Advances in Nanowire-Based Computing Architectures

Jun Wu, Sriram Venkateswaran and Minsu Choi
Missouri University of Science and Technology
United States

1. Introduction

The end of photolithography as the driver for Moore's Law is predicted within seven to twelve years and nanotechnologies are emerging that are expected to continue the technological revolution (Bourianoff et al., 2003). Recently, numerous nanoscale logic devices have been proposed based on nanoscale components such as CNTs and SiNWs; computing architectures are also being proposed using them as primitive building blocks.
One of the most promising nanotechnologies is the nanowire crossbar-based architecture, a two-dimensional array (i.e., nanoarray) formed by the intersection of two orthogonal sets of parallel and uniformly-spaced nanometer-sized wires (Dehon et al., 2003; Rueckes et al., 2000), such as carbon nanotubes (CNTs) and silicon nanowires (SiNWs). Experiments have shown that such wires can be aligned to construct an array with nanometer-scale spacing using a form of directed self-assembly and the formed crosspoints of nanoscale wires can be used as programmable diodes, memory cells or FETs (Field-Effect Transistors); therefore, nanoscale logic devices can be realized.
Nanowire crossbars offer both an opportunity and a challenge. The opportunity is to achieve ultra-high density which has never been achieved by photolithography (a density of 10^{11} crosspoints per cm^2 has been already achieved (Melosh et al., 2003)). The challenge is to make them simple enough to be manufactured and reliable enough to be used in everyday computing applications. Ultra-high density fabrication could potentially be very inexpensive if researchers can actualize a chemical self-assembly, but such a circuit would require laborious testing, repair and reconfiguration processes, implying significant overhead costs. All reconfigurable computing architectures based on nanowire crossbars are commonly envisioned to be used for synchronous circuits and systems. Thus, a complex clock distribution network should be fabricated along with nanowire crossbars and precise timing control should be practiced to avoid all timing-related faults induced by physical design parameter variations caused by nanoscale nondeterministic assembly.

2. Benefits and Challenges

Unlike the conventional clocked counterparts, a new asynchronous architecture for carbon nanotube (CNT) and silicon nanowire (SiNW)-based reconfigurable nanocomputing systems is proposed to address aforementioned issues in this chapter. The proposed

asynchronous nano-architecture is based on delay-insensitive data encoding and self-timed logic referred to as the Null Convention Logic (NCL) – making it totally clock-free. The potential benefits can be summarized as follows:
1. Manufacturability: Distributing the clock signal is not needed. All clocking-related hardware components can be removed from the overall hardware design. This not only saves on circuit area, but also facilitates the homogeneity of the system – an important issue for manufacturing based on molecular self-assembly.
2. Scalability: The overall circuit is self-timed – meaning that all timing information is embedded in the encoding. This unique aspect of the proposed asynchronous architecture provides better scalability than its clocked counterparts, since the timing is locally handled. The timing complexity remains the same even though the size of the circuit gets larger.
3. Robustness: Due to non-determinism of the directed self-assembly paradigm, nanowire crossbar circuits are anticipated to exhibit large variations in physical parameters. Since any physical variation in an electrical parameter may have its own negative effect on the timing behavior of the circuit, being able to design delay-insensitive circuits (i.e., correct operation of the circuit is independent of the timing) is a significant capability and it would greatly increase the robustness of the circuit to design parameter variations.
4. Modularity: Asynchronous circuits can be divided into modules that are designed without considering other modules, especially with respect to timing relationships. Modularity facilitates configurability of circuits.
5. Defect and Fault-Tolerance: In addition to the complete removal of many timing-related failure modes, testing complexity is reduced in that stuck-at-1 faults simply halt the circuit. Also, in case of dual-rail encoding, 11 is considered as an invalid code. So, any permanent or transient fault that results in this invalid codeword can be eventually detected. Only stuck-at-0 faults and some other transient faults need to be exercised with applied patterns. Design time and risk as well as circuit testing requirements are expected to be decreased because of elimination of complexity of clock and its associated critical timing issues.
In order to be a viable nanotechnology, the nanowire crossbar-base systems should be:
1. Structurally simple and scalable enough to be fabricated by bottom-up manufacturing technique.
2. Robust enough to tolerate extreme parametric variations.
3. Defect and fault-tolerant enough to overcome the extreme defect densities, aging factors and transient faults.
4. Able to support at-speed verification and reconfiguration.

3. Nanoscale Devices and Computing Architecture

3.1 CNT and SiNW Crossbar-Based Nanoscale Devices

Unlike CMOS, chemically-assembled nanoscale components (such as CNTs and SiNWs) are unlikely to be used to construct complex aperiodic structures (Goldstein et al,. 2001; Goldstein et al,. 2002; Mishra et al,. 2002). Thus, unconventional architectures are often desirable. One of the most promising computational nanotechnologies is the crossbar-based architecture (Kuekes et al,. 2000; Kuekes et al,. 2001; Chen et al,. 2003; Ziegler et al,. 2002; Stan et al,. 2001), a two-dimensional array (nanoarray) formed by the intersection of two orthogonal sets of parallel and uniformly-spaced nanometer-scale wires, such as carbon nanotubes (CNTs) and silicon nanowires (SiNWs). A typical nanoscale crossbar structure is shown in Fig.1. Experiments have shown that such nanoscale wires can be aligned to construct an array with nanometer-scale spacing using a form of directed self-assembly (Nicewarner et al,. 2002; Huang et al,. 2001; Cui et al,. 2001; Mirkin et al,. 2000;)

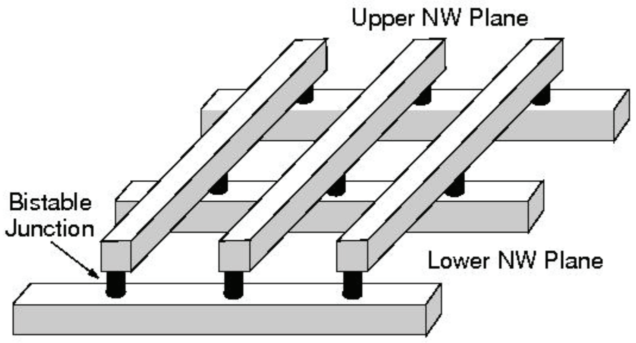


Fig. 1. Typical nanowire crossbar architecture

One or more crosspoints can be grouped together to form a memory or logic device. Lieber research group have shown electro-mechanical switching devices using suspended nanotubes (Rueckes et al,. 2000). The NT-NT junction is bistable with an energy barrier between the two states. SiNWs can be substituted for the lower wire, and these junctions can rectifying such that the connected state exhibits p-n-diode rectification behavior. Thus, diode-resistor logic can be realized in nanoscale. Doped SiNWs exhibit FET (Field Effect Transistor) behavior (Huang et al,. 2001). That is, oxide can be grown over the SiNW to prevent direct electrical contact of a crossed conductor. The electrical field of one wire can then be used to gate the other wire by locally evacuating a region of the doped SiNW of carriers to prevent conduction. In other words, as the gate voltage is changed, conductance increases or decreases between the source and drain. CNTs also demonstrate FET behavior (Bachtold et al,. 2001; Derycke et al,. 2001; Soh et al,. 1999) and can be also used as memory devices (Finkelstein et al,. 2003; Rueckes et al,. 2000) or interconnects (Dekker et al,. 1999).
Molecular resonant tunneling diodes, often called negative differential resistors (NDRs), have also been realized (Bhattacharya et al,. 2001; Chen,J. et al,. 1999; Chen ,J. Et al,. 2000; Yu et al,. 1999). Devices with useful peak-to-valley ratio have been measured at room temperature (Chen,J. et al,. 2000). These molecules can be used as the basis of logic families (Chen,J. et al,. 1999) or as the core of a molecular latch which also provides signal restoration and I/O isolation (Goldstein et al., 2002).

3.2 Nanoscale Universal Computing Architecture

DeHon et. al. h shown how to organize the CNTs, SiNWs and molecular-scale devices that are now being developed into an operational reconfigurable computing system (referred to as the NanoPLA) (Dehon et al,. 2003; Naeimi & Dehon et al. 2004). The molecular-scale wires can be arranged into interconnected, crossed arrays with switching devices at their crosspoints. Similar nanoscale reconfigurable homogeneous architecture, namely Nano-Fabric, has been also proposed by Goldstein et. al. ((Goldstein et al,. 2002). Lieber research group referred to such a nanoscale reconfigurable homogeneous structural paradigm for computing as the Universal Computing Architecture (UCA). Such nanoscale computer architectures share common characteristics - they support unconventional nanoscale manufacturing paradigm via simple homogeneous periodic structures and reconfigurability for post-fabrication design mapping. In general, nanowire-based reconfigurable systems based on the UCA concept are referred to as the **Nanowire Reconfigurable Crossbar Architecture (NRCA)**.

There are considerable on-going research and development efforts to fabricated nanoscale crossbar-based circuits and systems using advanced lithography as well. For example, researchers at HP successfully fabricated 8*8 (i.e., 64 bits) crossbar memory arrays (Paulson et al,. 2005; Chen, Y et al,. 2003) using nano-imprint lithography (Mcalpine et al,. 2005; Hiroshima et al,. 2002; Xia et al,. 1999). Non-volatile bistable Rotaxane molecules are sandwiched between two orthogonal metal wires to form a non-volatile memory cell.

About 75% of the memory cells are tested functional and all the other cells are either stuck-open or stuck-closed: therefore, non functional. It is also easily predictable that even higher defect densities will be induced as the size of such devices scale down. Therefore, enhancement of lithographic resolution and defect tolerance are two key challenges that advanced lithographic fabrication methods, such as nano-imprint lithography, face.

3.3 Bottom-Up Paradigm for Nanowire Crossbar Assembly

CNTs and SiNWs are the most promising building blocks for nanoscale computing systems. Unfortunately, synthesis of such nanowires and high-density integration of devices and systems based on nanowires are fully different from conventional top-down lithographic fabrication techniques, because such nanowires must be synthesized first, then assembled into functional devices and systems in a bottom-up manner.

Langmuir-Blodgett (LB) method (Huang et al,. 2001) is an effective way to hierarchically assemble 1D nanostructures into integrated nanosystems based on fluidic flow. In this method, SiNWs or CNTs can be aligned by passing a suspension of NWs through microfluidic channel structures. Ordered monolayers are formed over a large area and transferred to substrates by the LB method. It has been demonstrated that virtually all of the NWs are aligned along the flow direction. Alternating the flow in orthogonal directions in a two-phase assembly process results in crossbar structures having high yield.

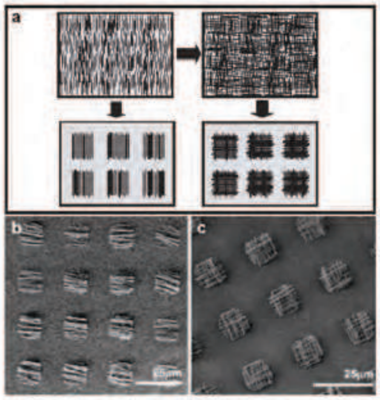


Fig. 2. Hierarchical patterning by fluidic flow and LB method (Whang et al. 2004)

The aligned, controlled spacing NW structures produced by the LB assembly method exhibit fluctuations in the average alignment direction and poor end-to-end registry. So, interconnected finite-size arrays of nanoscale devices are more desirable than monolithic structures for integrated nanosystems, because hierarchical organization reduces the probability that small numbers of defects will cause catastrophic failure in the whole system. Hence, by adjusting the array size to be less than the average NW length it is possible to minimize the number of NWs that fail to span the width of an array due to poor end-to-end registry. Following uniform transfer of NWs of a specified spacing onto a substrate, photolithography is used to define a pattern over the entire substrate surface, which sets the array dimensions and array pitch, and then the NWs outside the patterned array are removed by gentle sonication. SEM (Scanning Electron Microscope) images of both parallel NW arrays and crossed NW arrays are shown in Fig.2.((b) and (c)), respectively.

3.4 Overview of Asynchronous Design Methods - Null Convention Logic for Clock-Less Computing

Asynchronous circuits fall into two main categories: delay-insensitive and bounded-delay models. Paradigms, like NCL (Null Convention Logic), assume delays in both logic elements and interconnect to be unbounded, although they assume that wire forks are isochronic (Fant et al,. 1996). This implies the ability to operate in the presence of indefinite arrival times for the reception of inputs. Completion detection of the output signals allow for handshaking to control input wavefronts. On the other hand, bounded-delay models such as Huffman circuits (Unger, 1969), burst-mode circuits (Nowick & Dill, 1991), and micropipelines (Sutherland, 1989) assume that delays in both gates and wires are bounded. Delays are added based on worse-case scenarios to avoid hazard conditions. This leads to extensive timing analysis of worse-case behaviour to ensure correct circuit operation. Since NCL exhibits neither of these characteristics, it is well-suited for the proposed clock-free nanowire crossbar architecture and bounded delay models are not addressed further.

Null Convention Logic (NCL) is a delay-insensitive logic which is one of the Asynchronous circuits categories. NCL circuits utilize dual-rail or quad-rail logic to achieve this delay-insensitivity (Bandapati et al,. 2003). In dual-rail logic, a signal D consists of two wires D^0 and D^1; which may assume any value from the set DATA 0, DATA 1, and NULL. The DATA 0 state ($D^0 = 1$, $D^1 = 0$) corresponds to a Boolean logic 0, the DATA 1 state ($D^0 =0$, $D^1 = 1$) corresponds to a Boolean logic 1, and the NULL state ($D^0 = 0$, $D^1 = 0$) corresponds to the empty set meaning that the value of D is not yet available. The two rails are mutually exclusive, so that both rails can never be asserted simultaneously; this state is defined as an illegal state (Bandapati et al,. 2003).

NCL is self-timed logic paradigm in which control is inherent. Whenever the circuit inputs become available, it does not need timing analysis for correct operation. NCL circuits switch between DATA (data representation) and NULL (control representation). According to the Seitz's weak switching conditions: outputs transfer from DATA to NULL only until all inputs of a combinational circuit transition from DATA to NULL; likewise, outputs transfer from NULL to DATA only until all inputs of a combinational circuit transition from NULL to DATA; most delay insensitive systems (Singh et al,. 2002) have at least two register stages, including NCL (Smith, 2006). We call them NCL registers; they are connected by request and acknowledge handshaking signals (Ki and Ko). NCL uses symbolic completeness of expression to achieve self-timed behaviour. A symbolically complete expression is defined as an expression that only depends on the relationships of the symbols present in the expression without a reference to the time of evaluation. Traditional Boolean logic is not symbolically complete; the output of a Boolean gate is only valid when referenced with time. NCL uses threshold gates with hysteresis for its composable logic elements. There are 27 threshold gates in all, as Table 1 shows. One type of threshold gate is the THmn gate, where 1 ≤m < n. A THmn gate corresponds to an operator with at least m signals asserted as its set condition and all signals de-asserted as its reset condition. THmn gates have n inputs. At least m of the n inputs must be asserted before the output will become asserted. Because threshold gates are designed with hysteresis, all asserted inputs must be de-asserted before the output will be de-asserted. Hysteresis is used to provide a means for monotonic transitions and a complete transition of multi-rail inputs back to a NULL state before asserting the output associated with the next wavefront of input data. Fig. 3 shows the basic symbol of THmn gate and TH34w2 gate.

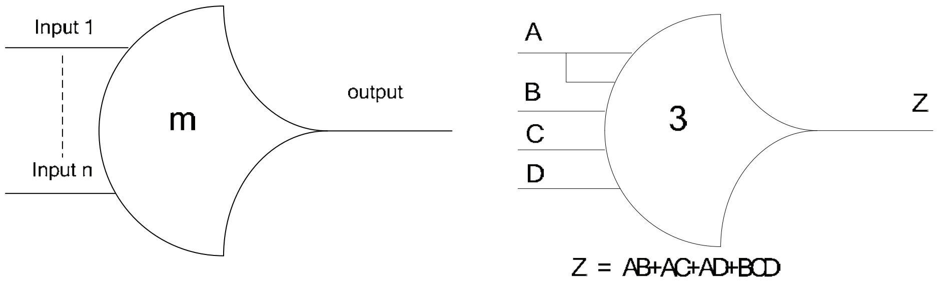


Fig. 3. THmn gate symbol

Another type of threshold gate is referred to as a weighted threshold gate. Weighted threshold gates are denoted as THmnWw_1..w_R. Weighted threshold gates have an integer value, m = w_R > 1, applied to inputR4. Here 1 = R < n; where n is the number of inputs; m is the gate's threshold; and w_1, w_2,...w_R, are the integer weights of input1, input2, ...inputR, respectively. For example, consider a TH34W2 gate, whose n = 4 inputs are labeled A, B, C, and D. The weight of input A, W(A), is therefore 2. Since the gate's threshold, m, is 3, implies that in order for the output to be asserted, either inputs B, C, and D must all be asserted, or input A must be asserted and any other input, B, C, or D must also be asserted. NCL threshold gates may also include a reset input to initialize the gate's output.

NCL Gate	Function
TH12	A + B
TH22	AB
TH13	A + B + C
TH23	AB + AC + BC
TH33	ABC
TH23w2	A + BC
TH33w2	AB + AC
TH14	A + B + C + D
TH24	AB + AC + AD + BC + BD + CD
TH34	ABC + ABD + ACD + BCD
TH44	ABCD
TH24w2	A + BC + BD + CD
TH34w2	AB + AC + AD + BCD
TH44w2	ABC + ABD + ACD
TH34w3	A + BCD
TH44w3	AB + AC + AD
TH24w22	A + B + CD
TH34w22	AB + AC + AD + BC + BD
TH44w22	AB + ACD + BCD
TH54w22	ABC + ABD
TH34w32	A + BC + BD
TH54w32	AB + ACD
TH44w322	AB + AC + AD + BC
TH54w322	AB + AC + BCD
THxor0	AB + CD
THand0	AB + BC + AD
TH24comp	AC + BC + AD + BD

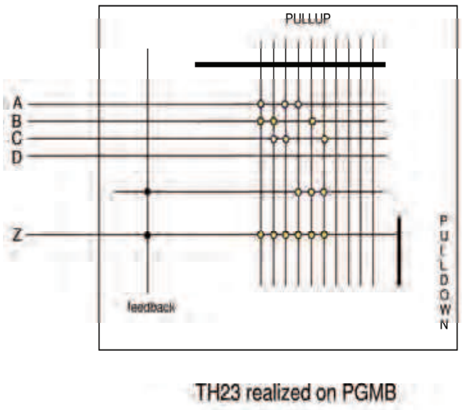
Table 1. 27 Fundamental NCL Gates (Smith, 2006).

By employing threshold gates for each logic rail, NCL is able to determine the output status without referencing time. NCL switches monotonically. All combinational logic clouds switch from a condition of all NULL to a condition where some of the wires are at DATA. Once a wire has switched to DATA, it remains there until the combinational logic cloud is returned to NULL state. This monotonic switching behaviour makes NCL designs glitch-free. Because every threshold element in NCL exhibits inherent storage behaviour due to hysteresis, storage cells such as latches or flip-flops are unnecessary. Data can be pipelined by creating registration elements from the gate primitives. Using these same registration elements, data can be fed back on itself to create sequencers in much the same way sequencers are built in traditional logic. Data Output: The proper conditions have been met to provide DATA at the output of the registration element.

4. Proposed Architecture-Clock Free Nanowire Crossbar Architecture

4.1 Programmable Gate Macro Block (PGMB)

The primitive unit of the proposed architecture is referred to as the Programmable Gate Macro Block (PGMB). Each block is made of one AND-plane and one OR-plane formed by the diode crossbars. Vertical nanowires with pull-up resistors form product terms and horizontal wires with pull-down resistor add them using OR-logic. So, each PGMB can be programmed to embed a NCL gate macro function in SOP (Sum-Of-Product) form. It also has a feedback loop which drives the output back to an input wire. The maximum number of inputs to any threshold gate is 4. Along with this, it needs a feedback to implement any of the 27 threshold gate macros. Notably, there is no need to add complemented primary inputs as done in NanoFabric (Goldstein et al., 2001), since inversion can be easily done in NCL by crossing a dual-rail signal, D, consists of two wires, D^0 and D^1. Therefore, there is no need to use FET-based (Field Effect Transistor) inverters as done in NanoPLA (Dehon et al,. 2003). This unique encoding-level inversion capability is another significant merit of the proposed architecture. Fig.4 shows the programming of TH23 and TH34W2 gates on PGMB where small dots represent activated programmable crosspoints. The output of the TH23 gate is given by the logic Z = AB + BC + CA + (A+B+C)Z', where Z' is the previous output of the TH23 gate which is fed back to the input nanowire. Likewise, the output of the TH34W2 gate is given by the logic Z = AB + AC + AD + BCD + (A + B + C + D)Z'. Notably, these two gates embedded in PGMB are core building blocks in NCL full adder design, which will be discussed in the next subsection.



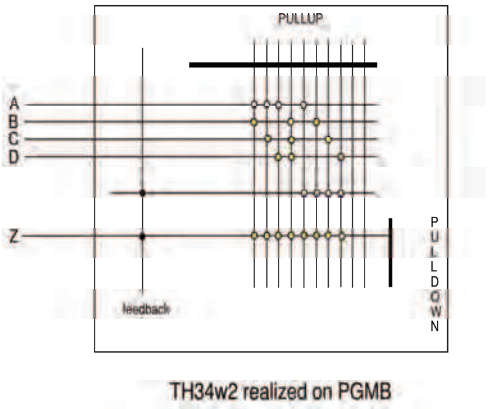


Fig. 4. TH23 and TH34w2 gates configured on PGMB. All 27 gate macros can be programmed on PGMB and additional row nanowires and column nanowires can be added as redundancy in case of high defect rate.

4.2 Addressing Decoding strategy

Snider et. al. (Snider et al., 2005) has show that the demultiplexers are expected to be the key components in interfacing submicrometer-scale and nano-scale electronic circuit. Demultiplexers are used for interfacing nanoelectronic circuits and outside conventional CMOS (Kuekes et al, 2001). We can through a small number of micro-scale wires to control a large number of nano-scale wires by forcing a 'selected' or 'unselected' voltage onto nanowires (Snider et al,. 2005). In our proposed clock-free nanowire crossbar architecture, we use it to control programmable junction(crosspoints) of different PGMBs. Fig.5 shows the implementation of a demultiplexer forming by a diode crossbar and resistors. The address lines of the demultiplexer are micro scale wires which could be implemented on a CMOS substrate, and the output lines are nanowires (Snider et al,. 2005). The relationship between the number of input microwires(M) and the number of output nanowires(N) is $N=2^M$; If we have eight outputs which need at least three inputs for constituting a demultiplexer. For example, each PGMB's size is 6*10, for covering all crosspoints which can be programed, demultiplexers located on rows and columns are needed respectively. See Fig.6.

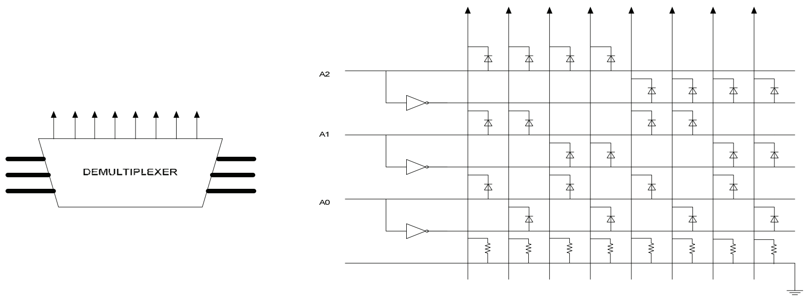


Fig. 5. A demultiplexer consist of diode crossbar and resisors,input address is A2,A1,A0 (Snider et al., 2005).

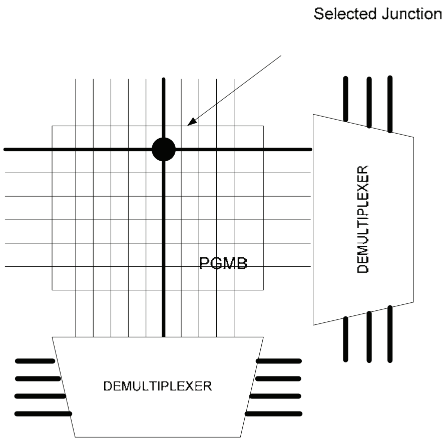


Fig. 6. Two demultiplexers control one PGMB for selecting different crosspoints.

4.3 Configurable Logic Block(CLB)

For the purpose of reconfiguring all of the 27 threshold gates arbitrarily, we need to design a reconfigurable architecture to make it work no matter what kind of logic those gates possess. We designed three different Configurable Logic Blocks (CLB) and named them CLB-1, CLB-2, CLB-3. Fig.7 shows that each CLB consists of four PGMBs which are surrounded by nanowires, and demultiplexers. Horizontal nanowires cross over vertical nanowires forming the crosspoints. In order to utilize the programmability of crosspoints for configuring logic functions, we use different demultiplexers to select them. Output signals from the top demultiplexer determines which nanowires are selected as the input signals of the PGMB. The demultiplexer along adjacent edges of nanowire/nanowire crossbar is used for selecting different PGMBs to receive the input signals. We program each crosspoint through controlling the up-side and right-side demultiplexer at the same time. For example, we program the crosspoints to ON state or OFF state by driving the up-side demultiplexer's output nanowires with positive voltage and driving the right-side demultiplexer's output with negative voltage. In this case we assume those unselected outputs are driven with ground, then the intersection of driven nanowires have a voltage drop which is different from other unselected crosspoints in the crossbar. This allows us to configure crosspoints sensitive to voltage drops across them (Snider et al,. 2005)..

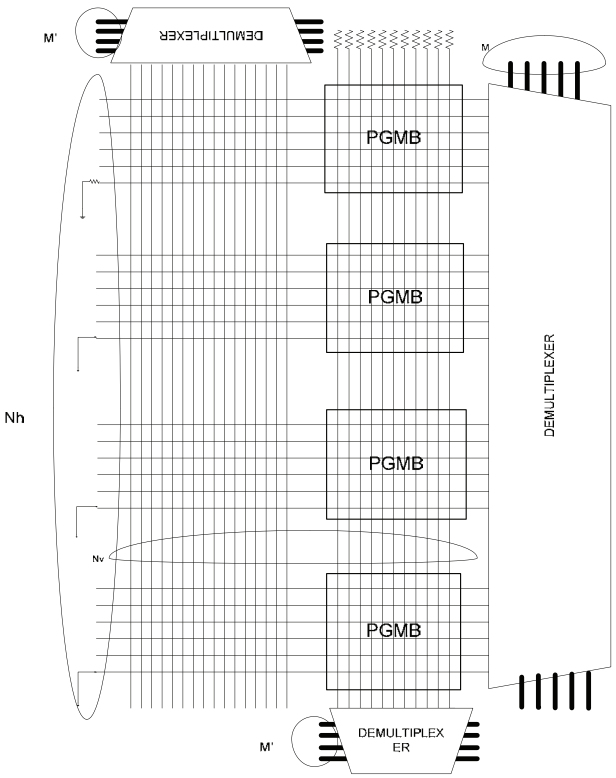


Fig. 7. The first version of Configurable Logic Block (CLB-1).

Fig.8 shows the other two versions of CLB. The main difference between them is the number of demultiplexers. For CLB-1, there is only one demultiplexer on the right-side, so only one crosspoint can be selected at one time. For CLB-2, there are two demultiplexers on the right side, so two crosspoints can be selected at the same time, however, the location of the two crosspoints have to be in the different group of PGMBs(assume the above two PGMBs are group1, another two PGMBs are group2). For CLB-3, there are four demultiplexers on the right side, so four crosspoints can be selected at the same time, the precondition is four crosspoints from the four different PGMBs respectively. Compare above mentioned three types of CLB architecture, it is possible to see the advantage of those who has less demultiplexers is easier to control, but it need more time to access crosspoints. We will analysis and compare the detail data of different CLBs in Mathematical Analysis section.

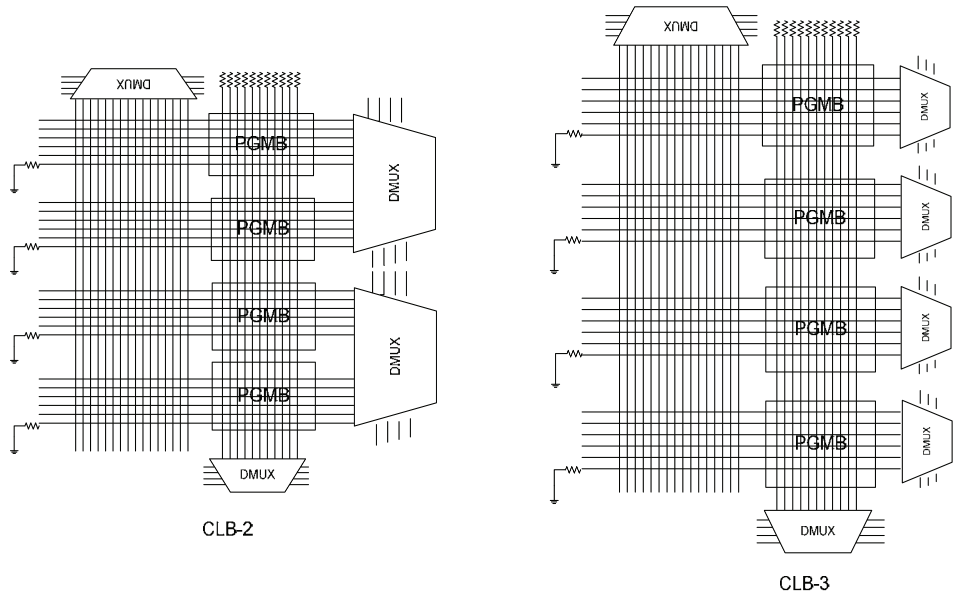


Fig. 8. The second and third version of Configurable Logic Block(CLB-2 and CLB-3).

4.4 Case Study: Asynchronous Full Adder

A full adder can be implemented by using threshold gates as shown in Fig.9.

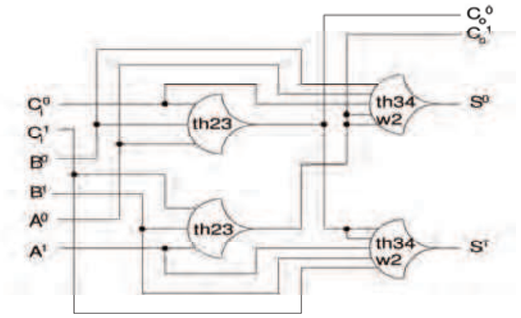


Fig. 9. 1-bit full adder implement by threshold gates (Smith, 2006).

As the figure shows, it consists of two TH23 gates and two TH34w2 gates. This requires three inputs which are X, Y for addition and Ci for carry in, two outputs are summation S and carry-out Co. These bits are represented by X^0, X^1, Y^0, Y^1, Ci^0, Ci^1, Co^0, Co^1, S^0, S^1 encoded in dual rail logic respectively; then we use our proposed architecture to implement 1-bit full adder.

As mentioned above, crosspoints can be programmed by demultiplexers, and the input signals can be routed to the corresponding PGMBs. Co^0 and Co^1 are the output signals of

TH23 gates; they are also the input signals of TH34w2 gates. Fig.10 shows these two signals are propagated to the TH34w2 PGMBs and located in the free nanowires.

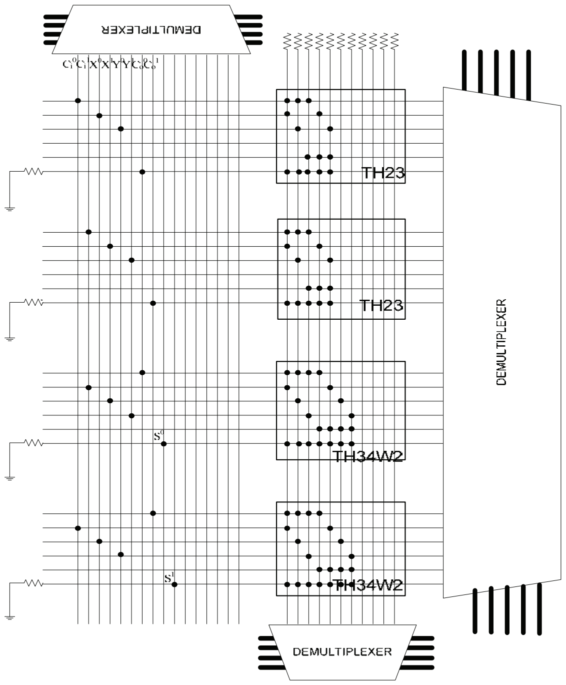


Fig. 10. 1-bit full adder implement by CLB-1.

4.5 NCL Register

NCL Register stage consists of three threshold gates. Fig.11 shows how the 1-bit NCL register is implemented by threshold gates. There are two TH22 gates are used to generate a handshaking signal which helps in synchronizing the circuit. There are two signals: 'REQUEST FOR DATA' and 'REQUEST FOR NULL' generated by the registers that are passed on the previous register. Ki (input of successive stage) and Ko (output of previous stage/input of next stage) are the handshaking signals, I^0, I^1, O^0, O^1 are input and output data rails respectively.

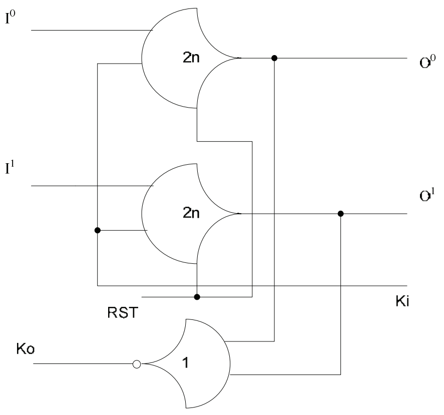


Fig. 11. 1-bit NCL register implement by threshold gates.

Fig.12 shows the 1-bit NCL register is implemented by our proposed NRCA. As the same routing method of 1-bit full adder, we route required signals into different crosspoints on the crossbar, and get the logic output. Since Ko is the invert value of output, the inversion can be easily done in NCL by crossing a dual-rail signal. For example, when the output is Boolean logic 0 ($Ko^0 = 1$, $Ko^1 = 0$), after crossing the dual-rail signal, it becomes ($Ko^0 = 0$, $Ko^1 = 1$), which corresponds the Boolean logic 1, output data successfully be inverted.

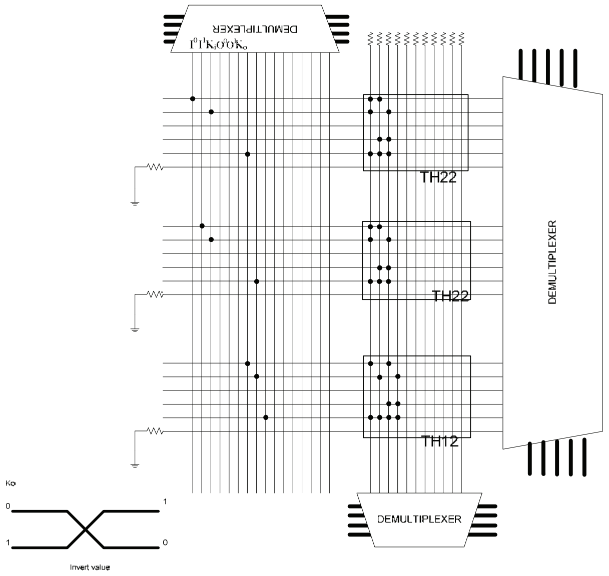


Fig. 12. 1-bit NCL register implement by CLB-1.

4.6 Connection between Combination Logic and NCL Registers

As NULL Convention Logic for Clock-Less computing section mentioned, we need handshaking signals to connect two delay-insensitive NCL registers with delay-insensitive combination logic circuit (Fig.13).

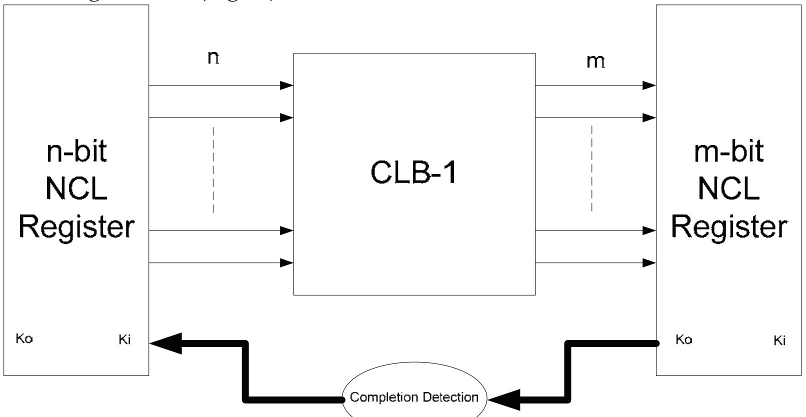


Fig. 13. NCL combinational logic block with input and output registrations.

The case of Asynchronous Full Adder can be used as an example to explain the details (Fig.14). For a 1-bit full adder, there are three inputs X^0, X^1, Y^0, Y^1, Ci^0, Ci^1 encoded in dual rail logic respectively, 3-bit NCL register is needed; also because the full adder has two outputs Co^0, Co^1, S^0, S^1 encoded in dual rail logic respectively, so another 2-bit NCL register is needed. Assume the initial value of Ki is zero, and then we get all zero for the outputs from 3-bit NCL register, now the system is on NULL state; at the same time, 2-bit NCL register which accepts outputs from full adder are all zero. Therefore the feedback handshaking signal Ko becomes 1. After feedback it to the previous 3-bit NCL register as an updated Ki, the system state makes a transition to the DATA state, finally we can get the correct output result of Full Adder. The input registration alternates NULL and DATA waves and those waves are initiated by the output registration's feedback signal.

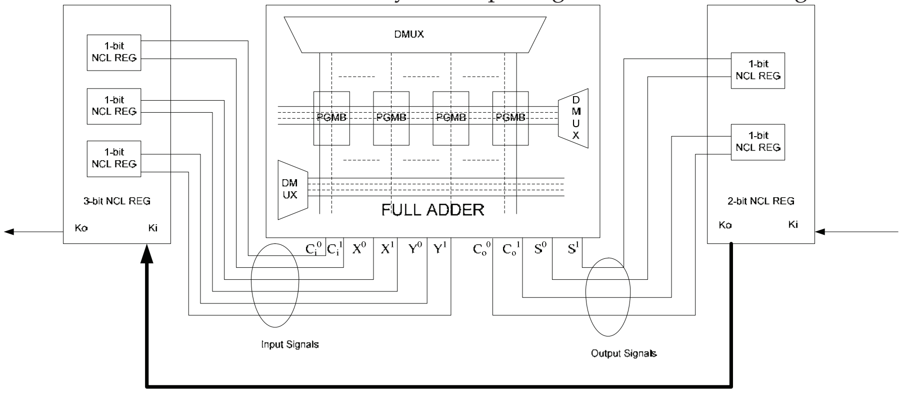


Fig. 14. Full Adder implementation with input and output registrations.

5. Hierarchical Architecture

Hierarchical interconnection architecture has been widely chosen by FPGA designs due to the easiness of programming and flexible routing. The proposed asynchronous NRCA is also based on hierarchical structure to interconnect PGMBs. The first level is Configurable Logic Blocks (CLB), which consists of 4 PGMBs, which is level 1. Level 2 consists of four CLBs using the same principle. Level 3 consists of four 4*CLBs and level 4 consists of four 16*CLBs and so on. The number of levels depends on what kind of logic circuit are processing. Level 1 has been introduced in the previous section. Hierarchical structures used at higher levels are described in this section.

5.1 Level 2 – 4*CLB

Each CLB has 16 distinct inputs and 4 outputs, the architecture shows on Fig.15. Interconnect switching block is used for interconnecting the input/output signals to the different blocks or up to the next level. A 4-bit NCL register would be a good design example for explaining this architecture. Fig.16 shows how to implement the logic circuit by using 4*CLB architecture. There are D^0~D^3 and Q^0~Q^3 are input and output data rails respectively, Ki and Ko are handshaking signals.

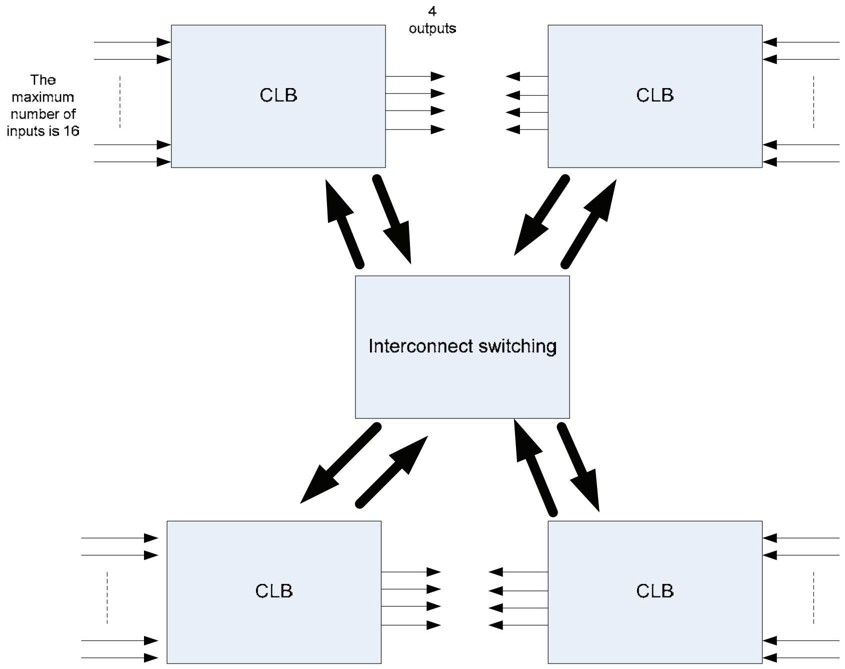


Fig. 15. level-2:4*CLB architecture.

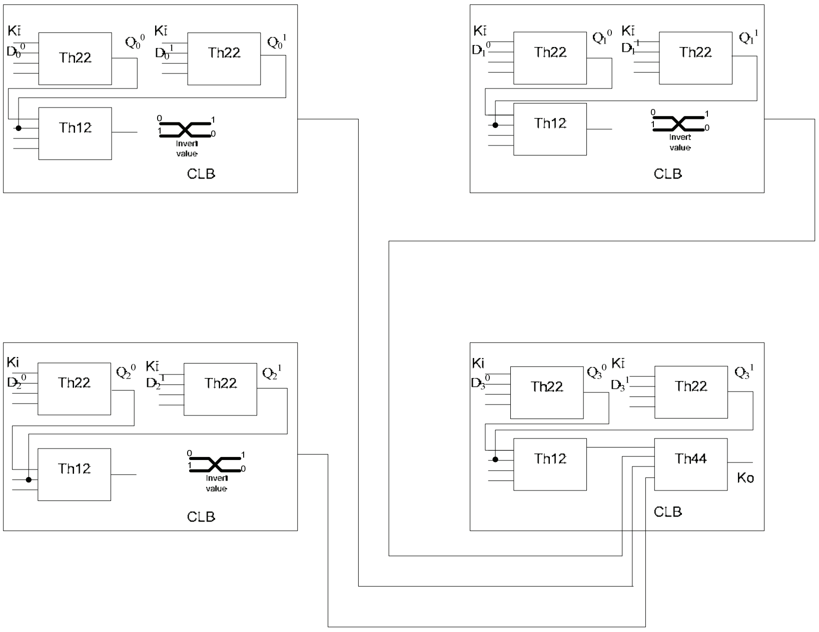


Fig. 16. 4-bit NCL register implemented at level-2 structure.

5.2 Hierarchical Interconnect Switching

The design of Hierarchical Interconnect Switching is one of key elements for the whole system, there are several programmable interconnect strategies used by various designers, one of very popular interconnect technique is pass transistor interconnect. We use this technique for our proposed architecture so that the interconnect switching system can be controlled flexibility through different arrangement of pass transistors. It constitutes by transistors and latches, the latch is used for storing the gate input of the transistor, we can also use SRAM to store the value for the switch input, and it can be programmed during the configuration phase (Singh et al., 2002).

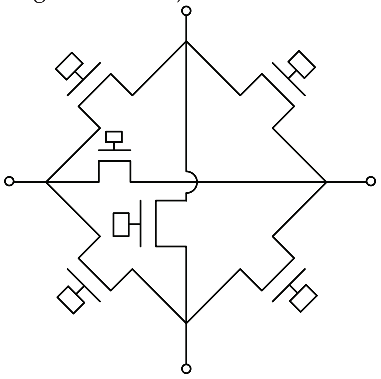


Fig. 17. Six Pass Transistors Interconnects (Lee et al., 1999).

Fig. 17 shows the different arrangements of various interconnect schemes using pass transistors. This scheme contains 6 transistors and for each interconnect point, it would has 3 different directions to be chosen. According to this property, we can build up our own interconnect switching matrix for routing a set of inputs to a set of outputs (Fig.18).

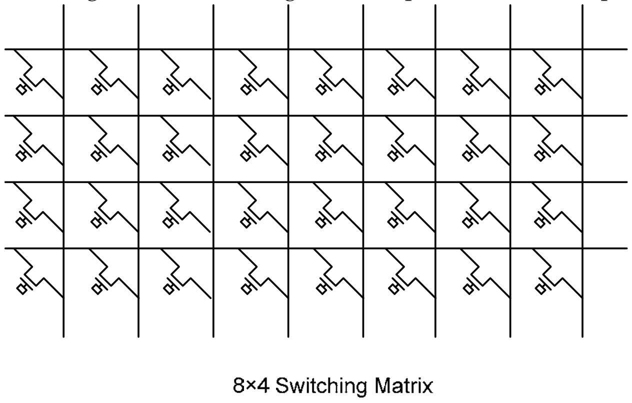


Fig. 18. 8*4 Switching Matrix

6. Parameters

Cost of the final manufactured product is become more and more important to a digital electronics. There are numerous factors such as area, manufacturability and so on would influence the cost. So analyzing them from mathematical angle is necessary. To show the design trade-offs of the three CLBs design, two difference benchmarks will be used; namely area performance and execution complexities. Section VI models the area, execution steps and timing issues of these structures.

6.1 Area

The following feature size parameters are used in this chapter:

- N_v: The number of nanowires of vertical.
- N_h: The number of nanowires of horizontal.
- M : The number of microwires of demultiplexers.
- N_d: The number of nanowires which outputs from demultiplexers.
- P_n: The nanowire pitch for nanowires which are inputs to diodes.
- P_m: The microwire pitch for microwires.
- D_n: Diameter(width) of each nanowires.
- D_m: Diameter(width) of each microwires.
- A_d: The area of each demultiplexers.

For the 45nm node, the lithographic interconnect pitch is 105nm [ITRS 2001]. Technology developments suggest we can build and assemble 10nm pitch nanowires with crosspoints at every nanowire-nanowire crossing (Dehon et al., 2003). So P_n = 10nm, P_m = 105nm. And nanowires with diameters down to 3nm have been demonstrated (Cui et al. 2001).
Compare the area from different design of CLBs from Fig.7.and Fig.8., we can see the basic area composition of each tile.

$$\text{Width} = Pn(Nv-1) + Pm(M-1) \quad (1)$$

$$\text{Height: } Pn(Nh-1) + Pm(M'-1)*2 \quad (2)$$

$$\text{Area} = \text{Width} * \text{Height} \quad (3)$$

The only difference between CLB-1,CLB-2,CLB-3 is the number of demultiplexers. So "M" (number of microwires of demultiplexers on the right side) is different fro m each other (M=5(CLB-1); M=4(CLB-2); M=3(CLB-3)). Fig.19 shows the area comp arison of different design of CLBs, the CLB-3 is the smallest since the four demultip lexers can be located on the same plate by using the assembly techniques.

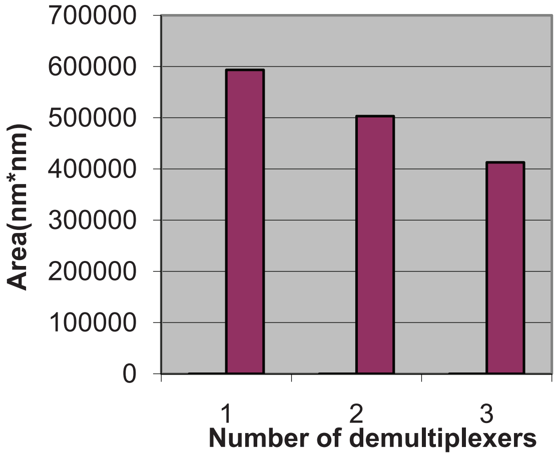


Fig. 19. Area comparison of three different designs of CLBs

6.2 Execute steps of each CLB

For improving the design, we not only take into account the area, we also consider about the complexity of the operation. One of factors is the number of steps for processing a logic circuit. Because it is directly impact the efficiency of design. The relationship between numbers of crosspoints needed to be programmed and the average steps for processing all of them is shown in Table 2. It shows that as increasing of number of points, the average steps are also increasing, but for different CLBs, the average steps are also different.

Numbers of Points	Average Steps		
	CLB-1	CLB-2	CLB-3
2	2	1.947	1.947
4	4	3.684	3.555
6	6	5.2105	4.926
8	8	6.5263	6.0625
10	10	7.6316	7.0013

Table 2. Execution steps comparison of different designs of CLBs

CLB-2 and CLB-3 have 2 and 4 demultiplexers on the right side respectively. We distinguish the first two PGMBs as a group, and the last two PGMBs as another group. For the CLB-2; as Fig.8 shows, it has capability of programming 2 points at the same time, only when the 2 points are distributed to the 2 different groups respectively, because each multiplexer controls 2 PGMBs at one time. Applying the same principle, the CLB-3 has the probability of programming 4 points at the same time, and the precondition is that 2 points have to be distributed to each PGMBs respectively because each demultiplexer can control only one PGMB.

7. Testing and Defect & Fault – Tolerance

Defects and faults are the worst problems with the nanoscale integration. Conventional fault-tolerant and reliable design techniques are primarily optimized for highly stable process technologies where reliability characterization of chip materials and circuit operation are well understood. However, conventional methods will become inadequate in the near future due to increased defect and failure rates, especially in case of nanoscale integration. New research in multi-scale fault-tolerance and reliability techniques is, therefore, critically needed for commercial adoption of emergent nanotechnologies (Heath et al., 1998). It is anticipated that NRCA will have significantly higher defect density than CMOS, as high as 10% (Huang et al., 2004; Jacome et al., 2004). High-density fabrication could potentially be very inexpensive if researchers can actualize a chemical self-assembly, but such a circuit would require laborious testing, diagnosis, repair and reconfiguration processes, implying a significant overhead cost, as well. Crossbar arrays and supporting nanoscale interfaces are also susceptible to transient faults that are usually caused by interferences such as crosstalk and radiation. Thus, reliability of NRCA is also an issue as the size of devices shrink and the integration density increases.

7.1 Defect-Unaware Testing and Reconfiguration

For the proposed clock-free nanowire crossbar architecture, we will take a different direction to overcome its defects. When we embed a large-scale system on the proposed clockless nanowire crossbar system, it is certainly time-consuming and impractical to scan through all programmable crosspoints of the given nanowire crossbars to locate defects. Instead, the given physical system design can be placed and routed without being aware of defects. Then, each PGMB can be functionally tested to significantly reduce the test space.

After testing, PGMBs and switching resources affected by those defects (i.e., stuck-OPEN crosspoints) can be further reconfigured to tolerate them. There are three possible ways to tolerate defects in PGMB. The first way is to reconfigure the order of primary input variables by reconfiguring the incoming switching block. In case of TH23, there are 3!=6 different ways to rearrange the order of inputs. Any order that can be used to avoid defective crosspoint(s) in the first three rows can be selected. The second way is to rearrange the order of columns of the given PGMB. Since all threshold gate macros have a simple Sum-of-Product (SoP) structure, by the commutative law, the order of these product terms can be rearranged without affecting the functionality of the gate. The third way is to include redundant rows and columns in each PGMB. Any row or column with an excessive number of defects that cannot be tolerated by the first two methods can be simply purged out by replacing it with a spare wire in this case. In case the PGMB under consideration is too defective to repair, it should be discarded and another PGMB should be used to realize the given threshold gate function.

7.2 Defect - Aware Approach

Defect Aware Approach needs to generate a defect map for the given PGMB, and compare the pattern of defect free crosspoints with the required pattern. This approach makes use of the available redundant crosspoints to instead of the defective crosspoints on each PGMB. According to the architecture of PGMB, columns are used for the AND plane, it can be interchanged without affecting functionality, that means for each threshold gate, it can be represented in several different ways without affecting the logic functionality. The defect rate of this approach is lower than Defect Unaware Approach, but the drawback being it needs more time for testing and reconfiguration because each PGMB has to be tested to locate all defective crosspoints, and make sure avoid them when the netlist is actually placed and routed.
The proposed clock-free architecture uses NCL encoding. Stuck-at-1 faults simply halt the circuit, since the NCL circuit cannot make a transition from DATA (either 01 or 10) to NULL (00) - therefore, can be easily screened out. Also, in case of dual-rail encoding, 11 is considered as an invalid code. So, any permanent or transient fault that results in this invalid codeword can be eventually detected as faulty. Only stuck-at-0 faults and some other transient faults need to be exercised with applied patterns.

7.3 Parametric Simulation Results

The illustration of the behaviour of programmability with varying defect rates for both the techniques is illustrated in Fig.20. The defect-aware approach surely outperforms the defect-unaware approach, but the defect-unaware approach still shows good programmability when the defect rate is considerably lower.

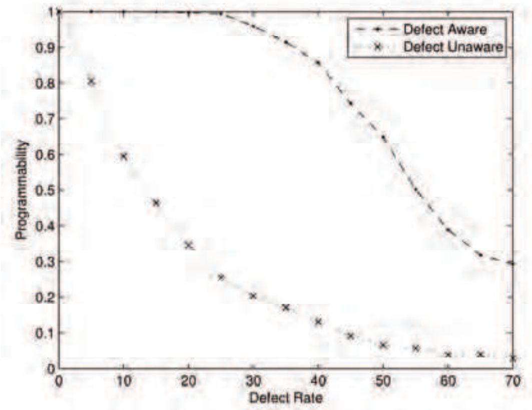


Fig. 20. Programmability comparison for defect-aware and -unaware approaches

It also shows almost all the gates are programmed even at 30% defect rate and then programmability (ratio of successfully programmed PGMB's to the total number of PGMB's) reduces, which is inevitable. This is much better in terms of programmability as compared to the defect aware technique but it takes time to generate defect map and then perform mapping and placement, but is very good at utilizing the inherent redundancy. This helps when we have a highly defective fabrication process; the algorithm bypasses the defective crosspoints and places the crosspoints without affecting the functionality of the gate. For both the techniques 20% of the rows have been dedicated to the OR crossbar logic. When the defect rate is considerably lower than 5%, the defect-unaware approach still shows good programmability.

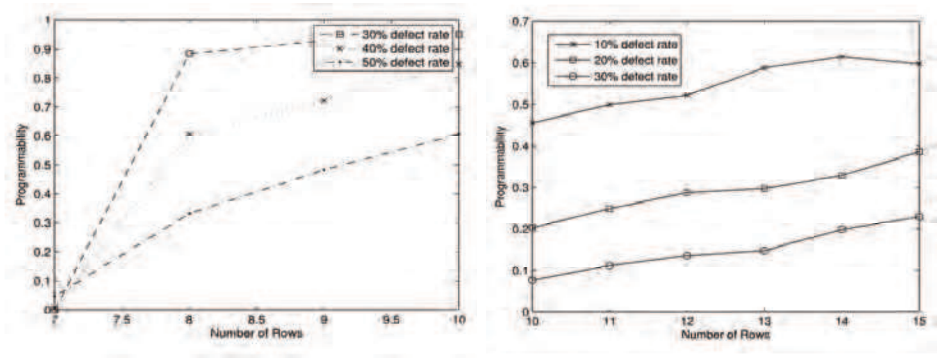


Fig. 21. Programmability VS Number of rows for defect-unaware and defect-aware.

Figure 21. provides useful analysis for the behaviour of programmability with change in dimension of the PGMB at different defect rates. These graphs also indicate the superiority of defect aware approach over the unaware approach.

7.4 Functional Test Algorithm (FTA)

The proposed functional test algorithm is a post configuration test scheme which makes use of the Boolean function of the threshold gate being implemented to test the programmable crosspoint locations on the PGMB. Each THmn gate has its own distinctive programmable co-ordinate locations. The proposed test scheme aims to test only those programmable ON crosspoints in the given programmable PGMB. This algorithm uses the functional expression that is unique to each THmn gate. The functional test scheme uses "test tuples" for the purpose of testing the programmable crosspoints. Test tuples are joint combinations of input bit patterns and previously asserted output. Table 3 can be used to clearly understand this concept. Consider the implementation of TH23 gate. Assume there is a fault at the coordinate location (1, 3). The fault at this ON point gives a faulty output F*=1 when input 001 is used. The desired output in case there is no fault at any crosspoint is F=F'(the previously asserted output). In case the previously asserted output is set to 0 and followed up with input pattern 001, it will be possible to stimulate the fault. By using a combination of F' and input bits, we can detect faults in the programmable crosspoints. These sets of inputs used to detect the faulty crosspoint locations are called "test tuples". Test tuples having one to one correspondence with the programmable cross- points are called lower order test tuples. Higher order test tuples can test for defects in more than a single crosspoint simultaneously.

ABC	F	F*(1,1)	F*(1,3)	F*(1,4)	F*(2,1)	F*(2,2)	F*(2,5)	F*(3,2)	F*(3,3)	F*(3,6)
000	0	0	0	F′=1	0	0	F′=1	0	0	F′=1
001	F′	F′	1	F′	F′	1	F′	F′	F′	F′
010	F′	1	F′	F′	F′	F′	F′	1	F′	F′
011	1	1	1	1	1	1	1	1	1	1
100	F′	F′	F′	F′	1	F′	F′	F′	F′	1
101	1	1	1	1	1	1	1	1	1	1
110	1	1	1	1	1	1	1	1	1	1
111	1	1	1	1	1	1	1	1	1	1
ABC	F	F* (5,4)	F* (5,5)	F* (5,6)	F* (6,1)	F* (6,2)	F* (6,3)	F* (6,4)	F* (6,5)	F* (6,6)
000	0	0	0	0	0	0	0	0	0	0
001	F′	F′	F′	1	F′	F′	F′	F′	F′	0
010	F′	F′	1	F′	F′	F′	F′	F′	0	F′
011	1	1	1	1	1	1	1	1	1	1
100	F′	F′	F′	F′	F′	F′	F′	0	F′	F′
101	1	1	1	1	1	1	F′=0	1	1	1
110	1	1	1	1	1	F′=0	1	1	1	1
111	1	1	1	1	1	1	1	1	1	1

Table 3. Truth Table for TH23 Gate and all faulty functions that can be resulted from single crosspoint defect

A pseudo code for the functional test algorithm is described in Table 4.

```
START
1.  Define PGMB dimensions and threshold gate to be programmed.
2.  Map THmn gate onto the PGMB and generate the corresponding truth
    table.
3.  IF OR plane defect detection is the prime objective THEN
     Set test tuples with low priority tuples having direct correspondence with
     programmable crosspoints to be used ahead of the high priority ones.
    ELSE
      Set test Tuples In order of decreasing priority levels.
4.  Use next test input tuple to check for programmable defect.
5.  IF False output is observed THEN
       Increment fault count and note possible defect location.
    ELSE
       The tested location is defect free.
6.  Go to Setp 4.
7.  Summarize the results of the test.
STOP
```

Table 4. Pseudo code describing the Functional Test Algorithm

7.5 Fault-Tolerant Placement Schemes Using FTA

Table 5 shows the number of OR locations utilized to implement few of the commonly and importantly used THmn gates.

TH12	TH23	TH24	TH34	TH33w2	TH34w2	TH44w2
4	6	10	8	5	8	7

Table 5. The number of programmable OR locations for THmn gates

Having studied the mapping patterns of THmn gates and defect distributions, it has been noticed that the OR plane is vulnerable to have a physical defect overlap with a programmable ON location of any threshold gate macro. Since a majority of programmable ON crosspoints fall on a single OR plane, it is essential to ensure OR plane reliability. With the inclusion of a redundant OR wire, the reliability of the OR plane can be enhanced. With the inclusion of a redundant OR wire, in case an OR point is defective, the connection can be moved to the redundant wire without programming other crosspoints in the column which contribute to the product term. Another advantage of introducing the OR plane is that since the OR planes are ORed together, the realization is not altered in any manner. As far as testing overheads are concerned, with the addition of a redundant row, only single additional input test tuple needs to be used to test the single OR location. If redundant OR row is not introduced, in case of a defect at OR location, the entire column will have to be moved to another location and all the corresponding crosspoint locations will have to be tested using additional test tuples for defects. Not only will the number of programmable locations increase with this approach, but the testing space will also increase drastically. In case of some of the THmn gates such as TH12, TH23w2 where no more than 50% of the programmable OR locations are used, it would be better to rearrange the columns instead of using a redundant OR row. In this section, different modelling and placement schemes that could be used to address these mapping issues are presented.

8. Conclusion

In this chapter, we introduced our new proposed clock-free nanowire crossbar architecture based on delay-insensitive logic known as Null Convention Logic. The complex clock distribution network can be removed from the hardware and many clocking-related failure modes can be intrinsically eliminated by the proposed clock-free architecture. Based on this architecture, we can construct a hierarchical architecture for executing more complex logic function, such as the 4-bit NCL register and so on. For defect tolerance, we proposed two mapping and placement techniques, namely Defect Unaware Approach and Defect Aware Approach. Even though the defect-aware approach is better than the defect-unaware approach especially when the defect rate is higher, it requires much more laborious testing (i.e., each PGMB should be tested to locate all defective crosspoints) and reconfiguration (i.e., all defective crosspoints should be avoided when the netlist is actually placed and routed) tasks. However, the defect-unaware approach can be simpler since the netlist is directly mapped without considering any defects. After that, PGMBs and can be functionally tested to locate ones with faults. These faulty ones then can be tested and reconfigured to avoid defects. Our future direction is to develop automated design optimization tools, testing schemes, and maximize the utility of PGMBs in spite of the inherent fabrication defects.

9. References

Bachtold, A.; Hadley, P; Nakanishi, T & Dekker, C. (2001). Logic circuits with carbon nanotube transistors, *Science*, vol. 294, pp. 1317-1320.

Bourianoff, G. (2003). The future of nanocomputing, *IEEE Computer,* Vol. 38, No. 8, pp. 44-53.

Bhattacharya, M. & Mazumder, P. (2001). Augmentation of SPICE for simulation of circuits containing resonant tunneling diodes, *IEEE Trans. Computer-Aided Design,* vol. 20, pp. 39-50.

Bandapati, S.K.; Smith, S.C. & Choi, M. (2003). Design and Characterization of Null Convention Self-Timed Multipliers, *IEEE Design and Test of Computers,* vol. 20, no. 6, pp. 26-36.

Chen, J.; Reed, M.A.; Rawlett, A.M. & Tour, J.M. (1999). Large on-off ratios and negative differential resistance in a molecular electronic device, *Science*, vol. 286, pp. 1550-1552.

Chen, J.; Wang, W.; Klemic, J. Reed, M.A. & Axelrod, B.W. (2002). Molecular wires, switches and memories, *Annals Of The New York Academy Of Sciences,* Vol. 960, pp. 69-99.

Chen, Y.; Ohlberg, D. & Li, X. et. al. (2003). Nanoscale molecular-switch devices fabricated by imprint lithography, *Applied Physics Letters,* Vol.82, No. 10, pp. 1610-1612.

Chen, Y.; Jung, G.; Ohlberg, D. Li, X. et. al. (2003). Nanoscale molecular-switch crossbar circuits, *Nanotechnology*, no. 14, pp. 462-468.

Culbertson, W.B.; Amerson, R.; Carter, R.; Kuekes, P. & Snider, G. (1997). Defect tolerance on the TeraMAC custom computer, I*EEE Symp. FPGAs Custom Computing Machines,* pp. 116-124.

Cui, Y. & Lieber, C.M. (2001). Functional nanoscale electronic devices assembled using silicon nanowire building blocks, *Science*, vol. 291, pp. 851-853.

DeHon, A. (2003). Array-Based Architecture for FET-Based, Nanoscale Electronics, *IEEE Transactions on Nanotechnology*, Vol. 2, No. 1, pp. 23-32.

Dekker, C. (1999). Carbon nanotubes as molecular quantum wires, *Physics Today*, pp. 22-28.

Derycke, V.; Martel, R.; Appenzeller, J. & Avouris, Ph. (2001). Carbon nanotube inter- and intramolecular logic gates, *Nano Letter,* Vol. 1, No. 9, pp. 453-456.

Durbeck, L.J.K. & Macias, N.J. (2001). The cell matrix: an architecture for nanocomputing, *Nanotechnology*, Vol. 12, pp. 217-320.

Fant, K.M. & Brandt, S.A. (1996). NULL Convention Logic: a complete and consistent logic for asynchronous digital circuit synthesis, *International Conference on Application Specific Systems, Architectures and Processors,* pp. 261 – 273.

Finkelstein, H.; Asbeck, P.M. & Esener, S. (2003). Architecture and analysis of a self-assembled 3D array of carbon nanotubes and molecular memories, *IEEE Conference on Nanotechnology*, pp. 12-14.

Franzon, P. & Nackashi, D. (2000). Moletronics: a circuit design perspective, *Int. Conf. SPIE Smart Electronics and MEMS,* vol. 4236, pp.80-88.

Goldstein, S.C. & Budiu, M. (2001). Nanofabrics: spatial computing using molecular nanoelectronics, *Proc. 28th Int. Symp. Computer Architecture,* 2001, pp. 178-189.

Goldstein, S.C. & Rosewater, D. (2002). Digital logic using molecular electronics, *Int. Solid-State Circuits Conf.,* pp. 125.

Heath, J.R.; Kuekes, P.J.; Snider, G.S. & Williams, R.S. (1998). A defect- tolerant computer architecture: Opportunities for nanotechnology, *Science*, Vol. 280, pp. 1716-1721.

Hiroshima, H.; Komuro, M.; Kasahara, N. & Kurashima, Y. (2002). Pattern defects of nanoimprint in atmospheric conditions, *International Microprocesses and Nanotechnology Conference*, pp. 22-23.

Huang, Y.; Duan, X.; Wei, Q. & Lieber, C.M. (2001). Directed assemble of one-dimensional nanostructures into functional networks, *Science*, Vol. 291, pp. 630-633.

Huang, Y.; Duan, X.; Cui, Y.; Lauhon, L.J.; Kim, K. & Lieber, C.M. (2001). Logic gates and computation from assembled nanowire building blocks, *Science*, vol. 294, pp. 1313-1316.

Huang, J.; Tahoori, M.B. & Lombardi, F. (2004). On the defect tolerance of nano-scale two-dimensional crossbars, IEEE International Symposium on Defect and Fault Tolerance in VLSISystems, pp. 96-104.

Jacome, M.; He, C.; Veciana, G. & Bijansky, S. (2004). Defect tolerant probabilistic design paradigm for nanotechnologies, *IEEE/ACM Design Automation Conference (DAC)*, pp. 1-6.

Kuekes, P.J.; Heath, J.R. & Williams, R.S. (2001). Molecular-wire crossbar interconnect (MWCI) for signal routing and communications, *U.S. Patent 6 314 019.*

Kuekes, P.J.; & Williams, R.S. (2001). Demultimplexer for a molecular wire crossbar network (MWCN DEMUX), *U.S. Patent 6 256 767.*

Lee,H. & Flynn, M.J.(1999). High-Speed Interconnect Schemes for a Pipelined FPGA, *Technical Report, Stanford University,* UMI Order Number: CSL-TR-99-786.

Mirkin, C. (2000). Programming the assembly of two- and three-dimensional architectures with DNA and nanoscale inorganic building blocks, *Inorganic Chemistry*, vol. 39, pp. 2258-2272.

Mishra, M. & Goldstein, S. (2002). Scalable defect tolerance for molecular electronics, *Workshop Non-Silicon Computation (NSC-1),* pp. 78.

Mcalpine, M.C.; Friedman, R.S. & Lieber, C.M. (2005). High-Performance Nanowire Electronics and Photonics and Nanoscale Patterning on Flexible Plastic Substrates, *Proceedings of the IEEE,* Vol. 93, No. 7, pp. 1357-1363.

Melosh, N.; Boukai, A.; Diana, F. & Gerardot, B. (2003). Ultrahigh-density nanowire lattices and circuits, *Science*, Vol. 300, pp. 112-115.

Naeimi, H. & Dehon, A. (2004). A greedy algorithm for tolerating defective crosspoints in nanoPLA design, *IEEE International Conference on Field-Programmable Technology*, pp. 49-56.

Nowick, S.M. & Dill, D.L. (1991). Synthesis of Asynchronous State Machines Using a Local Clock, *Proceedings of ICCAD,* pp.192-197.

Nicewarner-Peana, S.R.; Raina, S.; Goodrich, G.P.; Fedoroff, N.V. & Keating, C.D. (2002). Hybridization and extension of Au nanoparticle-bound oligonucleotides, J*ournal of American Chem. Soc.*, vol. 124, pp. 7314-7323.

Paulson, L.D. (2005). Researchers work on transistor successor, *IEEE Computer*, Vol. 38, No. 5, pp. 17.

Rueckes, T.; Kim, K.; Joselevich, E.; Tseng, G.; Cheung, C. & Lieber, C. (2000). Carbon nanotube based nonvolatile random access memory for molecular computing, *Science*, vol. 289, pp. 94-97.

Soh, C.; Quate, C. & Morpurgo, C. (1999). Integrated nanotube circuits: controlled growth and ohmic contacting of single-walled carbon nanotubes. *Appled Physics Letter,* vol. 75, no. 5, pp. 627-629.

Sutherland, Ivan E. (1989). Micropipelines, *Communications of the ACM,* Vol. 32, No. 6, pp. 720-738.

Stan, M. (2001). A scaling scenario for nanoelectronic technologies, Georgia Tech Conf. Nanoscience and Nanotechnology, pp.103.

Smith, S.C.; DeMara, R.F.; Yuan, J.S.; Ferguson, D. & Lamb, D. (2004). Optimization of NULL Convention Self-Timed Circuits, *Integration, The VLSI Journal,* Vol. 37, No. 3, pp. 135-165.

Smith, S.C.(2006). Speedup of NULL convention digital circuits using NULL cycle reduction, *Journal of system architecture*, pp. 411-422.

Seitz, C.L. (1980). System Timing, *Introduction to VLSI System*, pp. 218-162.

Singh, A. And Marek-Sadowska, M. (2002). FPGA interconnect planning, Proceedings of the 2002 international workshop on System-level interconnect prediction. pp 23-30.

Snider, G.S; Robinett, W.(2005). Crossbar demultiplexers for nanoelectronics based on n-hot codes, *Nanotechnology, IEEE Transactions on,* pp. 249-254.

Snider, G.S.; Kuekes, P.; Hogg, T.; Willams, R.S. (2005). Nanoelectronic Architectures, *Applied Physics,* pp 1183-1195.

Unger, S.H. (1969). Asynchronous Sequential Switching Circuits, *Wiley*, New York.

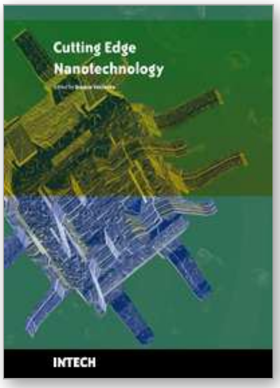
Whang, D.; Jin, S. & Lieber, C.M. (2004). Large-Scale Hierarchical Organization of anowires for Functional Nanosystems, *Japanese Journal of Applied Physics*, Vol. 43, No. 7B.

Xia, Y.; Rogers, J.; Paul, K. & Whitesides, G. (1999). Unconventional methods for fabricating and patterning nanostructures, *Chemistry Review,* vol. 99, pp. 1823-1848.

Yu, Y.S.; Jung, Y.I.; Park, J.H.; Hwang, S.W. & Ahn, D. (1999). Simulation of single-electron/CMOS hybrid circuits using SPICE macromodeling, *J. Korean Phys. Soc.,* vol. 20, no. 35, pp. S991-S994.

Ziegler, M.; Rose, G. & Stan, M. (2002). A universal device model for nanoelectronic circuit simulation, *IEEE Conf. Nanotechnology (IEEE-NANO),* pp. 83-88.

Ziegler, M. & Stan, M. (2002). Design and analysis of crossbar circuits for molecular nanoelectronics, *IEEE Conf. Nanotechnology (IEEE-NANO),* pp. 323 – 327.



Cutting Edge Nanotechnology
Edited by Dragica Vasileska

ISBN 978-953-7619-93-0
Hard cover, 444 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

The main purpose of this book is to describe important issues in various types of devices ranging from conventional transistors (opening chapters of the book) to molecular electronic devices whose fabrication and operation is discussed in the last few chapters of the book. As such, this book can serve as a guide for identifications of important areas of research in micro, nano and molecular electronics. We deeply acknowledge valuable contributions that each of the authors made in writing these excellent chapters.