A Study of Asynchronous Design Methodology for Robust CMOS-Nano Hybrid System Design

RAJAT SUBHRA CHAKRABORTY and SWARUP BHUNIA
Case Western Reserve University

Among the emerging alternatives to CMOS, molecular electronics based diode-resistor crossbar fabric has generated considerable interest in recent times. Logic circuit design with future nano-scale molecular devices using dense and regular crossbar fabrics is promising in terms of integration density, performance and power dissipation. However, circuit design using molecular switches involve some major challenges: 1) lack of voltage gain of these switches that prevents logic cascading; 2) large output voltage level degradation; 3) vulnerability to parameter variations that affect yield and robustness of operation; and 4) high defect rate. In this article, we analyze some of the above challenges and investigate the effectiveness of asynchronous design methodology in a hybrid system design platform using molecular crossbar and CMOS interfacing elements. We explore different approaches of asynchronous circuit design and compare their suitability in terms of several circuit design parameters. We then develop the methodology and an automated synthesis flow to support two different asynchronous design approaches (*Micropipelines* and *Four phase Dual-rail*) for system designs using nano-crossbar logic stages and CMOS interface data-storage elements. Circuit-level simulation results for several benchmarks show considerable advantage in terms of performance and robustness at moderate area and power overhead compared to two different synchronous implementations.

Categories and Subject Descriptors: B.2.1 [**Arithmetic and Logic Structures**]: Design styles—*Pipeline*; B.8.1 [**Performance and Reliability**]: Reliability, Testing and Fault-Tolerance—*Robust Circuit Design*; B.4.3 [**Input/Output And Data Communications**]: Interconnections(Subsystems)—*Asynchronous/synchronous operation*

General Terms: Design, Reliability

Additional Key Words and Phrases: Asynchronous design, CMOS-nano co-design, dual-rail circuits, logic degradation, micropipelines, nano-scale crossbar, robust design

ACM Reference Format:

Chakraborty, R. S. and Bhunia, S. 2009. A study of asynchronous design methodology for robust CMOS-nano hybrid system design. ACM J. Emerg. Technol. Comput. Syst. 5, 3, Article 12 (August 2009), 22 pages. DOI = 10.1145/1568485.1568486 http://doi.acm.org/10.1145/1568485.1568486

Authors' address: Department of Electrical Engineering and Computer Science, Glennan Building, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OR 44106; email: {rsc22,skb21}@case.edu.

DOI 10.1145/1568485.1568486 http://doi.acm.org/10.1145/1568485.1568486

1. INTRODUCTION

The advancement of *Nanoelectronics* in recent years has opened up the possibility of designing circuits with integration densities which are not realizable by present day photo-lithography based traditional IC fabrication techniques. Integration densities on the order of 10^{10} gate equivalents per cm^2 appear possible with emerging nanoscale devices [Goldstein and Budiu 2001]. However, having an integration density of this order would mean a significant change in prevalent design philosophy [Zieglar and Stan 2003]. In place of the traditional lithography-based "top-down" design methodology, bottom-up self-assembly based design approach appears more plausible for future nano-architectures.

Also, there is widespread agreement about the fact that these "self-directed, self-assembly" techniques would be unable to replicate the arbitrarily complex structures achievable by the conventional top-down design methodology [Zieglar and Stan 2003; Huang et al. 2004]. Instead, the bottom-up molecular circuits would be restricted to either randomly assembled structures or regular periodic structures. Although building functional circuit blocks from randomly assembled, molecular structures have been reported [Tour et al. 2002], a more attractive approach is to go for new circuit/architecture that rely on a regular and periodic fabric, which can then be configured to implement any target application. The "nanoscale crossbar" fabric [DeHon 2003; Huang et al. 2001; Li et al. 2003] is particularly suitable in this regard, where we have a two-dimensional array of devices arranged in rows and columns. The devices that form the nanoscale crossbar structure can be modeled as resistors, diodes or Field Effect Transistors (FETs) [DeHon 2003] depending on their electrical characteristics. In this work, we concentrate on nanoscale crossbar fabric realized with two-terminal diode-like devices (such as molecular monolayer of chemicals such as *rotaxanes* sandwiched between metal nanowires [Snider et al. 2005]) . The nanoscale circuits implemented using these devices fall under the "Diode-Resistor Logic" family [Zieglar and Stan 2003; Snider et al. 2005]. Figure 1(a) shows such a device structure where programmable bistable devices are sandwiched between two layers of interconnects, while Figure 1(b) shows its equivalent electrical circuit schematic. However, the main drawbacks with this simple logic family are:

(1) It is unable to provide any signal gain, which makes direct cascading of logic stages impractical, thus, requiring interface circuitry between logic stages [Zieglar and Stan 2003].
(2) The diode-resistor logic family is prone to input-dependant output voltage level degradation [Chakraborty et al. 2008] which requires level-restoring capability of the interface circuitry.
(3) These devices suffer from high defect rate and large parameter variations, which makes robust system design with these devices extremely challenging.

A straightforward way to address the first two problems is to go for a nano-crossbar structure where each junction can be configured to be a three-terminal

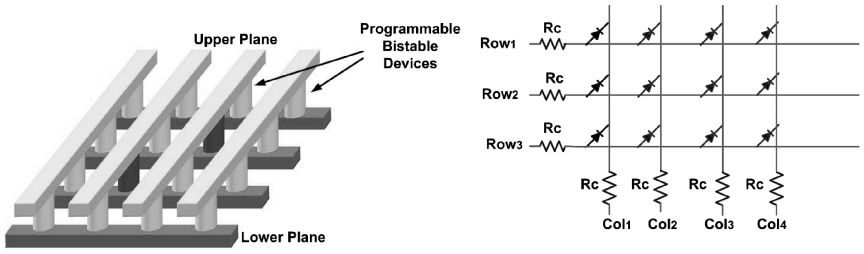

Fig. 1. Nanoscale crossbar structure consisting of two planes of crossed nanowires with bistable devices at the junctions. (a) Physical representation. (b) Equivalent circuit schematic.

device capable of providing signal gain. If this is possible, then we can simply extend the existing CMOS-based design methodology for circuit design using nanoscale crossbar structures [DeHon 2003; Snider et al. 2004]. Although progress has been made in designing molecular transistors and nano-transistor based circuits [Bachtold et al. 2001; Liu et al. 2001; Kuekes et al. 2005], the performance, fabrication and connection of three terminal devices in a dense bottom-up architecture face many major challenges and may not be realizable in the near future [Zieglar and Stan 2003; Snider et al. 2005]. For example, Kuekes et al. [2005] propose a molecular "latch" which works over a voltage range of only 0.1V, and thus suffers from serious robustness issues. In the absence of novel three-terminal transistor type devices, a "CMOS-nano co-design" approach has been proposed [Zieglar and Stan 2003] that utilizes CMOS-based level restoring and data latching circuitry at the crossbar interfaces. In this approach, each nano-block is required to have an interfacing stage of CMOS circuitry before cascading with another block. Thus this approach combines the best of both worlds by offering the high integration density of nanodevices, and the performance and predictability of CMOS. Recent advances in fabrication technology for hybrid systems and demonstration of functional CMOS-nano hybrid logic circuits [Borghetti et al. 2008] provide strong motivation for the analysis and development of hybrid computing architectures of these types.

In this article, we analyze some of the major challenges of logic circuit design using nanoscale crossbar in a nano-CMOS hybrid design paradigm and develop low-overhead design techniques to address them. We explore the asynchronous design methodology to be an attractive alternative in the context of CMOS-nano hybrid circuit designs, and among the different prevalent asynchronous design approaches, propose two design approaches which are particularly effective in such a hybrid framework. Note that our approach focuses on custom logic design and is thus fundamentally different from the "CMOL" approach [Strukov and Likharev 2007] or similar approaches based on the CMOL circuit fabric directed towards developing FPGA-like architectures. In particular, the article makes the following contributions:

(1) It investigates some of the major challenges of circuit design using diode-resistor logic based nano-crossbar circuits, including input-dependent output voltage level degradation and device process variation (Section 2)

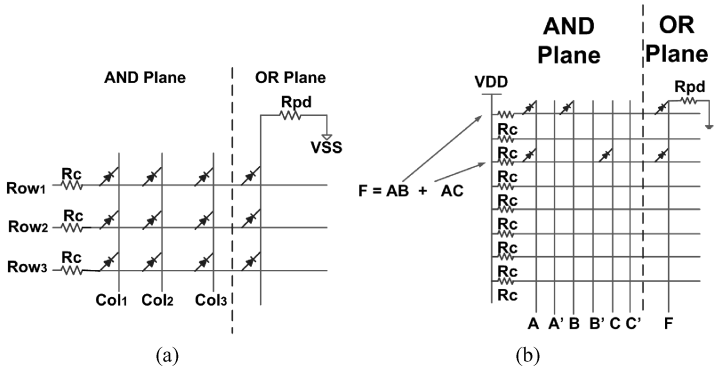

Fig. 2. (a) General structure and the PLA implementation of nano-crossbar circuits; (b) PLA implementation of the logic function $F(A, B, C) = AB + AC$.

and through simulations establish that CMOS-based asynchronous circuit elements are significantly more robust to withstand these compared to their synchronous counterparts (Section 3).

(2) It analyzes the different approaches of asynchronous circuit design to determine their suitability in the context of a hybrid nano-CMOS platform (Section 4). Through our analysis, we establish the 2-phase bundled data (*Micropipeline*) and the 4-phase dual-rail approaches to be suitable for system designs.

(3) We present a design automation flow for robust datapath design in nanoscale crossbar for the above-mentioned design approaches. An automatic circuit synthesis framework is developed to synthesize circuits from their specifications into the nanoscale crossbar platform (Section 5).

(4) We present simulation results for a set of combinational benchmark circuits to compare the two different asynchronous design methodologies with their synchronous counterpart. These simulation results show that asynchronous design can provide better performance than synchronous design approaches while incurring acceptable design overhead (Section 6).

2. CIRCUIT DESIGN CHALLENGES IN CMOS-NANO HYBRID FRAMEWORK

2.1 Nano-PLA Structure

The generic structure of a Boolean function implemented in the diode-resistor logic style is shown in Figure 2(a). The "diode-like" devices at the junctions are configurable, and the devices at the selected junctions can be independently activated and deactivated. The "row" and "column" terminals are in different planes. A device is turned "off" if a positive voltage is applied at the column terminal corresponding to it. Each row in the crossbar structure generates a unique *minterm* of a logic function. The contact resistance (R_c) is usually modeled as a lumped resistance attached to each row and column. An explicit pull-down resistance (R_{pd}) is added to each column corresponding to an output term. Boolean logic functions can be realized by this circuit architecture using two different

approaches: 1) the Look-up Table (LUT) approach and 2) the Programmable Logic Array (PLA) approach. In the LUT approach, the address decoder determines the values at the output corresponding to the minterm "address" that has been decoded, while in the PLA realization, only those minterms of the minimized function which have an output value "1" are mapped. The PLA implementation is usually more economical in terms of area and the savings in area increases exponentially as the number of inputs increases [Zieglar and Stan 2003]. In this article, we have considered the PLA representation because of its economy in terms of hardware resources. Figure 2(b) shows the PLA implementation of the simple logic function $F(A, B, C) = AB + AC$.

2.2 CMOS-nano Hybrid Deisgn: Previous Work

2.2.1 *Architectures.* CMOS-nano hybrid designs have been studied as an alternative to traditional CMOS designs and several design approaches have been proposed in the literature. Zieglar and Stan [2003], propose to use CMOS sense-amplifier and latch based level restorers at the interfaces of the diode-resistor nanologic blocks. However, this approach incurred high area and power overheads, with the level restoring circuitry occupying more than 96% of the total area and consuming 99% of the total power. Also, the authors do not discuss the effect of large process variations in the nano-devices on the performance of the level restorers. Strukov and Likharev [2007] propose another CMOS-nano hybrid architecture called *CMOL*. The *CMOL* approach is simple, dense and predicts exceptional savings with respect to power and area over traditional CMOS. However, it involves several operational and fabrication barriers to be realizable in the near future. For example, a major challenge is the nanowire size (4.5 nm) and pitch (9 nm) chosen for *CMOL*. These values are beyond demonstrated lithographic capabilities, and essentially represent an extrapolation of the ITRS roadmap out to the year 2030 [ITRS 2005]. Similarly, the supply voltage is projected to be around only 0.3V, which can make robust circuit design with such fabric difficult. In addition, it has other major issues related to routing algorithms, fabrication resolution, and logic-style [Snider and Williams 2007]. In Snider and Williams [2007], some of the shortcomings of *CMOL* has been alleviated by lifting the configuration bit and associated components out of the semiconductor plane and replacing them in the interconnect with nonvolatile switches, which decreases both the area and power consumption of the circuit.

2.2.2 *Interface Design.* The physical properties at the CMOS-nano interface create a number of fabrication challenges, such as:

—*Decoding*. The ability to address a large code space in the nano-crossbar with a smaller number of CMOS wires.
—*Pitch Compatibility*. Compatibility of the wire pitch in the CMOS technology, i.e., the microwire pitch, to the nanoscale pitch in the nano crossbar.

As in Zieglar and Stan [2003], the problem of decoding is solved in our approach by integrating the decoder into the nano-crossbar array. Since the nanoscale

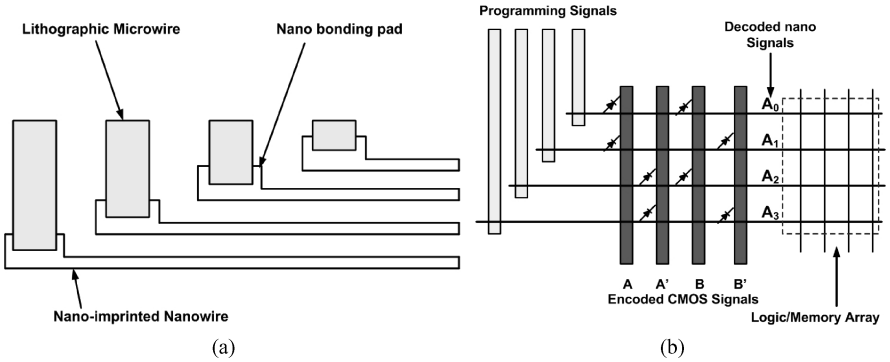

Fig. 3. (a) Nano–micro interfacing scheme; (b) Nano–micro interface signal decoding using address microwires and horizontal nanowires [Zeiglar and Stan 2003].

electronics will almost by definition be too small to fabricate by any existing generation of photolithography, the likely approach for fabricating the nanoelectronics will be imprint lithography with alignment capabilities at the scale of (or better than if available, but not necessarily) the alignment required for the CMOS circuitry [Snider and Williams 2007]. Following Zieglar and Stan [2003], we propose the pitch interfacing approach shown in Figure 3(a). It uses perpendicular microwires and nanowires; however, the nano-imprinted wires have *nano bonding pads* at the ends to provide a larger contact area for the microwires. This approach also compensates for any alignment difficulties in the nano-imprinting process. To further reduce the area overhead associated with the CMOS/nano interface structure, it may also be desirable for multiple decoder and data array pairs to share the horizontal nanowires during the initial crossbar programming. Figure 3(b) shows a probable *programmed decoding* scheme where the decoding of the signal at the interface is achieved using address microwires and horizontal nanowires. A method of postprogramming isolation can then be used to allow each decoder and data-array pair to function independently [Zieglar and Stan 2003].

2.3 Circuit Design Challenges

2.3.1 *Input-Dependent Output Voltage Level Degradation.* The basic working principle of the diode-resistor logic circuit-style is control of the amount of current flowing through the "pull-down" output resistor by the voltage applied at the input terminals. Because this current is a strong function of the input, the output voltage level varies widely with the input. It is useful for the circuit designer to have a pre-simulation estimate of the worst-case output logic voltage level for a given mapped nano-crossbar circuit, using a particular diode-like nanodevice. We have performed a detailed analysis of the input-dependent voltage variation effect in nano-crossbar circuits, leading to an accurate estimate of the crossbar output voltage level for different input combinations. In this analysis, we made several simplifying assumptions:

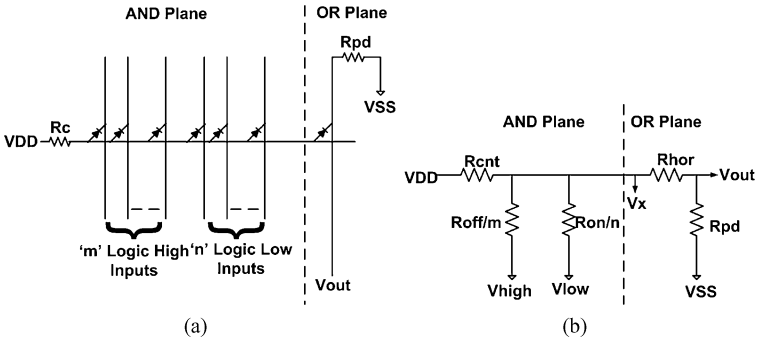

Fig. 4. Reduction of a single horizontal line in a diode-resistor logic circuit to simpler form: (a) General structure; (b) Reduced structure.

(1) The nanodevices at the crossbar junctions can be accurately represented by semiconducting diodes for the purpose of circuit analysis.
(2) The diodes within a small voltage range can be modeled as parameterized voltage-dependent nonlinear resistors.
(3) The capacitive effects of the diodes, being very small (with junction capacitances $\sim 10^{-18} F$) can be disregarded.
(4) The voltage of each horizontal line can be analyzed separately, and their effects can be "superimposed" to calculate the overall voltage of the logic function output.

These assumptions let us reduce the diode-resistor logic circuits to much simpler forms consisting only of parameterized linear and nonlinear resistors and voltage sources, as shown in Figure 4. This in turn allows one to derive closed-form analytical expressions for the output voltage levels for any input combination in a given circuit [Chakraborty et al. 2008]. These expressions are especially useful in elucidating the effect of input dependent output voltage level degradation in diode-resistor based nano-crossbar circuits. Four different scenarios for the output logic level variation are identifiable:

(1) Variation in logic-0 output voltage with the number of inputs at logic-0.
(2) Variation in logic-0 output voltage with the number of inputs at logic-1.
(3) Variation in logic-1 output voltage with the number of inputs at logic-0.
(4) Variation in logic-1 output voltage with the number of inputs at logic-1.

Among these we found the last situation to pose the most severe challenge in terms of voltage level degradation. Figure 5(a) shows the variation in the output voltage level of a 10-input OR gate implemented in diode-resistor based nano-crossbar circuits as the number of inputs at logic-1 varies for the diode-like device presented in Li et al. [2003]. From these and similar plots, we find that depending on the input, the output logic-1 voltage level can vary by more than 25%. It is evident that logic-1 degradation is a major problem in the design of diode-resistor logic based nano-circuits. The level degradation can be even more severe in case of larger PLAs or weaker diodes, as shown in Figure 5(b). In

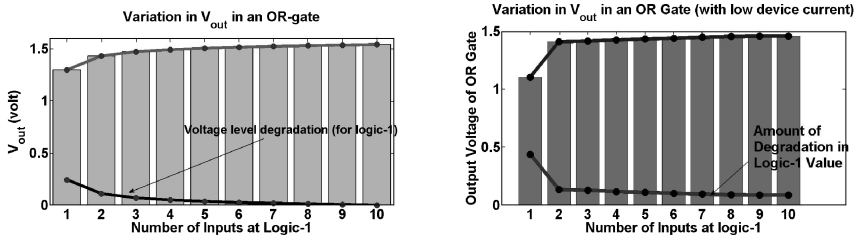

Fig. 5. Variation in logic-1 output of an OR gate with the number of inputs at logic-1: (a) Device with nominal current; (b) Device with reduced current.

our analysis we did not consider the effect of process variations on output level degradation. However, one can envisage that in the presence of large amounts of random process variations in the nanodevices, the design challenge would be greatly exacerbated. As mentioned in Section 1, level-restoring circuitry is mandatory in the diode-resistor circuit style as direct cascading is not possible due to lack of gain. The degradation of logic-1 makes the design of the interface circuitry in an electrically noisy environment very challenging. Hence, any CMOS-nano hybrid design methodology that we develop should be robust with respect to this voltage variation effect.

2.3.2 *Large Device Defect Rate.* The nanodevices are susceptible to extremely large manufacturing defect rates (as high as 10% defective devices in a given fabric) compared to present CMOS-based circuits [Nikolic et al. 2002]. To design functional circuits in the presence of such high defect rates, novel testing and application-mapping techniques are required. Also, such techniques should be *scalable* in the sense that they should incur acceptable overheads irrespective of the application size/complexity and the defect density [He et al. 2005]. Several highly effective defect-tolerant application mapping techniques and related design automation tools have been proposed in the literature [He et al. 2005; Nikolic et al. 2002; Hogg and Snider 2007]. The CMOS-nano hybrid logic design framework proposed in this paper can take advantage of them in achieving high levels of fault tolerance.

2.3.3 *Process Variation Effects.* Besides the input-dependent logic level degradation issue described in Section 2.3.1, the nanodevices are inherently susceptible to large scale process variations. In case of diode-like nanodevices and the diode-resistor nano-PLA circuits that we consider, these process variation effects manifest themselves as variation in the diode drive strength, the diode junction capacitance, the contact and pull-down resistance values, etc. In the next section, we examine the robustness and variation tolerance of synchronous interface circuitry and their asynchronous counterparts.

3. ASYNCHRONOUS CIRCUIT ELEMENTS IN THE CMOS-NANO HYBRID FRAMEWORK

Asynchronous circuits are attractive alternatives to traditional synchronous ones because of low power consumption, high operating speed, lesser

Table I. Comparison of Process-Variation Tolerance of Asynchronous and Synchronous Crossbar Circuits

Parameter	Asynchronous Interface Circuitry	Synchronous Interface Circuitry
CMOS Supply Voltage Variation (nom. 1.5V)	0.45–3.0 V	1.24–3.0 V
Crossbar Supply Voltage Variation (nom. 4.5 V)	0.30 V	0.20 V
Diode Drive Strength	1000X reduction	950X reduction
Zero-bias Junction Capacitance of Diode (C_{j0})	8500X increase	550X increase
R_c and R_{pd} (% nom.)	24.5%	10.0%

supply-voltage noise generation (due to randomized "ticking" of local clocks), absence of clock-skew problems, and most importantly: robustness towards supply voltage, temperature and fabrication process parameter variations [Sparsø and Furber 2001; Cortadella et al. 2002; Josephs et al. 1999]. This provided the motivation in investigating the process variation tolerance of CMOS asynchronous interface circuitry in a CMOS-nano hybrid design framework, where the level-restoring and data storage circuitry is implemented using CMOS. We considered *event-driven latches* and *dual-rail latches* as representative interface circuits for the asynchronous class [Sutherland 1989; Sparsø and Furber 2001]. We also considered edge-triggered clocked flip-flops and sense amplifier based latches described in Zieglar and Stan [2003] as their synchronous counterparts. Note that synchronous *pulsed latches* are not suitable as interface data storage elements, as large process variation effects in the nanodevices would increase the uncertainty in meeting the timing constraints in pulsed latches [Rabaey et al. 2003]. For our simulations, we used the SPICE level-1 diode model derived from the experimental data in Li et al. [2003], and 70nm *Predictive Technology Model* (PTM) [NIMO Group 2007] for the CMOS circuits. The variation was distributed randomly in the devices and circuit elements of the nanofabric, and HSPICE was used for the circuit simulations. Table I shows a comparison of variation tolerance between the asynchronous and synchronous data storage circuits for five major design parameters. The second and third column list the maximum tolerable variation in a parameter beyond which the circuit manifest functional failure. From the table, we can observe that the asynchronous system has significantly better variation tolerance than its synchronous counterpart for all the parameters considered, including the input-dependant output voltage variation (which is also related to the supply voltage variations) effect discussed in Section 2.3.1.

Having obtained the initial motivation for CMOS asynchronous interface circuitry-based hybrid design from our simulation results, in the next section we investigate the different popular approaches of asynchronous datapath design to analyze their suitability in a diode-resistor logic based CMOS-nano hybrid design framework.

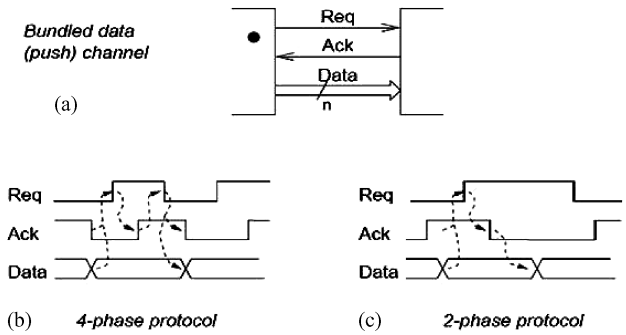

Fig. 6. Bundled data handshaking protocols [Sparsø and Furber 2001]: (a) general structure; (b) 4-phase protocol; (c) 2-phase protocol (also referred as *micropipeline*).

4. ASYNCHRONOUS NANO-CROSSBAR BASED SYSTEM DESIGN APPROACHES

In an asynchronous circuit, the clock signal is replaced by some form of *handshaking* between neighboring data-storage registers. These handshaking signals coordinate the data transfer between them. The combinational computation elements (other than the data storage and control path elements) in an asynchronous system are usually called *function blocks*. Two fundamental principles of an asynchronous circuit are:

—A register may input and store a new data token from its predecessor if its successor has input and stored the data token that the register was previously holding, and,
—the function blocks should be transparent to the handshaking between the data storage registers.

Next, we introduce the system design approaches in a typical asynchronous computing platform.

4.1 Asynchronous Design Approaches

The handshake between adjacent logic stages take place via the *Acknowledge* and *Request* signals, as shown in Figure 6(a). Based on the handshaking protocol between contiguous logic stages, asynchronous circuits are classified into three main types:

(1) The *Bundled Data* type (Figure 6(a)), which depend on *delayed* handshaking signals between adjacent logic blocks, with the delay matching the worst case function block delay. These are again classified into two main types: (a) the 4-phase bundled data type and (b) the 2-phase bundled data type.
(2) The *Dual-rail* type, where the handshaking signals are themselves generated from the processed data tokens.
(3) The *Completion-detection* type, where the handshaking signals are generated after sensing the completion of the generation of the data tokens.

We next describe the operation of these different asynchronous design approaches in detail.

4.2 Bundled-data Handshaking Protocol-based Asynchronous Circuits

Figure 6(b) shows the handshaking sequence in case of the *4-phase bundled data* protocol. The following events take place in sequence: (1) the sender issues data and sets *request* high; (2) the receiver absorbs the data and sets *acknowledge* high; (3) the sender responds by taking *request* low (at which point data is no longer guaranteed to be valid) and (4) the receiver acknowledges this by taking *acknowledge* low. At this point the sender may initiate the next communication cycle. The *advantage* of this approach is that it is easy to visualize and the interface elements are very easy to design. The main *disadvantage* of this approach is that it involves a superfluous return-to-zero, wasting time and energy. Another flavor of the bundled data protocol is the 2-phase bundled data protocol, also known as *Micropipelines*, introduced and popularized by Sutherland [1989]. In this protocol, information on the request and acknowledge wires is encoded as signal transitions on the wires and there is no difference between a $0 \rightarrow 1$ and a $1 \rightarrow 0$ transition, they both represent a "*signal event*." The main *advantage* of this approach is that it saves on the time and energy overheads of 4-phase bundled data, however, the main *disadvantage* is that often the implementation of the circuits is more complex, especially the "event-driven" elements compared to the level-sensitive elements of a 4-phase protocol [Sparsø and Furber 2001].

The real challenge of the bundled-data design methodology is that all the bundled-data protocols rely on delay matching, such that the order of signal events at the sender's end is preserved at the receiver's end. Hence, the generated data must be valid before the *request* signal is set high. This is usually ensured by introducing carefully designed explicit delay elements corresponding to the worst-case computation time of the combinational blocks. However, in a scenario where the combinational logic is susceptible to large process and parametric variations, the delay might vary widely. As a result, designing balanced delay elements and testing each such datapath stage for delay faults can be extremely challenging. The designer can follow a conservative methodology and make the introduced delays large enough to cover the process variation effects of the combinational block, but this adversely affects performance. Also, the introduced delay elements cause area and power overheads.

4.3 Dual-rail Handshaking Protocol-based Asynchronous Circuits

To avoid this problem, in the *dual-rail approach*, the *request* signal is encoded into the data signals themselves. In essence it is a 4-phase protocol using two request wires per bit of information d; one wire *d.t* is used for signaling a logic 1 (or true), and another wire *d.f* is used for signaling logic 0 (or false). When observing a 1-bit channel one will see a sequence of 4-phase handshakes where the participating *request* signal in any handshake cycle can be either d.t or d.f. This protocol is very robust; two parties can communicate reliably

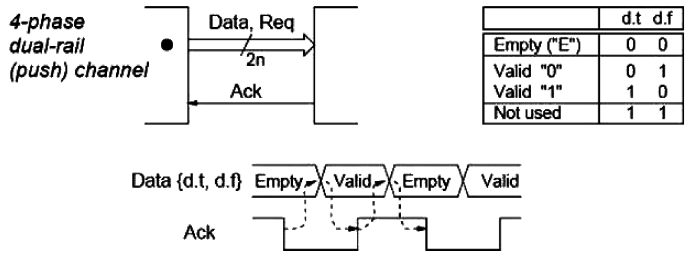

Fig. 7. 4–phase dual-rail handshaking protocols [Sparsø and Furber 2001].

regardless of delays in the combinational blocks between data storage stages—thus the protocol is *delay-insensitive*. For n-bits of data, the "*all valid*" codeword consists of $d.t \oplus d.f = 1$ for each data bit, and the "all empty" codeword consists of $d.t = d.f = 0$ for each data bit. The condition $d.t = d.f = 1$ is never used. The structure, codewords and handshaking signal waveforms for the 4-phase dual-rail circuits have been shown in Figure 7. The main *advantage* of this approach is obviously the delay-insensitivity, but the main *disadvantages* are the area (essentially double), timing and power overheads incurred compared to a bundled data approach. Also, the data storage elements are more complex compared to the 4-phase bundled data case.

4.4 Completion Detection Protocol-Based Asynchronous Circuits

The other approach of asynchronous design rely on completion detection of the data token computations (either by voltage sensing or current sensing). Two common approaches are *Differential Cascode Voltage Switch* Logic (DCVSL) based on dual-rail encoding [Rabaey et al. 2003] and the *Current-Sensing Completion Detection* technique [Dean et al. 1991]. However, none of these techniques is suitable in the context of diode-resistor crossbar circuits. In the DCVSL style, a logic function and its complement are generated and compared to detect completion. However, this requires an active device to pre-charge the output nodes before each logic evaluation [Rabaey et al. 2003] which is not available in a diode-resistor logic network. On the other hand, the current-sensing completion detection technique is based on detecting the completion of switching operation in the combinational logic output by monitoring the current flowing through the logic. However, this is more suited to the static CMOS logic family where ideally there is no current flow in the absence of switching, which is not the case in diode-resistor logic. Besides, this approach requires careful analog design for current monitoring [Rabaey et al. 2003] and ensuring the reliability of this operation under process variation is an obvious challenge.

4.5 CMOS-Nano Hybrid Asynchronous Design

We propose to design asynchronous logic structures in a nano-CMOS hybrid framework by implementing the function blocks (computation blocks) in diode-resistor logic based nano-PLA, and the data storage elements in CMOS. The first two asynchronous design approaches discussed in Section 4.1 are suitable

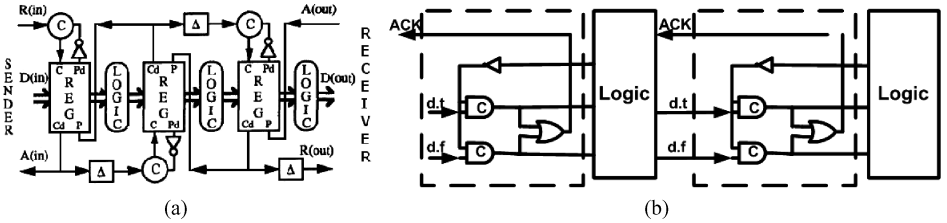

Fig. 8. Different datapath structure implementations in the proposed hybrid CMOS-nano framework: (a) 2-phase bundled data (Micropipeline) [Hauck 1995] and (b) 4-phase dual-rail [Sparsø and Furber 2001]

for datapath implementation; however, because of the complicated handshaking protocol, associated bidirectional timing requirements and power/performance overheads of the 4-phase bundled data convention, we consider only the 2-phase bundled data convention and the 4-phase dual-rail convention. Figure 8(a) shows the datapath implementation by the 2-phase bundled data approach, while Figure 8(b) shows the 4-phase dual-rail approach. The handshaking control is usually carried out by the *Muller-C elements* (denoted by C in the figure), which essentially act as *OR gates for events* [Sparsø and Furber 2001]. The dual-rail approach does not need any delay element in control signal paths. However, the 2-phase bundled data approach needs such matched delay elements (shown by Δ in the figure). Following the estimates in Strukov and Likharev [2007] and Snider and Williams [2007], for a nanowire with a pitch of 30nm and width 15nm, the resistance per unit length is $355\Omega/\mu m$, and the capacitance per unit length is 2pF/μm. The nanodevice junction capacitances are $\sim 10^{-18}$F [Zieglar and Stan 2003]. We consider crossbar sizes to be 8×8 and 16×16, which are typical of those considered in the literature for analysis and of the same order as the real implementation reported in [Borghetti et al. 2008] (21×21). Performing the circuit simulation following the nano-interconnect circuit model proposed in Chatterjee and Roy [2003], the *worst case* interconnect delay in a 8×8 crossbar circuit is $\sim$2.04ps and that for 16×16 crossbar is $\sim$8.18ps. The ITRS roadmap for interconnects suggests a Cu-interconnect delay of 0.27ps and 0.54ps respectively at similar crossbar dimensions. The simulated delay to a minimum-sized 70nm CMOS buffer was found to be $\sim$4.8ps. Hence, the 8×8 crossbar logic stage requires one buffer and the 16×16 logic stage requires two such buffers respectively. This is an obvious overhead in the *Micropipeline* approach and offsets its high performance advantage somewhat.

The structure and operation of the asynchronous pipelines essentially isolates the process variation effects of each logic stage such that it affects only the interface elements connected directly to it. As the asynchronous interface elements were shown to be resistant for wide variations in the diode-resistor circuit parameters, each stage of the asynchronous pipeline is individually robust to process variation effects. Due to the handshaking between stages and due to the absence of any global synchronizing signal, robustness of one pipeline stage implies the robustness of the entire pipeline.

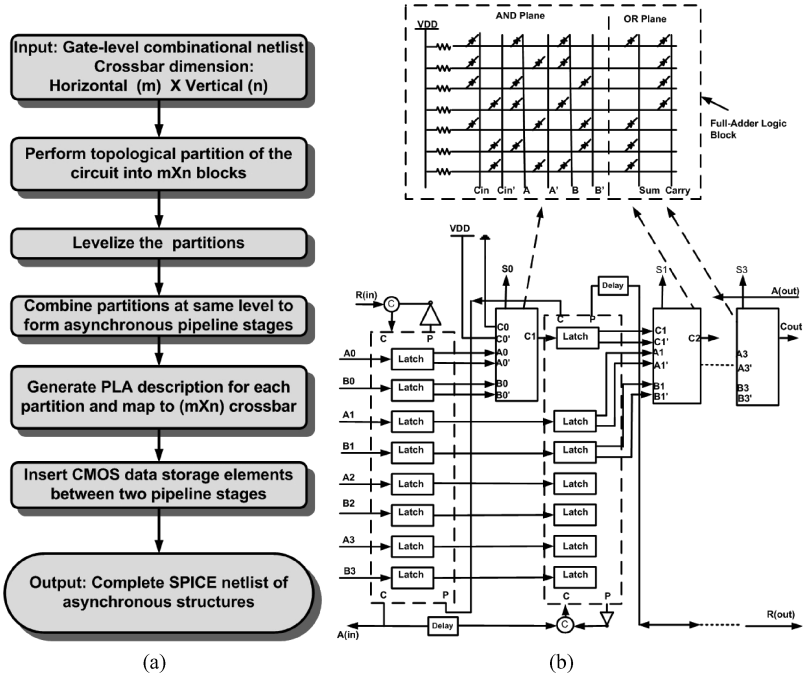

Fig. 9. (a) The proposed micropipeline-based asynchronous design flow for nano-crossbar; (b) Asynchronous micropipelined implementation of a 4-bit pipelined *ripple-carry adder*.

5. DESIGN METHODOLOGY AND AUTOMATION

In this section, we describe the proposed asynchronous design methodology and the related design flow for nanoscale crossbar-based datapaths. Figure 9(a) shows the complete design flow. The methodology is based upon creating a *topological partition* of a gate-level combinational netlist (in Verilog, SPICE or *bench* formats) based on two design parameters: the maximum allowed horizontal (m) and vertical dimension (n) of the crossbar matrix. Following the partitioning step, an optimized equivalent description of each partition in the PLA format is generated which is then mapped to diode-resistor logic structure. This method of partitioning gives rise to a description of the circuit that is amenable to pipelined implementation, where the pipeline stage corresponding to each partition is determined by the input-output dependencies between the partitions. Based on an input flag, either the Micropipeline or the dual-rail structure is synthesized. In case of a dual-rail implementation, both the PLA structures corresponding to a Boolean function (F) and its complement ($\overline{F}$) are generated. Once this is achieved, latches (event-driven in case of Micropipelines and C-element based in case of 4-phase dual-rail) along with the associated control circuitry are introduced at the interface of two partitions between successive pipeline stages. Note that this method of inserting latches can automatically handle any feed-forward situation. For the Micropipeline implementation, the required number of buffers (acting as delay elements) are

also inserted in the design. In the rest of this section, we describe the steps in details.

We start by reading the SPICE netlist of the circuit expressed in terms of a predefined library of basic gates (such as AND, OR, NOR, NAND, INVERTER). User-specified values of the parameters m and n determine the maximum crossbar block size, and an user-defined flag determines the design technique (2-phase bundled data or 4-phase dual-rail). We then represent the netlist as a hypergraph and create a *topological partition* of the hypergraph using a modified hypergraph partitioning [Karypis et al. 1999] approach. The partitioning routine tries to minimize the number of partitions, while maintaining the input/output relationship and constraining the input/output cut edges to m and n, respectively. When the partitioning is over, an alternate description of the circuit in terms of the optimized PLA representations of the partitions is generated. A PLA optimization tool (*Espresso* [Rudell and Sangiovanni-Vincentelli 1987] in our case) is used to minimize the PLA expression for the partition outputs. Both true and complementary form of each partition output is considered to maximize compression of PLA logic.

In the next step, the SPICE level subcircuit corresponding to each partition is mapped to diode-resistor crossbars of size $m \times n$. Next, the partitions are levelized based on their input/output dependencies. Parallel partitions in a particular level constitute the stage logic in the pipeline. Thus, more partition level parallelism can help to reduce the number of stages. Similarly, larger partitions (with higher m and/or n) can also help in reducing the pipeline depth. Each partition is assigned a *level* which shows the *logical depth* of the partition stage. If two or more partitions belong to the same level (i.e., they are parallel), the latches at the interface between them and the following stages are driven by the same control signals. Next, the input nodes of the partition are examined and feed-through signals are propagated through extended pipeline latches to control the pipeline timing. For example, if a partition at level–3 has one of its inputs coming from another partition at level–1, two latches connected in series would be introduced for the asynchronous structure. Finally, the individual partitions, described as SPICE level subcircuits, are instantiated with the appropriate interface, control and inserted delay circuitry (in case of Micropipelines) to regenerate the complete circuit. Figure 9(b) shows the micropipelined implementation of a 4-bit pipelined ripple-carry adder following the above design methodology. Similar structures are generated for the 4-phase dual-rail approach, with both the implementation of the partition as well as its complement, and the event-driven latches being substituted by Muller C-element based latches.

For the sake of comparison with asynchronous implementation, we also implemented an equivalent synchronous pipeline in which the interface elements are clocked flip-flops. We also implemented the sense amplifier based approach of Zieglar and Stan [2003], without the level-shifting circuitry at the interface. We did not need the level-shifting circuitry because our contact resistance values and CMOS device sizing was done in such a way that the diode-resistor based nano-PLA circuit was capable of operating with CMOS interface circuit directly, without any level shifting. This caused considerable savings in area

and power compared to the values reported in Zieglar and Stan [2003]. We verified the equivalence of these different implementations by comparing logic values in the output of two pipelines for all test circuits.

6. RESULTS

In this section, we present simulation results for several combinational benchmark circuit implementations by the proposed asynchronous methodology, For the sake of comparison, we also include the results for the synchronous implementations of the same benchmark circuits using clocked flip-flop and sense amplifier based latches [Zieglar and Stan 2003] respectively.

6.1 Simulation Setup

A subset of the ISCAS-85 benchmark combinational circuits[1] were subjected to the proposed design flow. The simulations were carried out using HSPICE. The synchronous pipelines were operated at their maximum feasible clock frequencies, determined by worst-case delay of the combinational logic blocks and the respective data storage elements. Input vectors were applied at a rate of 100MHz. The input vectors were generated by an automatic weighted random pattern (WRP) generation tool. The average activity of the input bits was set to be 33%. For the nano-diodes, we used the SPICE level-1 diode model which was derived from the experimental data presented in Li et al. [2003], by fitting the I_S and N SPICE parameters to this experimentally obtained data. We chose this diode model because this device is capable of delivering a current in the order of 1μA at a voltage of around 1.5V, which is sufficient of driving the CMOS circuitry at the interfaces at high frequencies. We used the 70nm *Predictive Technology Model* (PTM) [NIMO Group 2007] for the CMOS circuits. The pull-down and contact resistance values in the diode resistor logic were taken as 1MΩ and 100kΩ, respectively. These values were chosen since they gave the largest voltage swing at the output with the other factors remaining constant. As in Zieglar and Stan [2003], the voltage supplies for the diode-resistor logic were taken as 3V and –1.5V and 1.5V for the CMOS logic. Although ultrahigh-density nanowire structures have been reported [Melosh et al. 1998], we assume a nanowire pitch to be 30nm, following [Snider and Williams 2007].

To estimate the parasitic capacitance of the nanowires, we follow Strukov's model for a regular, parallelepiped nanowire crossbar [Strukov and Likharev 2007], where nanowires within a layer are separated by a distance equal to their width. Given a 3-nm-thick switching layer separating the two nanowire layers, a nanowire width of 15 nm, and an insulator between and around all nanowires with a dielectric constant of 3.9 (that of SiO_2), that model predicts a capacitance per length of approximately 2.8 pFcm^{-1}. We estimated the effective resistivity of a 15nm-wide nanowire interconnect from a projection of the ITRS roadmap value to be 8 $\mu\Omega$cm, yielding the resistance per unit length of a 15nm-square nanowire to be 355 $\Omega\mu$m^{-1}. The interconnect parasitics were included in the

[1]http://www.fm.cslib.cz/kcs/asic/iscas/.

Table II. Area Comparison of the Asynchronous and Synchronous CMOS-Nano Hybrid Datapath Implementations

Size	8×8				16×16			
Type	Async.		Sync.		ASync.		Sync.	
Ckt.	2-phase (μm^2)	4-phase (μm^2)	F/F (μm^2)	SA (μm^2)	2-phase (μm^2)	4-phase (μm^2)	F/F (μm^2)	SA (μm^2)
c17	43	86	70	21	39	80	65	23
c499	3922	8501	7079	2152	3167	6875	5670	2027
c432	4635	10053	18180	2572	3134	6781	5588	2229
c1908	7079	15385	12938	3941	4819	10470	8606	3452
c880	7825	17003	14285	4346	4146	8988	7400	3056
c1355	10953	23044	19395	5912	7675	16684	13725	5496
c2670	12374	26872	23644	7296	8056	17465	14190	8640
c5315	36255	78889	68222	20945	29253	63624	52097	25392
c6288	50781	110645	92751	28216	33707	73408	60485	23317
c7552	30659	66488	61992	19411	28498	61786	52784	25757

SPICE simulations as lumped resistances and capacitances between stages, proportional to the inter-stage distance (i.e., the parasitics between stage-1 and stage-3 is double of that between stage-1 and stage-2). The simulations were carried out at a nominal process corner at room temperature without any defects introduced in the crossbar fabric.

6.2 Area Results

Table II shows the area required by the combinational blocks and interface circuitry for the asynchronous and synchronous implementations. The asynchronous implementations considered are the 2-phase bundled data and the 4-phase dual-rail approaches, while the synchronous implementations use clocked flip-flops and sense-amplifier based latches [Zieglar and Stan 2003] respectively. The area A of the combinational part mapped into the crossbar fabric is given by Zieglar and Stan [2003]:

$$A = c \cdot (2N + f) \cdot P_w^2 \tag{1}$$

where N is the number of literals in all the functions implemented by the crossbar, f is the number of functions, c is the number of unique two-level minimized product terms in all the functions, and P_w is the pitch of the nanowire (being 30nm, as discussed before). In a 4-phase implementation and the sense amplifier based implementations, on average roughly twice this area is needed because both F and $\overline{F}$ needs to be implemented. In the 2-phase asynchronous pipeline, the interface area is contributed by one event-driven latch per interface signal, one Muller-C element and one inverter per stage, plus the required bondpads at the CMOS-nano interface It also requires one minimum-sized buffer acting as a delay element per logic-stage. On the other hand, the 4-phase implementation requires 4-phase latches for each interface signal pair and one inverter per logic stage along with the required bondpads. The bondpad dimension was estimated to be half the microwire pitch (90nm according to the ITRS roadmap in 2010) on each side. The flip-flop-based synchronous pipeline has clocked flip-flops at the interfaces of the pipeline stages, while the sense

Table III. Power Comparison Between the Asynchronous and Synchronous CMOS-Nano Hybrid Datapath Implementations

Size	8×8				16×16			
Type	Async.		Sync.		ASync.		Sync.	
Ckt.	2-phase (mW)	4-phase (mW)	F/F (mW)	SA (mW)	2-phase (mW)	4-phase (mW)	F/F (mW)	SA (mW)
c17	0.54	1.08	0.80	1.07	0.54	1.12	0.80	1.13
c499	1.24	2.69	3.41	1.06	6.74	14.63	14.16	6.34
c432	2.75	5.96	6.04	2.13	2.75	5.95	6.04	2.50
c1908	4.76	10.34	7.06	3.43	8.82	19.16	10.79	7.86
c880	3.30	7.17	8.87	2.64	7.64	16.56	15.47	7.18
c1355	6.91	15.03	11.05	5.08	10.73	23.33	13.04	9.13
c2670	2.82	6.12	9.08	2.57	8.37	18.14	12.57	10.37
c5315	8.93	19.43	16.51	6.99	15.80	34.36	20.38	15.98
c6288	8.44	18.39	15.18	6.07	12.81	27.90	18.45	10.40
c7552	14.49	31.42	33.03	12.47	16.93	36.71	24.31	23.94

amplifier based approach requires sense amplifier based latches at the interfaces, plus the bondpads. Separate inverters to generate the inverted versions of the latched signals were not required, as they were generated by the CMOS data storage circuits themselves.

Note that generally, the interface data storage elements are larger than those in CMOS-only designs, because a diode-resistor based PLA circuit outputs a signal without any gain, and as such stronger feedback circuitry is required to latch a new data. We also note that the interface area overhead generally decreases for larger combinational circuits, since the area of the combinational blocks (implemented with crossbar) constitutes a larger fraction of the total area. The area results consider only the *active area*, and is not the actual layout area. Also, the area results are computed as the sum of the area requirement for the nano-crossbar and CMOS circuitry. In a scenario where the nano-PLA can be implemented in a different "stacked" layer than the CMOS, the area estimate can be given by the larger of the nano-crossbar and CMOS parts.

6.3 Power Results

Table III compares the power dissipation in ISCAS-85 benchmark circuits for asynchronous (2-phase bundled data and 4-phase dual-rail) and synchronous (clocked flip-flop and sense amplifier based implementations). The clocked flip-flop implementation and the 4-phase dual-rail implementations usually incur the highest power overhead. The relatively high power dissipation in the clocked flip-flop implementation is due to the clocking of *every* data-storage element in the design, while that of the 4-phase dual-rail implementation is due to the doubling in the number of data storage elements and approximate doubling of the combinational circuitry. Also, among the synchronous implementations, the clocked flip-flop implementation has higher power dissipation than the sense–amplifier based design style. An exception to this trend is observed in the case of c17. The c17 circuit is the smallest circuit among the ISCAS-85 benchmarks, and had only a single stage of logic between the pipeline stages. It was the only circuit where the power dissipation of the nano-crossbar dominated the power

dissipation of the interface elements. Because the S/A based implementation requires both the logic function F and $\overline{F}$, hence in the case of c17, the S/A based implementation consumes more power than the clocked flip-flop based implementation.

6.4 Delay Results

The basis of comparison of the four implementations that we have considered was how frequently a new set of data can be processed by the datapath pipeline, that is, after the time interval between the application of the input data and the availability of the processed data at the output. In the CMOS-nano hybrid implementation, the speed of data processing is determined mainly by the delay of the interface elements, because the diode resistor logic based nano-PLA circuitry as well as the metal bondpads at the microwire/nanowire interface have extremely small delay. In the asynchronous implementations, this is determined by how the CMOS interface elements react to a change of values at their input and latch the data correctly. In our case, the event-driven latches (in the 2-phase implementation) this delay was 1.20ns and the dual-rail latches (in case of the 4-phase dual-rail approach) had a corresponding value of 0.80ns for the 8×8 crossbar structure and 1.05ns for the 16×16 crossbar structure. The discrepancy is due to the fact that the completion detection circuitry for the 16-bit registers at the interfaces are larger and more complex than those of the 8-bit registers.

For the synchronous implementation using clocked flip-flops, this is given by the well-known relationship between minimum feasible clock period (T_{min}), the clock-to-Q delay ($T_{C\rightarrow Q}$), the *setup time* (T_{setup}) of the flip-flops and the worst-case logic-stage delay (T_{pd}):

$$T_{min} = T_{C\rightarrow Q} + T_{setup} + T_{pd} \tag{2}$$

In our case, $T_{C\rightarrow Q}$ was 1.5ns, T_{setup} was 15ps and the worst case T_{pd} was found to be 0.27ps for a 8×8 crossbar and 0.54ps for a 16×16 crossbar. Hence, the minimum clock rate for the synchronous implementation using flip-flops was 1.52ns. For the sense amplifier based interface design approach, we estimated the operating frequency from the total time required to process a new set of data in a cascaded pipeline stages with transparent latches. A single set of transparent latches added a delay of 1.10ns. Table IV compares the delays of the different asynchronous and synchronous implementations, which includes the delay due to the interface elements and the combinational logic delay. The latency of the synchronous implementation with clocked flip-flops is the highest because its operating clock frequency is limited by its relatively high clock-to-Q delay.

The process variation effects in nanocircuits are usually random and rarely systematic. A major advantage of the asynchronous design style is its robustness to these effects. The data storage elements in an asynchronous pipeline automatically adjust to the variations in the logic stage and interconnect delays. As mentioned before, we implemented the delay elements as CMOS buffers, not as *matched delay* lines. The process variations the CMOS-based delay elements

Table IV. Delay Comparison between the Asynchronous and Synchronous CMOS-Nano Hybrid Datapath Implementations

Size	8×8				16×16			
Type	Async.		Sync.		ASync.		Sync.	
Ckt.	2-phase (ns)	4-phase (ns)	F/F (ns)	SA (ns)	2-phase (ns)	4-phase (ns)	F/F (ns)	SA (ns)
c17	1.23	0.83	1.52	1.18	1.25	1.10	1.52	1.15
c499	11.04	7.44	13.68	10.59	8.78	7.73	10.64	8.08
c432	12.27	8.27	15.20	11.77	13.79	12.14	16.72	12.69
c1908	11.04	7.44	13.68	10.59	8.78	7.73	10.64	8.08
c880	13.50	9.10	16.72	12.95	12.54	11.04	15.20	11.54
c1355	11.04	7.44	13.68	10.59	11.29	9.94	13.68	10.39
c2670	12.27	8.27	15.20	11.77	11.29	9.94	13.68	10.39
c5315	11.04	7.44	13.68	10.59	11.29	9.94	13.68	10.39
c6288	13.50	9.10	16.72	12.95	15.05	13.25	18.24	13.85
c7552	12.27	8.27	15.20	11.77	11.29	9.94	13.68	10.39

are much less compared to those in the nano diode-like devices and the nano interconnects. Hence, any process variation effect of the CMOS delay elements can be lumped with that of the nano crossbar and nano interconnects.

6.5 Exploration of the Design Space in Search of the Optimal Design Choice

The area, power and delay results presented can be used to answer two important questions for a nano-system designer:

(1) Which asynchronous implementation style to choose?
(2) What maximum crossbar dimension to choose for a partition?

The answer to the first question will depend on both design targets as well as the nanodevice characteristics, for example, process variation effects. For an implementation where robustness with respect to parameter variations is the priority, an asynchronous approach should be chosen. Among the asynchronous implementations, the 2-phase bundled data implementation incurs considerably less overhead compared to 4-phase dual-rail implementation. However, if the design of accurate delay-balancing elements in case of the 2–phase bundled data implementation becomes difficult due to large process variations, the designer should opt for the 4-phase dual-rail implementation at the cost of increased design overhead. Note that the 4-phase bundled data circuits work with a complicated handshaking protocol and has associated bidirectional timing requirements and power/performance overheads due to a superfluous return-to-zero transition. For this reason, it is rarely implemented in practice [Sparsø and Furber 2001]. We did not consider this approach since such timing requirements are difficult to meet with molecular crossbar that suffer from high variations.

As mentioned in section 4.5, we considered crossbar sizes to be 8X8 and 16X16, which are typical of those considered in the literature for analysis and of the same order as the real implementation reported in Borghetti et al. [2008]. Large crossbar partition dimensions are not very attractive for the

following reasons.

(1) Nanoscale crossbars suffer from high defect density (10%), which makes application mapping difficult for large crossbars [He et al. 2005].
(2) Large crossbars are inefficient with respect to the area utilization. This is because only a small fraction of the available crossbar junctions are utilized to generate the minterms of the logic functions mapped onto it.

7. CONCLUSION

Diode-resistor based nanoscale crossbar circuits have emerged as a promising alternative to CMOS. However, circuit design in this fabric faces several challenges including input-dependent logic-level degradation, lack of gain of the devices that prevent logic cascading, and vulnerability to large process variations. To address these challenges, we have presented a CMOS-nano asynchronous codesign methodology. We have shown that such a design technique is more robust to parametric variations than its synchronous counterpart. We have analyzed prevalent asynchronous design techniques regarding their suitability in the context of nano-PLA based CMOS-nano codesign, and found the two-phase bundled data and the 4-phase dual-rail approaches to be effective for robust system design. An automated design flow has been developed to synthesize datapaths at the circuit level from higher level descriptions. Future work would involve extending this methodology to complex sequential logic and achieving defect tolerance towards a complete CMOS-nano asynchronous computing platform.

REFERENCES

BACHTOLD, A., HADLEY, P., NAKANISHI, T., AND DEKKERDAGGER, C. 2001. Logic circuits with carbon nanotube transistors. *Science 294*, 2, 1317–1320.

BORGHETTI, J., LI, Z., STRAZNICKY, J., STEWART, D., LI, X., OHLBERG, D., WU, W., AND WILLIAMS, S. 2008. An integrated nanocrossbar/MOSFET logic circuit: Demonstration of self-programming hardware. In *Proceedings of the Spring Meeting of the Materials Research Society*. ACM, New York, 96–104.

CHAKRABORTY, R. S., PAUL, S. AND BHUNIA, S. 2008. Analysis and robust design of diode-resistor based nanoscale crossbar PLA circuits. In *Proceedings of the 21st International Conference on VLSI Design (VLSID'08)*. IEEE Computer Society, Los Alamitos, CA, 441–446.

CHATTERJEE, A. AND ROY, K. 2003. Performance estimation of molecular crossbar architecture considering capacitive and inductive coupling between interconnects. In *Proceedings of the 3rd IEEE Conference on Nanotechnology*. 445–448.

CHEN, Y., OHLBERG, D. A., LI, X., STEWART, D. R., WILLIAMS, R. S., JEPPESEN, J. O., NIELSEN, K. A., STODDART, J. F., OLYNICK, D. L., AND ANDERSON, E. 2003. Nanoscale molecular-switch devices fabricated by imprint lithography. *Appl. Phys. Lett. 82*, 10, 1601–1612.

CORTADELLA, J., KISHINEVSKY, M., KONDRATYEV, A., LAVAGNO, L., AND YAKOVLEV, A. 2002. *Logic Synthesis of Asynchronous Controllers and Interfaces*. Springer Series in Advanced Microelectronics.

DEAN, M., DILL, D. L., AND HOROWITZ, M. 1991. Self-timed logic using current-sensing completion detection. In *Proceedings of the IEEE International Conference on Computer Design*. 187–191.

DEHON, A. 2003. Array-based architecture for FET-based, nanoscale electronics. *IEEE Trans. Nanotech*. 2, 1, 23–32.

GOLDSTEIN, S. C. AND BUDIU, M. 2001. NanoFabrics: Spatial computing using molecular electronics. In *Proceedings of the 28th Annual International Symposium on Computer Architecture*. ACM, New York, 178–189.

He, C., Jacome, M. F., and de Veciana, G. 2005. Scalable defect mapping and configuration of memory-based nanofabrics. In *Proceedings of the 10th Annual IEEE International High-Level Design Validation and Test Workshop* (HLDVT'05). IEEE Computer Society, Los Alamitos, CA, 11–18.

Hogg, T. and Snider, G. S. 2007. Defect-tolerant logic with nanoscale crossbar circuit. *J. Electron. Test. Theor. Appl. 23*, 2–3, 117–129.

Huang, J., Tahoori, M. B., and Lombardi, F. 2004. On the defect tolerance of nano-scale two-dimensional crossbars. In *Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*. ACM, New York, 96–104.

Huang, Y., Duan, X., Cui, Y., Lauhon, L. J., Kim. K., and Lieber, C. M. 2001. Logic gates and computation from assembled nanowire building blocks. *Science 294*, 2, 1313–1317.

ITRS. 2005. International technology roadmap for semiconductors (ITRS). http://public.itrs.net.

Josephs, M. B., Nowick, S. M., and van Berkel, C. H. 1999. Modeling and design of asynchronous circuits. *Proc. IEEE 87*, 2, 234–242.

Karypis, G., Aggarwal, R., Kumar, V., and Shekhar, S. 1999. Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Trans. VLSI 7*, 1, 69–79.

Kuekes, P. J., Stewart, D. R., and Williams, R. S. 2005. The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits. *J. Appl. Phys*. 97, 3, 034301–034301-5.

Liu, X., Lee, C., and Zhou, C. 2001. Carbon nanotube field-effect inverters. *Appl. Phys. Lett.* 79, 20, 3329–3331.

Melosh, N. A., Boukai, A., Diana, F., Gerardot, B., Badolato, A., Petroff, P. M., and Heath, J. R. 1998. Ultrahigh-density nanowire lattices and circuits. *Science 280*, 5, 1716–1721.

Naeoscale Integration and Modeling Group. 2007. The predictive technology model. Anzona State University. http://www.eas.asu.edu/ptm/.

Nikolic, K., Sadek, A. and Forshaw, M. 2002. Fault-tolerant techniques for nanocomputers. *Nanotech. 13*, 2–3, 357–362.

Rabaey, J. M., Chandrakasan, A. and Nikolic, B. 2003. *Digital Integrated Circuits: A Design Perspective* (2nd Edition). Prentice Hall, Reading, MA.

Rudell, R. L. and Sangiovanni-Vincentelli, A. 1987. Multiple-valued minimization for PLA optimization. *IEEE Trans. CAD Integr. Circ. Syst. 6*, 727–750.

Snider, G. S. and Williams, R. S. 2007. Nano/CMOS architecture using a field programmable nanowire interconnect. *Nanotech. 18*.

Snider, G. S., Kuekes, P., and Williams, R. S. 2004. CMOS-like logic in defective, nanoscale crossbar. *Nanotech. 15*, 8, 881–891.

Snider, G., Kuekes, P., Hogg, T., and Williams, R. S. 2005. Nanoelectronic architectures. *Appl. Phys. A 80*, 6, 1183–1195.

Sparsø, J. and Furber, S. 2001. *Principles of Asynchronous Circuit Design—A Systems Perspective*. Kluwer Academic Publishers, Reading, MA.

Strukov, D. B. and Likharev, K. 2007. CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. *Nanotech. 16*, 888–900.

Sutherland, I. E. 1989. Micropipelines. *Comm. ACM 32*, 6, 720–738.

Tour, J. M., Van Zandt, W. L., Husband, C. P., Husband, S. M., Wilson L. S., Franzon, P. D., and Nackashi, D. P. 2002. Nanocell logic gates for molecular computing. *IEEE Trans. Nanotech. 1*, 2, 100–109.

Zieglar, M. M. and Stan, M. R. 2003. CMOS/Nano co-design for crossbar-based molecular electronic systems, *IEEE Trans. Nanotech. 2*, 4, 217–229.

Received June 2008; revised October 2008; accepted November 2008 by Luca Benini