Circuit and System Architecture for DNA-Guided Self-Assembly of Nanoelectronics

Jaidev P. Patwardhan†, Chris Dwyer†, Alvin R. Lebeck†, and Daniel J. Sorin‡
{jaidev,dwyer,alvy}@cs.duke.edu, sorin@ee.duke.edu

†Dept. of Computer Science
Duke University
Durham, NC 27708

‡Dept. of Electrical & Computer Engineering
Duke University
Durham, NC 27708

Abstract

This paper explores the architectural challenges introduced by emerging bottom-up fabrication of nanoelectronic circuits and develops an architecture that meets these challenges. While our implementation is based on one specific technology, we believe the architecture is compatible with other emerging technologies. The specific nanotechnology we explore uses patterned DNA nanostructures and carbon nanotube FETs to create a hierarchical design. Patterned DNA nanostructures provide a scaffold for the placement and interconnection of CNFETs to create a limited size circuit (node). These nodes are interconnected using DNA-guided self-assembly, but without the control available in the patterned nanostructures, thus producing a random interconnect. Three characteristics of this technology that significantly impact architecture are 1) limited node size, 2) random node interconnection, and 3) high defect rates. We present an accumulator-based active network architecture that addresses these three challenges.

1 Introduction

Technology change is fuel for architectural innovation. Evolutionary changes in CMOS have inspired research on several important topics including wire dominated designs, power dissipation, and fault tolerance. A revolutionary technology change, such as replacing CMOS, is a potentially disruptive event in the design of computing systems. Emerging technologies for further miniaturization have capabilities and limitations that can significantly influence computer architecture, and require re-examining or rebuilding abstractions originally tailored for CMOS. This paper explores the architectural challenges introduced by emerging bottom-up fabrication of nanoelectronic circuits and develops an architecture that meets these challenges.

In Section 2, we describe the one specific nanotechnology on which we focus in this paper: DNA-guided self-assembly of carbon nanotube devices and wires. However, we believe our architecture is applicable to a broader class of technologies with similar characteristics. Carbon nanotubes have been used to create transistor behavior [3, 17], thus maintaining the same device abstraction as used in CMOS with the potential of greater device densities, lower cost, and lower power consumption. Unfortunately, current top-down fabrication techniques (e.g., photolithography) cannot precisely place or interconnect components as small as carbon nanotubes (which have diameters on the order of a couple of nanometers).

In Section 3, we discuss the impact of this nanotechnology on circuit design. Circuit designs must balance three competing issues: regularity, complexity, and defect tolerance. A circuit design must use regular patterns of structures to simplify the DNA self-assembly and avoid defects in it. A circuit, however, also requires some amount of complexity if it is to perform useful computation. Lastly, circuits that are self-assembled with DNA are liable to have defects that must be tolerated.

To overcome these challenges and develop nanoelectronic circuits, we propose in Section 4 to use patterned DNA nanostructures [38] as a scaffold to which we can attach carbon nanotubes. The DNA nanostructures create a limited size circuit (node) of carbon nanotube transistors (CNFETs). DNA-guided self-assembly can also pro-

vide a scaffold for metal that forms the interconnect between nodes [38], but without the control available in the patterned nanostructures, thus producing a random interconnect.

In Section 5, we discuss the architectural implications of this technology and other technologies with similar characteristics. In particular, we identify three aspects of this technology that significantly impact architecture: 1) limited node size, 2) random interconnection of nodes, and 3) high defect rates. Our goal is to develop an appropriate architecture that can be implemented in any technology with these three characteristics. We also enumerate several important issues that must be addressed during architectural development.

As an initial solution to the above three challenges, in Section 6 we develop an active network architecture with an accumulator-based ISA. The limited node size prevents the design of a single node that can perform all operations. Instead, we design several different node types (e.g., add, memory, shift) based on node size constraints. To execute, an instruction searches for a node with the appropriate functionality (e.g., add), performs its operation, and passes its result to the next dependent instruction. In this active network execution model, the accumulator and all operands are stored within a packet rather than at specific nodes, thus reducing per node resource demands. The active network execution model enables us to encode a series of dependent instructions within a single packet, and it enables execution to take an arbitrary path through the network. A configuration phase at system startup maps out defective nodes and links, organizes a memory system, and configures routing options within the network. This architecture matches our technology characteristics since it 1) allows for differing node types with specialized functionality, 2) tolerates a random interconnection of nodes, and 3) tolerates node and interconnect fabrication defects.

2 Emerging Nanotechnologies

In this section, we describe the specific nanotechnologies used in this paper. We discuss the electronic components (Section 2.1), DNA self-assembly of these components into circuit nodes (Section 2.2), and the large-scale interconnection of these circuit nodes (Section 4.4).

2.1 Carbon Nanotube Electronics

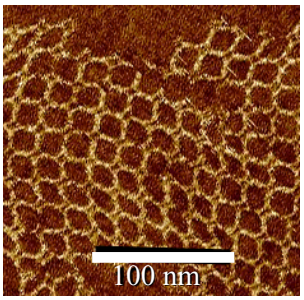
There are many choices for constructing nanoelectronic devices and nanowires. One such promising nanoelectronic device is a carbon nanotube field effect transistor [31], in which the application of a gate voltage [13, 34, 31] can modulate the conductivity of a semiconducting nanotube. Recent advances enable separating metallic nanotubes from semiconducting nanotubes and controlling the length of individual nanotubes [30, 39, 40, 25]. Therefore, we can use both types of carbon nanotubes to construct logic gates, memory (e.g., with cross-coupled NOR gates), and circuit interconnect. To explore the potential of CNFETs, we simulate several circuits using a customized SPICE 3f5 kernel that can model CNFET behavior in logic gates. Preliminary comparisons against a 45nm predictive CMOS model and the 18nm ITRS prediction indicate that CNFET circuit performance will likely be better. The added benefit that CNFETs are amenable to self-assembly makes this an attractive alternative, or supplement, to silicon device technology.

2.2 DNA Tiles and Nanostructures

To overcome the challenge of nanoelectronic integration, we exploit the ability of DNA self-assemblies to produce patterned nanostructures. We use DNA self-assemblies to create a patterned scaffolding onto which we can programmably attach carbon nanotubes. DNA's well-known double-helix structure is formed through its well-understood base-pairing rules—adenine (A) to thymine (T) and cytosine (C) to gua-

nine (G). By specifying a particular sequence of base pairs on a single strand of DNA, we can exploit the base-pair rules as organizational instructions. We use unpaired, single-strand regions (ssDNA) extending from the ends of one nano-scale object to specifically bind to another object displaying the complementary ssDNA. Therefore, a region of ssDNA and its complementary ssDNA act as tags[1] (T and T') for orienting the two objects in 3-space. These tags are a key feature of DNA self-assembly techniques. By appending DNA to nano-scale objects it can act as "smart glue" for organizing those objects in space [29].

As scaffolding for nanoelectronic integration, we plan to use DNA tiles [35,22] that are nanostructures composed of several DNA strands. Most of the strands bind together to form a complex of double-stranded DNA. Each tile can have a small number of ssDNA tags for binding with other tiles. During annealing, the tiles bind together to form regular structures in two dimensional space. Although the bindings and resulting structure can be used to perform computation [1, 27], we are interested in DNA's ability to self-assemble into large-scale nanostructures. DNA tiles can be assembled into large 2-dimensional sheets or lattices by properly designing their tags. Lattices composed of hundreds of thousands of tiles and extending up to at least 10 microns on their long edge have been created [22, 35].



AFM of DNA Lattice

FIGURE 1. A DNA scaffolding for carbon nanotube circuits

For this paper we focus on a particular structure that creates a 'waffle'-like lattice with repeating cavities of ~16x16nm and 4nm separation between cavities [38, 37]. Figure 1 shows an atomic force microscopy (AFM) image of a lattice.[2] This type of lattice has been experimentally demonstrated and can achieve sizes that extend beyond 1 micron on each side (i.e., more than 50 cavities on a side).

To attach the carbon nanotube devices to the DNA tile scaffolding, we plan to adopt a recently demonstrated technique for attaching ssDNA tags to carbon nanotubes [9]. We can attach ssDNA tags to nanotubes (a process called functionalization[3]), and these tags will bind to the complementary ssDNA strand protruding from the DNA tile scaffolding. While the demonstrated operation of CNFETs and the ability to functionalize them make them a promising option for nanoelectronic devices, the DNA self-assembly technique is independent of the specific nanoelectronic device used.

3 Implications for Nanoelectronic Circuit Architecture

To use the technology described in Section 2, the nanoelectronic circuit architecture must strike a balance between 1) the *regularity* of DNA self-assembly patterning capabilities, 2) the *complexity* required for sophisticated system designs and 3) *tolerance* to the inevitable defects present in nanoscale systems. The remainder of this section elaborates on each of these issues, and we focus on the fundamental differences between this nanoarchitecture and current CMOS based architectures.

---

1. These codes are sometimes called sticky-ends due to their glue-like binding behavior.

2. AFM image courtesy of Thomas LaBean.

3. Functionalization is the process of adding a molecule to another material by chemical reaction.

3.1 Regularity

While the design of CMOS based circuits can be simplified by the use of regularity (e.g., standard cell VLSI), regularity is not a fundamental requirement. However, as described previously, DNA self-assembly technology can currently create only periodic arrays of identical unit cells. DNA self-assembly has a potential limitation in that the probability of incorrect tag matches increases as the number of unique tags increases. For each type of connection, we need a unique pair of complementary ssDNA tags. With more types of connections and a fixed number of base-pairs per tag, the tags become more similar (i.e., differ in fewer base-pairs) and partial matches become more likely. For example, if a functionalized nanotube binds to a partially matched tag, then it is in the wrong position. This situation is analogous to the Hamming distance between encodings of symbols; if we need to encode more symbols with the same number of bits, then the Hamming distance is smaller and the probability of an error is greater. Minimizing the number of tags reduces the chances of partial matches, which could cause positional defects, during annealing. Therefore, repetitive structures are desirable, and circuit and system designers should strive to use them as much as possible.

3.2 Complexity

Design complexity is a function of the number of different component types and the placement of these components. Current CMOS based circuits can arbitrarily place hundreds of millions of devices (both nFET and pFET) and wires with precision on the order of 0.10μm. This precision is achieved by using photolithography to specify exactly where each individual component belongs. With the combination of carbon nanotube devices and DNA self-assemblies, we are trying to develop circuits that are complex enough to perform interesting computation. We can limit the number of component types to just CNFETs, nanotube wires, and metal plating for connecting wires. However, with DNA self-assemblies, we cannot specify component placement with nearly the same degree of complexity as CMOS. Complexity must be introduced without requiring a large number of tags. This mirrors the desire to use regular structures that minimize the number of tags. However, regular structures typically limit complexity.

Thus, the utility of self-assembled DNA arrays depends on the amount of complexity that we can introduce at various abstraction levels without causing an intractable number of partial matches. Consider the graph generated from the netlist of a transistor-level design of a combinational circuit. The vertices are transistor terminals and the edges are wires connecting the device terminals. A two-input CMOS NAND gate has about ten vertices. Clearly, the naive approach of assuming a unique tag for each vertex in the graph requires a large number of unique tags (even ignoring fan-out issues). This will cause too many partial matches that create bridging faults (shorts), rendering the circuit mostly useless.

3.3 Defect Tolerance

A defect is a permanent physical fault that was introduced during fabrication. We consider two types of defects: functional and positional. A *functional defect* corresponds to a component that does not perform its specified function (e.g., a transistor that does not conduct when it should). A *positional defect* corresponds to a (functionally correct) component that is placed incorrectly. Both CMOS and DNA self-assembled nanoelectronics can incur functional defects, but only self-assembly is likely to incur positional defects. Positional defects can be both defects of omission and commission. An omissive positional defect occurs when a component is not placed where it belongs. A commissive positional defect occurs when a component is placed where it does not belong (i.e., the partial match described above). Omissive defects behave

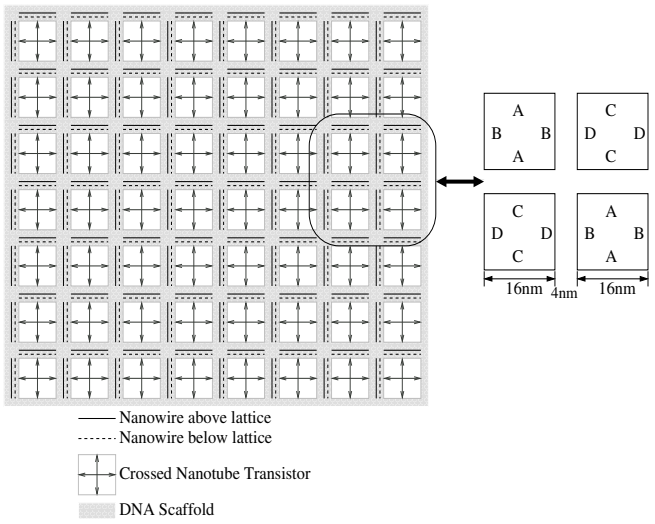


FIGURE 2. DNA Scaffold for Nanoelectronics

similar to functional defects. Commissive defects are more dangerous, since they can behave like bridging faults. For example, a misplaced nanowire could cause a short between power and ground or it could change circuit functionality in unpredictable ways (e.g., by erroneously connecting the output of a gate to its input).

In CMOS based circuits, there is limited support for defect tolerance. Photolithographic placement of components is a mature technology that incurs few defects. However, in architectures with hundreds of millions of devices and wires, defects will still occur with some probability (i.e., yield is less than 100%). CMOS microchips are thus tested for defects. If a defect is uncovered and it cannot be tolerated, the chip is discarded. However, some limited number of defects can be tolerated. For example, a defect in a cache or memory cell can be tolerated by systems that provide redundant cells and allow for re-mapping.

Functional defect rates for carbon nanotube devices and positional defect rates for DNA assembled nanoelectronics are currently unknown due to the relative immaturity of the technologies. Functional defect tolerance can be achieved with the same techniques used in CMOS, since the problem is not fundamentally different. Tolerance of commissive positional defects, however, is a new challenge. Because of the unknown positional defect rates, our approach is to first strive to minimize positional defects by exploiting regularity in DNA self-assemblies. However, as complexity increases and regularity decreases, the probability of positional defects increases. Thus, more sophisticated circuitry will require more defect tolerance.

4 Nanoelectronic Circuit Building Blocks

This section describes a proposed nanoelectronic circuit architecture (shown in Figure 2) with structures based on a grid of CNFETs interconnected with conducting carbon nanotubes. At a high level, our proposed design addresses the conflicting goals of regularity and complexity by placing identical unit cells in the cavities of an aperiodic patterned DNA lattice. The lattice is regular in structure, but it has aperiodic binding points which we will use to connect the unit cells in complex patterns. This highlights a key difference between our work and existing approaches. Current nanoelectronic architectural approaches assume regularity in both the structure and the interconnect. We first present our initial proposed unit cell and then the proposed lattice. This is followed by an example design of a full adder. We then discuss how to self-assemble multiple building blocks into a larger system.

4.1 Exploiting Regularity: A Replicated Unit Cell

The unit cell in our design is a three terminal pFET sitting in the cavity of a DNA lattice. We could place a complete NOR gate in the cavity, but we leave that as future work. To place the pFET in the cavity, we need to functionalize one semiconducting and one conducting nanotube such that they bind to the complementary ssDNA tags on the cavity edges and form a cross. We assume one of the nanotubes is wrapped in a thin insulating layer, such as SiO_2 [12]. The conducting nanotube functions as the gate of the pFET.

Using carbon nanotubes of a short length (~16nm) precludes commissive positional defects in which a carbon nanotube binds in two different cavities. By using two sets of tags in alternating cavities in each dimension (see Figure 2) and by using carbon nanotubes of a precise length, a nanotube cannot span across the DNA lattice to another cavity with the same tags. The distance between adjacent cavities is only 4nm, so if the same tag is used in adjacent cavities, then a nanotube may bind across the lattice arm rather than within a cavity. Using a checkerboard pattern of alternating tags, with sufficient Hamming distance, eliminates positional commissive defects. This approach requires carbon nanotubes of a precise length, which may be possible using a sonochemical method [24] to cut the originally long nanotubes into short lengths and then using size-exclusion chromatography to separate the nanotubes by their length. This technique must be applied to both the semiconducting and conducting nanotubes.

We augment this unit cell with short conducting carbon nanotubes that lie adjacent to the cavity on both the top and bottom of the DNA lattice. The short nanotubes are far enough apart to avoid cross-talk and may also be wrapped with an insulating polymer if necessary. The nanotubes initially do not intersect to form complete circuits. Instead, an electrical connection between nanotubes must be explicitly created by specifying an appropriate tag on the DNA lattice to which a gold nanosphere will bind. The nanosphere nucleates metal ions to form the connection with the help of an electroless plating process [5, 20]. Similarly, connecting transistors may require specifying whether the device connects to the top or bottom conducting nanotube. Forming these connections is where we add complexity to our design, and we explain how to introduce this non-regular patterning in Section 4.2.

This unit cell design fosters regular, repetitive structures. All nanotubes are the same length (16nm) and we have five sets of nanotubes that are functionalized with different tags. Four sets of nanotubes are used for the CNFETs; two semiconducting sets and two conducting sets. This corresponds to the two tag sets of the checkerboard pattern of cavity tags. A nanotube from one set can bind to any cavity with the complementary tag. Similarly, the interconnect nanotubes (the fifth set) can bind adjacent to any cavity directly on either the top or bottom of the DNA lattice in either the vertical or horizontal direction. This approach enables the use of a regular pattern for the base DNA lattice scaffolding.

4.2 Introducing Complexity: An Aperiodic Pattern for Interconnecting Cells

Our building block, while regular in structure, has aperiodic binding points for connecting together the nanowires of the unit cell. We plan to achieve this aperiodic pattern through either sequential assembly of tiles or extending recent work on one-dimensional aperiodicity [36] to two dimensions.

We can now construct complex circuits by specifying the electroless plating points in the DNA lattice. For each of the top and bottom of the lattice, the plating point options include: the three transistor terminals to nanowire, interconnect nanowire in the vertical directions North and South, and interconnect nanowire in the horizontal directions East and West. We assume that to create a straight-through connection in

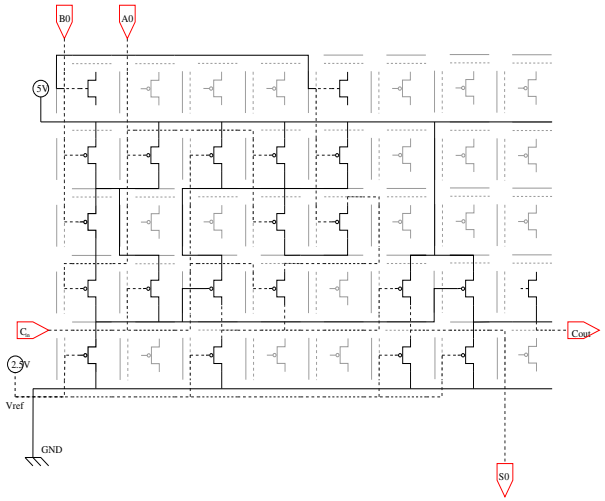


FIGURE 3. 1-bit Adder

the vertical direction requires both the North and South connections; similarly, both the East and West connections are required for a straight connection in the horizontal direction. We can build pass-throughs from the top-level interconnect to the bottom-level by connecting a transistor terminal to both interconnects.

Only a single tag on the DNA lattice is required to specify the plating points where the gold nanospheres can bind on the lattice. It is this tag that has the aperiodic pattern, and gold will bind only where the tag appears. We note that this approach minimizes positional defects since the nanotubes are of specific lengths that can only bind in the appropriate positions of the lattice. In contrast, if we used long nanowires to connect distant points, then the number of tags to which they could potentially incorrectly bind is the number of tags on the circumference of a circle with radius equal to the nanowire's length.

4.3 A Proposed Adder

Given the above ability to connect nanodevices and nanowires together, we can create a complete circuit. This subsection presents the design of a full adder. Figure 3 shows one design that uses only p-type transistors[4] and a corresponding layout on an eight by six DNA lattice. This layout was performed manually with minimal effort to optimize the layout. From this diagram we see that this design uses only 18 transistors out of the 48 available transistors. With only two levels of interconnect and that interconnect sharing a unit cell with a transistor, we are often forced to leave the transistor unused while the unit cell nanowires provide connectivity for the circuit. We are exploring CAD tools to automate and optimize the layout. In this design, power and ground must be explicitly routed with our interconnect, consuming additional routing options and unit cells. We are currently investigating alternative approaches to providing power and ground planes. One such approach is to sandwich the DNA lattice between two insulating layers that permit only large metallic nanospheres to protrude through the layers. A metallic layer on each side of the insulated DNA lattice can serve as the power and ground electrodes.

4. CNFETs are naturally p-type, but research has demonstrated the ability to electrostatically dope them to be n-type [3]. Future research will explore complementary logic designs with both n-type and p-type devices.

The above full adder is designed for easy expansion to an arbitrary n-bit adder. The carry out of one bit directly aligns with the carry in of the next higher-order bit. Therefore, no additional overhead, in terms of lattice cells, is incurred to support simple multi-bit ripple carry adders.

4.4 Large-scale Interconnection of Circuit Nodes

The computational capabilities of the proposed building block (node) is limited by the size of the DNA lattice. Increasing the computing capacity requires interconnecting multiple building blocks. Using inexpensive laboratory equipment we could simultaneously self-assemble as many as 10^{12} identical, but small, nodes. This number of nodes, if placed on a two micron pitch, would cover a 175 cm **X** 175 cm area, or the equivalent area of ~40 x 300 mm wafers. Although the size of an individual node is well above the minimum feature size of photolithography, the number of nodes fabricated through self-assembly limits how heavily the overall process can rely on silicon fabrication processes. Self-assembling nodes onto a substrate at well-defined places is also difficult without "naming" each placement site (pick and place methods will not scale to this number of components). Even with DNA tags on the substrate, the nodes are not guaranteed to fall into place precisely. Most conventional architectures require precise placement and interconnection between circuits. Therefore, even if we could use a conventional photolithographically patterned network to interconnect nodes, the result would be a random interconnection due to the random placement of nodes on the substrate. This is the sacrifice a self-assembly process imposes: precision and control exist only at small length scales (e.g., < 1 micron, for now). Our solution to this problem involves a large scale self-assembling process that can interconnect nodes on a substrate using another form of DNA-guided self-assembly. Individual DNA strands self-assemble between node edges, providing a scaffold for metal that forms an electrical connection, which has been experimentally demonstrated [23, 38]. This larger scale process cannot deliver the precise control found in the earlier process used to assemble the nodes, but it can fabricate single wire interconnections between the edges of the nodes, as illustrated in Figure 4.

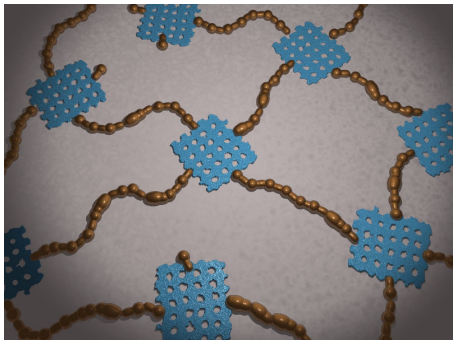


FIGURE 4. Schematic rendering of a self-assembled DNA interconnection network after metal deposition

5 Architectural Implications

The process of using DNA-guided self-assembly to create nanoelectronic circuits presents several challenges that must be addressed when designing a system. The three primary aspects of the process are 1) small-scale control of placement and connectivity within a single node, 2) large-scale randomness in node placement and interconnection, and 3) high defect rate. These three aspects significantly impact architectural decisions, particularly since conventional architectures assume precise control at both the small and large-scale.

5.1 Limitations of Small-scale Control

The ability of DNA-guided self-assembly to achieve only small-scale control impacts architectural decisions in several ways. Three of the most significant are: limited space, limited coordination, and limited communication.

Limited space. The 50x50 node can have a maximum of 2500 CNFETs, however it is unlikely that the usable number will reach even 40% of this. On-node interconnect will reduce efficiency since a node only has two-levels of interconnect. Furthermore, a portion of each node must be allocated as a "pad" for the DNA interconnect to other nodes. These two factors can dramatically reduce the usable area on a node. This limited node size presents a trade-off in node design. At one extreme, we could design just a single node type that contains both computation and storage capabilities. However, since the storage and computation circuits must share the node, each may be severely limited in capability. Alternatively, we could design a few specialized node types, some devoted to computation and others devoted to storage. Even when designing a specialized node, the limited space impacts architectural decisions. For example, large state machines are not an option since there is insufficient space for state storage. Similarly, the number of bits available in a storage node may be limited, thus affecting an architecture's word size.

Limited communication. Without large-scale control, there is limited communication among nodes. Each node has four neighbors and there is no long haul communication. Furthermore, the connections from a node to each of its neighbors is limited to a single wire. Although the degree of each node or the number of connections between neighbors could be increased, each connection occupies precious edge space. Conventional designs exploit multiple metal layers for long-haul communication and large-scale control to create multi-wire connections between components. Therefore, the architecture must avoid relying on sophisticated communication hardware.

Limited coordination. Conventional CMOS designs rely on precise control during fabrication to create sophisticated circuits (e.g., a 64-bit adder with carry lookahead). For our technology, if the most sophisticated node is a full-adder, then it is unlikely that 64 such nodes can be coordinated to implement a 64-bit adder. Coordination among nodes is very limited and it is difficult to *a priori* configure a group of nodes to operate in a coordinated manner. Each node can perform only limited coordination with its immediate neighbors.

5.2 Large-Scale Randomness

Our proposed self-assembly process provides excellent control at the small-scale, however it cannot achieve such control at large scales. The resulting randomness introduces some additional issues that architectures must address.

Node placement. The self-assembly process does not guarantee where any particular node will lie in the final circuit. Each node simply attempts to connect to other nearby nodes. The architecture and machine organization must accommodate this arbitrary placement of functional blocks.

Node orientation. Similar to the random node placement, the assembly process we envision does not provide control over node orientation. Any system design must tolerate arbitrary node orientations, and cannot make *a priori* assumptions on orientation.

Node connectivity. Connections between nodes are not guaranteed to succeed during self-assembly. Therefore, it is possible for any node to have between zero and four functioning connections to its neighbors. The architecture must not make any *a priori* assumptions about available connectivity. When combined with random orientation, it is possible for nodes to connect in a triangular shape rather than the 2x2 grid one would assume with nodes that have degree four.

5.3 High Defect Rates

An inherent aspect of any self-assembly process is defects. These fabrication defects can influence node functionality and connectivity. Some of the interconnect defects cause the above problems with connectivity. While some aspects of fabrica-

tion can reduce the likelihood of defects (e.g., purification steps or overdesign of DNA tags), there will always be a significant number of defects and any architecture using these technologies must tolerate these defects.

5.4 Architectural Challenges

The above discussion exposes several aspects of this fabrication technique for nano-scale circuits that must be addressed by any architecture and its corresponding implementations. In this subsection, we enumerate several important challenges to developing an appropriate architecture for this emerging technology. This list is not meant to be exhaustive, but rather to highlight some important challenges.

Node Design. The architect must decide what functionality to place in each node. Should there be homogeneous nodes or heterogeneous nodes? If heterogeneous, then what types of nodes? How does node design affect connectivity/communication with other nodes, and what primitives should be provided?

Utilizing Multiple Nodes. Since individual nodes do not contain sufficient computation and storage to perform much useful work in isolation, then an architect must determine how to exploit multiple nodes. This must be achieved given the above limitations on coordination, communication, placement, orientation, and connectivity.

Routing with Limited Connectivity. Traditional routing techniques may not directly apply since there is limited space for the complexity of dynamic routing and there are insufficient guarantees on node placement and connectivity to use conventional static routing. The designer must develop a routing technique that overcomes these challenges.

Developing an Execution Model. The execution model embodies the software visible aspects of the architecture and can be influenced by implementation constraints or instruction set requirements. For the envisioned fabrication technique, the execution model must overcome the severe implementation constraints outlined above while still enabling a reasonable instruction set.

Developing an Instruction Set. Programmable systems require an interface that enables software to specify operations. Typically this is achieved by the instruction set architecture (ISA). The ISA may be influenced by the underlying capabilities of the technology. Given our fabrication technique, the architect must design an appropriate ISA that supports the above execution model.

Providing a Memory System. Storage is a crucial component of most computing systems regardless of the execution model. The ability to retain values for future use and to name and find particular values is necessary for most computing paradigms.

Interfacing to the Micro-scale. An important aspect of any nano-scale system is the interconnection to larger-scale components (e.g., micro-scale). This connection is necessary for at least providing an I/O interface for communication with the outside world. It may be possible for the architecture to exploit this interface in other ways.

The challenge is to address each of these issues such that we arrive at a functioning system. There are likely many possible approaches to developing a functioning system. In this paper, we adopt the philosophy of "make it work first, optimize later."

6 An Architecture for Self-Assembled Nano-electronics

As an initial approach for addressing the issues raised in Section 5, we propose an architecture that is compatible with our fabrication technology. The architecture is like an active network [32] in that execution packets that contain instructions and operands search through a loosely structured sea of processing and memory nodes for the functionality that they need at each step of execution. This architecture matches our technology characteristics since it 1) allows for differing node types with special-

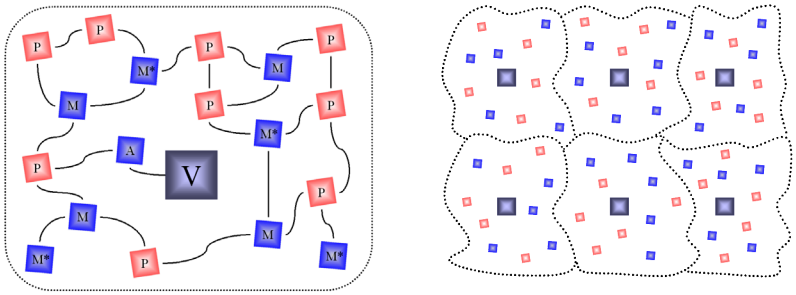


FIGURE 5. System Model. Processing nodes (P), memory nodes (M), memory port nodes (M*), anchor node (A), and viaduct (V). This schematic is not to scale (w.r.t. nodes per cell)

ized functionality, 2) tolerates a random interconnection of nodes, and 3) tolerates node and interconnect defects.

6.1 System Overview

The system model (see Figure 5) is a random interconnection of various node types, in which each node contains circuitry for communication and some specialized circuitry (e.g., processing, memory, etc.). Groups of nodes are organized into *cells*. A node communicates with a neighboring node via a single link that is asynchronous and bidirectional (time-multiplexed on a single physical wire). Each cell has a *viaduct* that is its connection to a micro-scale, and one of the many nodes connected to the viaduct acts as the *anchor node* for the cell. Inter cell communication occurs through a micro-scale interconnection network. The memory nodes in each cell comprise a portion of the global memory space. Some fraction of memory nodes are configured as *memory ports* to provide an interface between execution packets and memory storage.

To impose structure on the interconnection network and the memory system, there is a configuration phase that occurs before any execution. Reconfigurable architectures have demonstrated that this approach is important to achieve high performance in the context of highly focused (i.e., aggressive) or highly defective technologies, including nanotechnology. We describe the purpose, beyond defect tolerance, and operation of the configuration in detail later in this section.

While node functionality is heterogeneous, all nodes have some common responsibilities. Each node generates its own local clock and communicates asynchronously with its neighboring nodes using signaling techniques similar to push-style pipeline systems. High level communication between two devices over a single wire can be managed using simple two- and four-phase single wire techniques [33]. Each node must also contain routing functionality for determining the outgoing link for an incoming packet (or the result of an operation).

6.2 Execution Model

The execution model relies on an accumulator-based ISA. Conceptually, the accumulator is initialized and then a sequence of operations are performed on the corresponding series of operands. The accumulator-based ISA reduces the need for widespread *a priori* coordination and communication among many components, since the only data dependence involves the accumulator and instructions are processed in order [21]. We support the accumulator-based execution by forming an *execution packet* that contains the operations, the accumulator, and all operands in

appropriate order. The packet simply searches within a cell for the requisite functional units (or memory ports) in the order specified by the operation sequence.

Logically, each functional unit performs its specified operation, removes the operand and forwards the new accumulator and the remaining operands to the subsequent functional units. Each subsequent functional unit performs a similar sequence until all operations in the packet are completed. Memory operations generate *memory packets* that are handled by the memory ports. When a given execution packet finishes, it instantiates the next execution packet in a process called *chaining*. Chaining involves a conditional test at the end of the packet execution to determine what packet to instantiate next.

Our system and execution model enables significant parallelism by instantiating multiple execution packets within a cell and in multiple cells. While this parallelism is an important aspect of our architecture that fully exploits the capabilities of the underlying technology, in our initial work we focus primarily on the operation of a single cell and sequentially instantiate execution packets.

6.3 Interconnection Network: Finding Resources for Execution

The active network architecture must enable packets to find what they need without deadlocking or livelocking, despite high defect rates and traveling through a randomly interconnected sea of nodes. To avoid request/response deadlock (i.e., fetch deadlock), we support three logical networks: one for execution packets, one for memory request packets and one for memory response packets. We implement these logical networks using a combination of virtual channels [6] and separate physical networks for memory and execution. Each logical network uses back pressure flow control and up*/down* routing to provide deadlock free routing [28].

To implement these techniques with our limited node capacity, we equip each node with two forms of communication: 1) broadcast and 2) routing along a *gradient* [19, 18]. The gradients represent a spanning tree with a viaduct as the root. This tree is used for up*/down* routing. During the configuration phase, a gradient is established by initiating a broadcast at a viaduct with a specialized packet. A node only forwards the first packet it receives. Creating spanning trees using a broadcast flood maps out defective nodes and links, since no other node will have a gradient pointer to the defective node. We use five gradients in our system: one for each planar direction (north, south, east, and west) and an additional gradient that establishes the cell boundaries and the direction toward the viaduct within each cell (called the *cell gradient*). Each cell represents a local name space for memory and includes both data and instructions. As part of the configuration phase, memory locations and memory ports are allocated and the physical memory network is established. Further details of the configuration and routing are beyond the scope of this paper.

7 Current Status

We are currently pursuing the various aspects of fabricating a simple circuit, including: carbon nanotube length control and separation and aperiodic DNA lattices. We are also developing a tool chain to aid in the design and evaluation of our circuits and systems. In particular, we have initial SPICE simulations of CNT circuits, and we are developing a circuit layout tool to automate design. This layout tool will also provide input for SPICE and a custom DNA sequence generator. We are developing a custom simulator to evaluate the architectural aspects of our system. So far, we have simulated the correct operation of all configuration phases and the correct execution of a two packet sequence in a single cell.

8 Related Work

The most related work is Dwyer's proposal to use a DNA guided self-assembly technique to build a massively parallel computer [10, 8]. The proposed machine has no communication between processing elements and thus targets problems that are "embarrassingly parallel." Goldstein's inspiring work on nanofabrics leverages reconfigurable self-assembled nanoelectronics to provide a defect tolerant architecture [14]. Resonant tunneling diodes (two terminal devices) are configured into supernodes of appropriate functionality after a test phase maps out defective components. The nanofabric is reconfigured for each program that executes. DeHon presents an architecture that exploits three terminal devices (FETs) by self-assembling arrays of nanowires and FETs [7]. Sparing and remapping are used to provide defect tolerance. Heath et al.'s Teramac design reconfigures redundant nanoscale components, with high individual defect probabilities, into a functional system [16]. Han exploits NAND multiplexing and reconfiguration to support a defect tolerant architecture [15]. More generally, Nikolic et al. argue that reconfiguration is the best approach for handling fabrication defects, but that other redundancy techniques are necessary to handle transient faults [26]. Ancona proposes a systolic array architecture for single electron transistors [2], but it requires precise control over fabrication. Beckett proposes a nanoarchitecture based on integrated processing and memory nodes with a local interconnection [4]. Fountain's propagating instruction processor is a pipelined SIMD machine [11].

9 Conclusions

In this paper, we have explored the effects of emerging nanotechnologies on computer architectures. We discussed one particular set of technologies that we use for designing nanoelectronic circuits, and we highlighted the differences between circuits in this technology and CMOS. We then presented an architecture that addresses the challenges posed by DNA-based self-assembly of carbon nanotubes and other nanotechnologies with similar characteristics. To overcome (1) limited node size, (2) random interconnection of nodes, and (3) a high defect rate, we developed an active-network architecture with an accumulator-based ISA. This architecture enables execution packets to search through a sea of heterogeneous nodes for the functionality they need, while avoiding defective nodes. We use an initial configuration phase to impose some limited structure on the computing substrate, particularly for routing and memory allocation. While this architecture is only a relatively unoptimized first step, it addresses some of the key challenges in this class of nanotechnology and it highlights the technology's architectural implications.

10 Acknowledgments

This work is supported primarily by an NSF ITR grant CCR-0326157 and a grant from the Duke University Provost's Common Fund. Other support includes NSF EIA-9972879 and equipment donations from IBM and Intel. We thank the many members of the TROIKA project: Thomas LaBean, Hao Yan, John Reif, Jie Liu, Sean Washburn, Dorothy Erie, and Paul Franzon.

References

[1] L. Adleman. Molecular Computation of Solutions to Combinatorial Problems. *Science*, 266(5187):1021–1024, November 1994.

[2] M. G. Ancona. Systolic Processor Designs Using Single-Electron Digital Circuits. *Superlattices and Microstructures*, 20(4), 1996.

[3] Adrian Bachtold, Peter Hadley, Takeshi Nakanishi, and Cees Dekker. Logic Circuits

with Carbon Nanotube Transistors. *Science*, 294:1317–1320, November 2001.

[4] Paul Beckett and Andrew Jennings. Toward Nanocomputer Architecture. In *Proceedings of the Seventh Asia-Pacific Computer Systems Architecture Conference*, pages 141–150, 2002.

[5] Erez Braun, Yoav Eichen, Uri Sivan, and Gdalyahu Ben-Yoseph. DNA-Templated Assembly and Electrode Attachment of a Conducting Silver Wire. *Nature*, 391:775–778, 1998.

[6] William J. Dally. Virtual Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[7] Andre DeHon. Array-Based Architecture for FET-Based, Nanoscale Eletronics. *IEEE Transactions on Nanotechnology*, 2(1):23–32, March 2003.

[8] C. Dwyer. *Self-Assembled Computer Architecture: Design and Fabrication Theory*. PhD thesis, University of North Carolina, May 2003.

[9] C. Dwyer, M. Guthold, M. Falvo, S. Washburn, R. Superfine, and D. Erie. DNA Functionalized Single-Walled Carbon Nanotubes. *Nanotechnology*, 13:601–604, 2002.

[10] C. Dwyer, L. Vicci, J. Poulton, D. Erie, R. Superfine, S. Washburn, and R. M. Taylor. The Design of DNA Self-Assembled Computing Circuitry. *IEEE Transactions on VLSI*, To appear 2003.

[11] T. J. Fountain, M. J. B. Duff, D. G. Crawley, C. D. Tomlinson, and C. D. Moffat. The Use of Nanoelectronic Devices in Highly-Parallel Computing Systems. *IEEE Transactions on VLSI Systems*, 6(1):31–38, 1998.

[12] Qiang Fu, Chenguang Lu, and Jie Liu. Selective Coating of Single Wall Carbon Nanotubes with Thin SiO2 Layer. *Nano Letters*, 2(4):329–332, 2002.

[13] M. S. Fuhrer, J. Nygard, L. Shih, M. Forero, Young-Gui Yoon, M. S. C. Mazzoni, Hyoung Joon Choi, Jisoon Ihm, Steven G. Louie, A. Zettle, and Paul L. McEuen. Crossed Nanotube Junctions. *Science*, 288:494–497, April 2001.

[14] Seth C. Goldstein and Mihai Budiu. NanoFabrics: Spatial Computing Using Molecular Electronics. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*, pages 178–191, July 2001.

[15] Jie Han and Pieter Jonker. A Defect- and Fault-Tolerant Architecture for Nanocomputers. *Nanotechnology*, 14:224–230, January 2003.

[16] James R. Heath, Philip J. Kuekes, Gregory S. Snider, and R. Stanley Williams. A Defect-Tolerant Computer Architecture: Opportunities for Nanotechnology. *Science*, 280:1716–1721, June 1998.

[17] Yu Huang, Xiangfeng Duan, Yi Cui, Lincoln J. Lauhon, Kyoun-Ha Kim, and Charles M. Lieber. Logic Gates and Computation from Assembled Nanowire Building Blocks. *Science*, 294:1313–1317, November 2001.

[18] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[19] David B Johnson and David A Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[20] K. Keren, M. Krueger, R. Gilad, G. Ben-Yoseph, U. Sivan, and E. Braun. Sequence-Specific Molecular Lithography on Single DNA Molecules. *Science*, 297:72, 2002.

[21] Ho-Seop Kim and James E. Smith. An Instruction Set and Microarchitecture for Instruction Level Distributed Processing. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, May 2002.

[22] Thomas H. LaBean, Hao Yan, Jens Kopatsch, Furong Liu, Erik Winfree, John H. Reif, and Nadrian Seeman. Construction, Analysis, Ligation, and Self-Assembly of DNA Triple Crossover Complexes. *Journal of the American Chemistry Society*, 122:1848–1860, 2000.

[23] D. Liu, J.H. Reif, and T.H. LaBean. DNA Nanotubes: Construction and Characterization of Filaments. In *The 8th International Meeting on DNA Based Computers (DNA 8), Sapporo, Japan*, June 2002.

[24] Jie Liu, Andrew G. Rinzler, Hongjie Dai, Jason H. Hafner, R. Kelley Bradley, Peter J.

Boul, Adrian Lu, Terry Iverson, Konstantin Shelimov, Chad B. Huffman, Fernando Rodriguez-Macias, Young-Seok Shon, T. Randall Lee, Daniel T. Colbert, and Richard E. Smalley. Fullerene Pipes. *Science*, 280:1253–1256, 1998.

[25] S. R. Lustig, E. D. Boyes, R. H. French, T. D. Gierke, M. A. Harmer, P. B. Hietpas, A. Jagota, R. S. McLean, G. P. Mitchell, G. B. Onoa, and K. D. Sams. Lithographically Cut Single-walled Carbon Nanotubes: Controlling Length Distribution and Introducing End-group Functionality. *Nano Letters*, 3(8):1007–1012, August 2003.

[26] K. Nikolic, A. Sadek, and M. Forshaw. Fault-Tolerant Techniques for Nanocomputers. *Nanotechnology*, 13:357–362, 2002.

[27] J.H. Reif, T.H. LaBean, and N.C. Seeman. Challenges and Applications for Self-Assembled DNA Nanostructures. In A. Condon and G. Rozenberg, editors, *Proc. Sixth International Workshop on DNA-Based Computers, Leiden, The Netherlands. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelber*, volume 2054, June 2000.

[28] Michael D. Schroeder, Andrew D. Birrell, Michael Burrows, Hal Murray, Roger M. Needham, Thomas L. Rodeheffer, Edwin H. Satterthwaite, and Charles P. Thacker. Autonet: A High-speed, Self-Configuring Local Area Network Using Point to Point Links. *IEEE Journal on Selected Areas in Communications*, 9(8), October 1991.

[29] N.C. Seeman. DNA Engineering and its Application to Nanotechnology. *Trends in Biotech*, 17:437–443, 1999.

[30] M. S. Strano, C. A. Dyke, M. L. Usrey, P. W. Barone, M. J. Allen, H. W. Shan, C. Kittrell, R. H. Hauge, J. M. Tour, and R. E. Smalley. Electronic Structure Control of Single-walled Carbon Nanotube Functionalization. *Science*, 301:1519–1522, September 2003.

[31] S.J. Tans, A.R.M. Verschueren, and C. Dekker. Room-temperature Transistor Based on a Single Carbon Nanotube. *Nature*, 393:49–52, 1998.

[32] David L. Tennenhouse and David J. Wetherall. Towards an Active Network Architecture. *Computer Communication Review*, 26(2), 1996.

[33] K. van Berkel and A. Bink. Single-track Handshake Signaling with Application to Micropipelines and Handshake Circuits. In *Procceding of the Seconds International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 122–133, March 1996.

[34] S. J. Wind, J. Appenzeller, R. Martel, V. Derycke, and Ph. Avouris. Vertical Scaling of Carbon Nanotube Field-Effect Transistors using Top Gate Electrodes. *Applied Physics Letters*, 80:3817–3819, May 2002.

[35] E. Winfree, F. Liu, L. A. Wenzler, and N.C. Seeman. Design and Self-Assembly of Two-Dimensional DNA Crystals. *Nature*, 394:539, 1998.

[36] Hao Yan, Thomas H. LaBean, Liping Feng, and John H. Reif. Directed Nucleation Assembly of Barcode Patterned DNA Lattices. *Proceedings of the National Academy of Sciences*, 100(14):8103–8108, July 2003.

[37] Hao Yan, Sung Ha Park, Liping Feng, Gleb Finkelstein, John H. Reif, and Thomas H. LaBean. 4x4 DNA Tile and Lattices: Characterization, Self-Assembly, and Metallization of a Novel DNA Nanostructure Motif. In *Proceedings of the Ninth International Meeting on DNA Based Computers (DNA9)*, June 2003.

[38] Hao Yan, Sung Ha Park, Gleb Finkelstein, John H. Reif, and Thomas H. LaBean. DNA Templated Self-Assembly of Protein Arrays and Highly Conductive Nanowires. *Science*, September 2003.

[39] Ming Zheng, Anand Jagota, Ellen Semke, Bruce Diner, Robert Mclean, Steve Lustig, Raymond Richardson, and Nancy Tassi. DNA-Assisted Dispersion and Separation of Carbon Nanotubes. *Nature Materials*, 2:338–342, May 2003.

[40] Ming Zheng, Anand Jagota, Michael S. Strano, Adelina P. Santos, Paul Barone, S. Grace Chou, Bruce A. Dine, Mildred S. Dresselhaus, Robert S. Mclean, G. Bibiana Onoa, Georgii G. Samsonidze, Ellen D. Semke, Monica Usrey, and Dennis J. Walls. Structure-Based Carbon Nanotube Sorting by Sequence-Dependent DNA Assembly. *Science*, 302:1545–1548, November 2003.