CNTs [12]. 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 [5].

It is well known that the delay of a MOSFET follows a Gaussian distribution, and highly likely that a CNFET will as well. To quantify the effects of the aforementioned variations, we perform a Monte Carlo simulation of CNFET devices with 2,000 runs. We consider the sources of variation listed in Table 1, targeting the number of CNTs per channel as a normally distributed 8 ± 3 . The diameter range, doping level range, and CNFET model are suggested in [5]. The result of the simulation shows that the delay distribution of a CNFET device under these variations fits the Gaussian distribution (Figure 10)

Table 1: Sources of variation

Parameter	Mean	Variation (3σ)
CNTs per channel	8	± 3
CNT diameter	1.5nm	$\pm 0.3\text{nm}$
Doping level	0.6ev	$\pm 0.03\text{ev}$

We also evaluate the performance of our CNT-based LUT design. The LUT decoder consists of multiple stages of p-type CNFETs, simulated under the variations mentioned in Table 1. The contact resistance between an electrode and a single nanotube is assumed to be $20\text{K}\Omega$ [14]. Ribbon contact resistance is considered to be inversely proportional to the number of semiconducting nanotubes in the ribbon. Similarly, the contact resistance between a bending nanotube and the base electrode is reported as $10\text{K}\Omega$, and the NRAM contact resistance is inversely proportional to the number of metallic nanotubes. Figure 11 shows the LUT delay distribution generated by Monte Carlo simulation in HSPICE.

Unlike traditional CMOS circuits, the delay of our nanotube-based LUT has a distribution similar to log-normal.

5.2 Crossbar Characterization

As explained in Section 3, the routing in FPCNA is implemented using crossbars. We capture the delay and variation of these crossbars using HSPICE. Both CNT bundle interconnect and metal interconnect are assumed to be 32nm in width, with an aspect ratio of 2. We set the dielectric constant of the insulation material around the crossbar at 2.5, and derive a unit resistance of $16.526 \Omega/\mu\text{m}$ and capacitance of $276.214 \text{ aF}/\mu\text{m}$ for metal interconnect, and a unit resistance of $10.742 \Omega/\mu\text{m}$ and capacitance of $359.078 \text{ aF}/\mu\text{m}$ for the carbon nanotube bundles. Both types of interconnect are evaluated for 10% geometrical variation of wire width, wire thickness, and spacing according to [31]. 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 100Ω ON resistance [10] and 10% variation to capture via contact resistance.

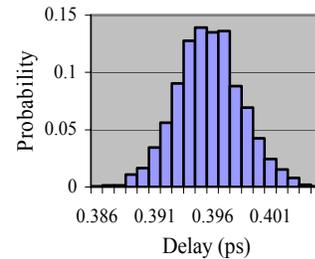


Figure 10. Delay distribution of a CNFET under process variation

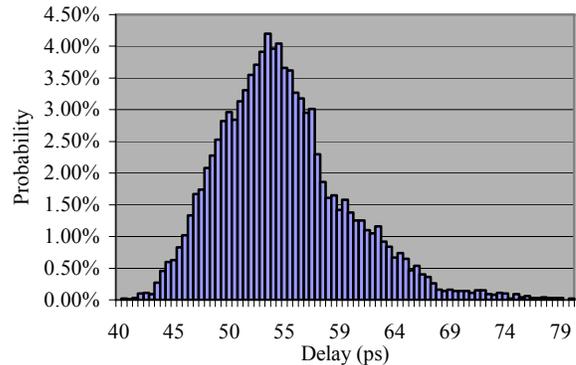


Figure 11. CNT-based LUT delay considering variation

5.3 Basic Timing Block of FPCNA

To support the CAD flow, various circuit models have to be built to capture the specific characteristics of the FPCNA architecture. In the architecture specification file of VPR, we specify the delay values for various combinational circuit paths to enable accurate timing analysis. For example, in Figure 4, there are paths $A \rightarrow B$, $B \rightarrow C$, and $D \rightarrow C$, etc. In FPCNA, these paths would contain buffers, metal wires, and solid-electrolyte switches and are also susceptible to process variation.

We extract these different paths for FPCNA and perform 1,000 Monte Carlo simulations to compute their delay and variation. By varying the CNFET parameters and the contact resistance between CNT and metal, the delay (μ) and delay variation (σ) are calculated as shown in Table 2. The CMOS counterparts are calculated using Monte Carlo simulation by assuming 12% channel width variance, 4% gate dielectric thickness variance, and 5% doping variance (quoted from [38] for 32nm CMOS). Figure 12 illustrates the delay distribution of wire track to subblock input connections ($A \rightarrow B$) and subblock output to subblock input connections ($D \rightarrow C$). Based on these results, we observe that the timing blocks follow a normal-like distribution.

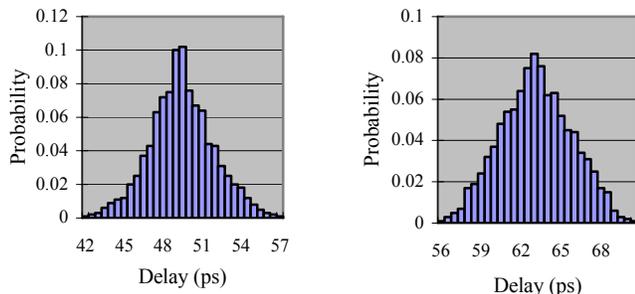


Figure 12. Delay distribution of wire track to sub-block input (left) and sub-block output to sub-block input (right)

Table 2: Delay Comparison of Baseline and FPCNA

Paths	CMOS-Based		FPCNA	
	μ (ps)	σ (ps)	μ (ps)	σ (ps)
$A \rightarrow B$	141.66	7.13	49.23	2.50
$B \rightarrow C$	107.59	5.37	36.98	2.13
$D \rightarrow C$	107.59	5.37	63.10	3.20
$D \rightarrow \text{Out}$	28.48	1.22	36.83	2.14

6. CAD FLOW

In the previous sections, we show that both deep sub-micron CMOS and nano devices are susceptible to variation. Traditional static timing analysis assumes 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, we use the timing-driven, variation-aware CAD flow shown in Figure 13. Each benchmark circuit goes through technology independent logic optimization using SIS [18] and is technology-mapped to 4-LUTs using DAOMap [3]. The mapped netlist then feeds into T-VPACK and VPR [2], which perform timing-driven packing (i.e., clustering LUTs into the CLBs), placement, and routing. To take variation into consideration, we enhance the VPR tool [2] to make it variation aware.

Existing works have shown that statistical optimization techniques are useful during the physical design stage. Variation

aware placement is implemented in [37] and variation aware routing is developed in [25]. Based on the ideas presented in these works, we implement a complete variation aware physical design flow. In this holistic solution, the placer calls the variation aware router to generate delay estimates for its timing cost calculations.

From the Monte Carlo simulation results in section V, we observe that the CNT-based LUT delay follows a non-Gaussian distribution. Reference [14] also reports a non-Gaussian distribution for CNT bundle interconnect. However, all of the existing CAD work targeting CMOS assumes normally distributed random variables [25][37][34]. The Gaussian-based SSTA algorithms that these works use to evaluate CMOS are not suitable for modeling the non-normal variables of our molecular-based architecture. Therefore, in this work we develop a new statistical timing analyzer that can handle an arbitrary distribution, based on discretization techniques adapted from [32][33].

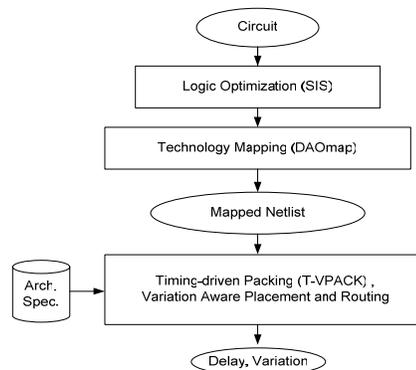


Figure 13. FPCNA Evaluation Flow

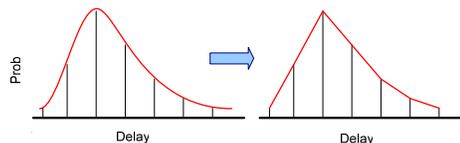


Figure 14. Discretization Process of Log-Normal Probability Density Function

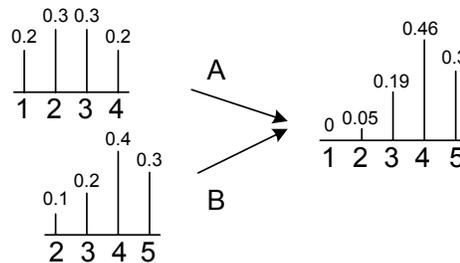


Figure 15. The discretized MAX operation

One such technique is the probabilistic event propagation developed in [32], in which discretized random variables of cell delays are used for timing analysis. As illustrated in Figure 14, a non-Gaussian probability density function can be represented as a set of delay-probability pairs which contain the time t and the

probability a signal will arrive at time t . In [33], ADD and MIN operations are developed for propagating multiple event groups. We use these operations, and additionally define a MAX operation for our statistical timing analyzer. In Figure 15, we illustrate how the discretized MAX operation is performed using an example point.

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 , we define the probability that both inputs arrive 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

Take $t=1$ for example. The earliest arrive time for B is 2, so $P(1)=0$. The probability that both A and B arrive at $t=2$ is $0.1 \times 0.3 = 0.03$. The probability that A arrives at $t=2$, and B arrived before $t=2$ is $0.3 \times 0 = 0$. The probability that B arrives at $t=2$, and A arrived before $t=2$ is $0.2 \times 0.1 = 0.02$. Therefore, $P(2)$ sums up to 0.05. The remaining points are processed in a similar fashion. The accuracy of this technique is dependent on the number of points used for piecewise linear approximation. It is shown in [32] that 7 points are sufficient to obtain an accuracy of less than 1% error compared to Monte Carlo. Therefore, we use 7-point sampling throughout our discretized SSTA.

Figure 16 shows the pseudo-code of our 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, we express the arrival time of pin j in net i 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:

$$t_s = t_r - t_a \quad \sigma_s = \sqrt{\sigma_a^2 + \sigma_r^2}$$

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

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

We modify the original VPR cost function in this way so when two slacks have a similar mean but different variations, the $slack(i, j) - 3\sigma_s(i, j)$ 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 17, where the

distribution with slack variation σ_1 will be assigned a higher criticality than the distribution with slack variation σ_2 , even they have the same mean. This cost function also considers the critical path variation in 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:

$$Disc_Crit(i, j) = 1 - \frac{E[disc_slack(i, j)]}{E[disc_t_{crit}]}$$

After each routing iteration, SSTA is executed by traversing the updated timing graph to calculate the new slack and critical path delay.

```

Starting: 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
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 16. Pseudo-code of modified VPR router

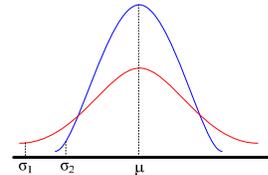


Figure 17. Criticality estimation

The variation aware placement also uses these criticality functions to calculate the timing cost of each move during simulated annealing. In the placer's cost function, the critically value is raised by the exponent β . We determined the optimal value of β to be 6 for our design. This differs from the original VPR method of incrementing β from 1 to 8, and from [37] where a β value of 0.3 was used. As in [37], we calculate the variation during the delta array creation, and store these pre-calculated values in the delta arrays. The main difference is that we use a variation aware router to generate the delay and store sets of discretized delay-probability points for each delay value in addition to the mean and variance.

This CAD flow is flexible. We can choose various parameters for LUT size K , CLB size N , routing architectures, and interconnect buffer sizes, etc. In our study, we set $K = 4$, $N = 10$ and the routing channel width to 110. A fixed routing channel width is often used for CAD evaluation since modern FPGAs usually provide sufficient routing resources and have a fixed channel width. The interconnect in an FPGAs is segmented and driven by

buffers. It is known that a mixture of different length interconnects can provide improved performance [2]. In our experiments, we use an equal mixture of length-4 and length-8 wire segments (wires crossing either four CLBs or eight CLBs) to route the signals, which is reported as one of the optimal combinations in [2]. We set $F_c = 0.5$, another typical value which connects the CLB input to half of the routing tracks in the channel, and set the number of CLB inputs to 22.

7. EXPERIMENTAL RESULTS

7.1 FPCNA 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, we use the architecture parameters defined in Section 6, and assume a transistor feature size of 32nm for both CNT and CMOS-based transistors. In our CNT-based LUT (Fig. 5), the nanotube ribbons are 64nm wide and spaced 64nm apart. The distance between the contact electrode and a gate, and the distance between gates is assumed to be 64nm. The nanotube memory, NRAM, offers a much smaller area than an SRAM cell. The NRAM region in Fig. 5 is assumed to be 256nm in width which has been fabricated in [27]. Crossbars in FPCNA have a wire pitch of 64nm. By replacing the MUX-based routing with these crossbars, as detailed in Section 3, a large area reduction can be achieved. Since the routing path in FPCNA is controlled by non-volatile solid-electrolyte switches, SRAM cells used in the baseline CMOS FPGA can be eliminated. We estimate the footprint of the 32nm CMOS FPGA tile as 34623T. Multiplying this by the $0.0451 \mu\text{m}^2$ area for a 32nm transistor gives a tile area of $1561.5 \mu\text{m}^2$. These calculations show that FPCNA achieves a roughly 4.5X area reduction over CMOS (Table 3).

Table 3: Area Reduction

	CMOS FPGA	FPCNA	Reduction
CLB Area	$665.2 \mu\text{m}^2$	$100.3 \mu\text{m}^2$	6.63×
CB Area	$337.7 \mu\text{m}^2$	$82.5 \mu\text{m}^2$	4.1×
SB Area	$558.6 \mu\text{m}^2$	$162.2 \mu\text{m}^2$	3.4×
Tile Area	$1561.5 \mu\text{m}^2$	$345.0 \mu\text{m}^2$	4.53×

7.2 FPCNA Performance Gain

In this section, we evaluate the experimental CAD flow presented in Section 6, quantifying the overall performance improvement of FPCNA from the baseline CMOS counterpart. We run our simulations using 20 MCNC benchmarks.

When considering variations, the performance evaluation becomes complicated. Critical path delay can no longer serve as the absolute measure. Due to variation, near critical path delays may also cause a time violation. This is illustrated in Fig. 18 (a), where the target clock period is set according to critical path delay. In this case both PO₂ and PO₃ can fail due to variation. Because of this phenomenon, it is necessary to consider the statistical delay of every path. We express the performance yield as a delay-probability pair (t, p), so that by setting the clock period t , we can evaluate the system yield p . This method allows us to directly compare the performance of the statistical information generated by our study.

Fig. 19 demonstrates how we calculate performance yield for both Gaussian and non-Gaussian distributions. By selecting a target clock period T_c , the yield of each of the POs is computed. For a Gaussian distribution, the yield is derived by applying an inverse cumulative distribution function (CDF) on 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 (Fig. 18 (b)). The system yield is simply a multiplication of all path yields. If the system yield is not satisfied, we increase the target clock period and repeat the process until the target yield is satisfied. The final clock period which guarantees the target yield is reported.

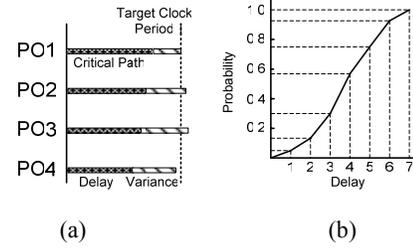


Figure 18. (a) Yield vs. clock period (b) Piecewise linear CDF in discretized timing analysis

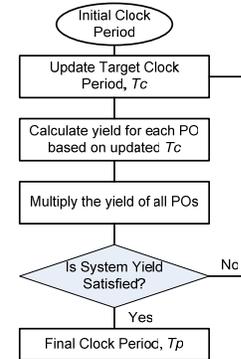


Figure 19. Flowchart of performance yield estimation

To compare to a conservative deterministic solution, we evaluate the worst case performance using the original VPR tool [2] with a guard-banded delay of mean + 3σ for each element in VPR's architecture file.

Table 4 shows the achievable clock period for the worst case CMOS results and the 90%, 95%, and 99% performance yield for both CMOS and FPCNA reported by our CAD flow. Columns 1 to 4 show that by satisfying 90%, 95%, and 99% yield, the variation aware CAD flow reduces the critical path delay by 14.98%, 14.33%, and 13.07% compared to the worst case (delay averages are calculated using the geometric mean). Column 5 shows the performance gain at 95% yield of the variation aware CAD flow over the worst case design (in terms of maximum clock frequency, F_{max}).

Table 5 identifies the performance gain of adding variation aware placement to variation aware routing. At 99% yield, variation aware placement and routing produces a 2.5% and 3.6% performance gain over deterministic placement and variation

Table 4: CMOS Performance Yield and Comparison

	CMOS	CMOS with Variation Aware CAD Flow				FPCNA with Variation Aware CAD Flow			
	Worst Case 3 σ Guard Banded (ns)	90% Performance Yield (ns)	95% Performance Yield (ns)	99% Performance Yield (ns)	Performance Gain over Worst Case	90% Performance Yield (ns)	95% Performance Yield (ns)	99% Performance Yield (ns)	Performance Gain over CMOS
alu4	9.262	7.269	7.338	7.469	1.26 \times	2.555	2.590	2.711	2.83 \times
apex2	10.51	8.368	8.444	8.587	1.25 \times	3.271	3.299	3.534	2.56 \times
apex4	9.796	7.538	7.602	7.726	1.29 \times	3.452	3.492	3.529	2.18 \times
bigkey	4.580	4.295	4.336	4.416	1.06 \times	1.571	1.585	1.627	2.74 \times
clma	20.55	18.88	18.98	19.18	1.08 \times	6.738	6.817	6.880	2.78 \times
des	8.900	8.778	8.853	8.994	1.01 \times	2.923	2.959	2.987	2.99 \times
diffeq	7.241	6.302	6.351	6.448	1.14 \times	2.891	2.919	3.195	2.18 \times
dsip	4.790	4.805	4.856	4.954	0.99 \times	1.617	1.626	1.694	2.99 \times
elliptic	14.87	11.20	11.26	11.39	1.32 \times	3.371	3.434	3.563	3.28 \times
ex1010	16.39	12.90	12.99	13.15	1.26 \times	5.233	5.287	5.331	2.46 \times
ex5p	9.885	8.612	8.693	8.847	1.14 \times	3.988	4.033	4.069	2.16 \times
frisc	16.11	14.91	14.99	15.15	1.07 \times	4.270	4.314	4.440	3.48 \times
misex3	8.284	6.488	6.543	6.649	1.27 \times	3.146	3.197	3.404	2.05 \times
pdc	17.25	16.03	16.13	16.32	1.07 \times	5.417	5.551	5.764	2.91 \times
s298	15.14	14.02	14.10	14.25	1.07 \times	4.177	4.333	4.623	3.25 \times
s38417	10.97	10.56	10.62	10.74	1.03 \times	3.080	3.122	3.226	3.40 \times
s38584.1	8.456	6.964	7.024	7.140	1.20 \times	2.477	2.481	2.484	2.83 \times
seq	10.78	7.792	7.859	7.987	1.37 \times	3.379	3.420	3.480	2.30 \times
spla	15.20	11.96	12.04	12.20	1.26 \times	4.731	4.758	4.888	2.53 \times
tseng	8.851	6.648	6.700	6.804	1.32 \times	3.044	3.062	3.262	2.19 \times
Average	10.59	9.001	9.070	9.203	1.17 \times	3.357	3.399	3.518	2.67 \times

aware routing for CMOS and FPCNA respectively. Together with the comparison data to the worst-case design shown in Table 4, it is clear that our variation-aware physical design flow is effective for obtaining shorter clock period given the same performance yield target or obtaining higher performance yield given the same clock period target.

In the last column of Table 4, we compare the performance of FPCNA with the performance of the CMOS design using the same variation aware CAD flow. At a 95% yield, the gain of FPCNA is an average of 2.67 \times over the CMOS counterpart. This significant improvement in performance is achieved by the synergistic combination of CNT logic, CNT bundle interconnects, and routing crossbar design in FPCNA.

8. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel FPGA architecture called FPCNA. We designed CNT-based and nanoswitch-based building blocks such as the CNT-based LUT, and characterized them under nano-specific process variations such as CNT diameter and CNT doping level. We also addressed some issues in CNT device fabrication. 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 4.5 \times footprint reduction and a 2.67 \times Fmax gain (targeting a 95% performance yield) compared to the baseline FPGA in the same technology node. This clearly demonstrates potential for using CNTs and nanoswitches to build the next-generation FPGA circuits. In the future, we plan to consider the effects of correlation on nano-component variation, and continue to characterize and experiment with CNT-based designs. In addition, we are exploring collaboration opportunities

with scientists in material science and nanotechnology to fabricate a prototype of the CNT-based LUT.

Table 5. Performance gain achieved by making placement variation aware

	CMOS		FPCNA	
	Performance Gain at 95% Yield	Performance Gain at 99% Yield	Performance Gain at 95% Yield	Performance Gain at 99% Yield
alu4	1.074 \times	1.077 \times	1.297 \times	1.253 \times
apex2	0.998 \times	1.005 \times	0.954 \times	0.962 \times
apex4	1.125 \times	1.130 \times	0.954 \times	0.957 \times
bigkey	0.936 \times	0.941 \times	1.105 \times	1.104 \times
clma	0.990 \times	0.988 \times	0.893 \times	0.970 \times
des	0.898 \times	0.902 \times	1.000 \times	1.015 \times
diffeq	1.131 \times	1.139 \times	1.217 \times	1.183 \times
dsip	0.866 \times	0.873 \times	0.986 \times	0.957 \times
elliptic	1.069 \times	1.071 \times	0.980 \times	0.981 \times
ex1010	1.128 \times	1.133 \times	1.052 \times	1.057 \times
ex5p	0.982 \times	0.986 \times	0.974 \times	0.985 \times
frisc	1.016 \times	1.020 \times	1.002 \times	1.007 \times
misex3	1.074 \times	1.083 \times	1.089 \times	1.059 \times
pdc	1.032 \times	1.037 \times	0.982 \times	1.019 \times
s298	0.910 \times	0.914 \times	1.073 \times	1.065 \times
s38417	0.893 \times	0.896 \times	1.031 \times	1.072 \times
s38584.1	1.017 \times	1.023 \times	1.036 \times	1.103 \times
seq	1.116 \times	1.121 \times	1.156 \times	1.176 \times
spla	1.020 \times	1.025 \times	0.823 \times	0.861 \times
tseng	1.124 \times	1.127 \times	0.973 \times	0.939 \times
Average	1.020 \times	1.025 \times	1.029 \times	1.036 \times

9. ACKNOWLEDGEMENTS

This work is partially supported by NSF Career Award CCF 07-46608, NSF grant CCF 07-02501, and a gift grant from Altera Corporation. We also appreciate the helpful discussions with Prof. John Rogers of the University of Illinois at Urbana Champaign and Prof. Subhasish Mitra of Stanford University.

10. REFERENCES

- [1] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," *IEEE Trans. on VLSI*, Vol 12, No. 3, pp. 288-298, March 2004.
- [2] V. Betz, J. Rose and A. Marquardt, "Architecture and CAD for Deep-Submicron FPGAs," *Kluwer Academic Publishers*, February 1999.
- [3] D. Chen and J. Cong, "DAOmap: A Depth-Optimal Area Optimization Mapping Algorithm for FPGA Designs," *ICCAD*, Nov. 2004.
- [4] A. DeHon, "Nanowire-based programmable architectures," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 1, no. 2, pp. 109-162, 2005.
- [5] J. Deng, et al., "Carbon Nanotube Transistor Circuits: Circuit-Level Performance Benchmarking and Design Options for Living with Imperfections," *International Solid-State Circuits Conference*, 2007.
- [6] C. Dong, D. Chen, S. Haruehanroengra, and W. Wang, "3-D nFPGA: A Reconfigurable Architecture for 3-D CMOS/Nanomaterial Hybrid Digital Circuits," *IEEE Transactions on Circuits and Systems I*, Vol. 54, Issue 11, pp. 2489-2501, Nov. 2007.
- [7] A. Gayasen, N. Vijaykrishana, M. J. Irwin, "Exploring Technology Alternatives for Nano-Scale FPGA Interconnects", *DAC*, 2005.
- [8] S. C. Goldstein and M. Budiu, "NanoFabric: Spatial Computing using Molecular Electronics," *Int. Symp. on Computer Architecture*, 2001.
- [9] A. Hassanien, et al., "Selective Etching of Metallic Single-Wall Carbon Nanotubes with Hydrogen Plasma," *Nanotechnology*, Vol. 16, pp. 278 – 281, 2005.
- [10] 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.
- [11] S. J. Kang, et al., "High-performance electronics using dense, perfectly aligned arrays of single-walled carbon nanotubes," *Nature Nanotechnology*, Vol. 2, Issue 4, pp. 230-236, 2007.
- [12] Y. Li, et al., "Preferential Growth of Semiconducting Single-Walled Carbon Nanotubes by a Plasma Enhanced CVD Method," *Nano Letters*, vol. 4, pp. 317, 2004.
- [13] X. Liu, S. Han, and C. Zhou, "Novel Nanotube-on-Insulator (NOI) Approach toward Single-Walled Carbon Nanotube Devices," *Nano Lett. (Letter)*, 6(1), pp 34-39, 2006.
- [14] 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, 2006.
- [15] P. McEuen, M. Fuhrer, and H. Park, "Single-Walled Carbon Nanotube Electronics," *Tran. on Nanotechnology*, Vol. 1, No. 1, Mar. 2002.
- [16] R. M. P. Rad and M. Tehranipoor, "A New Hybrid FPGA with Nanoscale Clusters and CMOS Routing," *DAC 2006*.
- [17] A. Raychowdhury, S. Mukhopadhyay, and K. Roy, "A Circuit-Compatible Model of Ballistic Carbon Nanotube Field-Effect Transistors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, , vol. 23, pp. 1411-1420, 2004
- [18] E. M. Sentovich et. al. "SIS: A System for Sequential Circuit Synthesis," Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, 1992.
- [19] G. Snider and S. Williams, "Nano/CMOS architecture using a field-programmable nanowire interconnect," *Nanotechnology*, vol. 18, 2007.
- [20] G. Snider, P. Kuekes, and R. S. Williams, "CMOS-like logic in defective nanoscale crossbars," *Nanotechnology*, vol. 15, 2004.
- [21] N. Srivastava and K. Banerjee, "Performance Analysis of Carbon Nanotube Interconnects for VLSI Applications," *ICCAD*, pp. 383-390, 2005.
- [22] D. B. Strukov and K. K. Likharev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," *Nanotechnology*, vol. 16, no. 888-900, 2005.
- [23] B. Q. Wei, R. Vajtai, and P. M. Ajayan, "Reliability and Current Carrying Capacity of Carbon Nanotubes," *Applied Physics Letter*, vol. 79, no. 8, pp. 1172-1174, 2001.
- [24] W. Zhang, N. Jha, and L. Shang, "NATURE: A Hybrid Nanotube/CMOS Dynamically Reconfigurable Architecture," *DAC*, 2006.
- [25] S. Sivaswamy and K. Bazargan, "Variation-aware routing for FPGAs", *Intl Symp. on Field programmable gate arrays*, 2007.
- [26] NRAM™, Nantero, <http://www.nantero.com/tech.html>.
- [27] J.W. Ward, M. Meinhold, B.M. Segal, J. Berg, R. Sen, R. Sivarajan, D.K. Brock, T. Rueckes, "A nonvolatile nanoelectromechanical memory element utilizing a fabric of carbon nanotubes," *Non-Volatile Memory Technology Symposium*, 2004 , vol., no., pp. 34-38, 15-17 Nov. 2004
- [28] 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," 2008 Symp. VLSI Technology, 2008.
- [29] Y. Zhou, S. Thekkel, and S. Bhunia, "Low power FPGA design using hybrid CMOS-NEMS approach," *International Symposium on Low Power Electronics and Design*, August, 2007.
- [30] Carbon Nanotubes <http://phycomp.technion.ac.il/~talimu/structure.html>
- [31] D. Boning, S. Nassif, "Models of Process Variations in Device and Interconnect," *Design of High-Performance Microprocessor Circuits*, Wiley-IEEE Press, ISBN: 978-0-7803-6001-3, 2000.
- [32] A. Devgan, C. Kashyap, "Block-based static timing analysis with uncertainty," *ICCAD 2003*, pp 607-614.
- [33] J. Liou, K. Cheng, S. Kundu, and A. Krstic, "Fast statistical timing analysis by probabilistic event propagation," *DAC 2001*, pp 661-6.
- [34] C. Visweswariah, K. Ravindran, K. Kalafala, S.G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," *DAC 2004*, pp 331-336.
- [35] S. J. Kang, C. Kocabas, H. S. Kim, Q. Cao, M.A. Meitl, D. Y. Khang, and J.A. Rogers, "Printed multilayer superstructures of aligned single-walled carbon nanotubes for electronic applications," *Nano Letters*, v 7, n 11, Nov. 2007, p 3343-3348.
- [36] E. Pop, "The role of electrical and thermal contact resistance for Joule breakdown of single-wall carbon nanotube," *Nanotechnology*, v 19, n 29, 23 July 2008, p 295202 (5 pp.)
- [37] Y. Lin, M. Hutton, L. He, "Placement and Timing for FPGAs Considering Variations," *Field Programmable Logic and Applications*, 2006. *FPL '06. International Conference on* , vol., no., pp.1-7, 28-30 Aug. 2006
- [38] International Technology Roadmap for Semiconductors, <http://www.itrs.net/>.