Appearing in *IEEE International Conference on Field-Programmable Technology* (FPT 2009), December 9–11, 2009

VMATCH: Using Logical Variation to Counteract Physical Variation in Bottom-Up, Nanoscale Systems

Benjamin Gojman [#1], André DeHon [*2]

[#] *Computer and Information Systems, University of Pennsylvania*
3330 Walnut Street, Philadelphia, PA, USA 19104
[1] bgojman@seas.upenn.edu

[*] *Electrical and Systems Engineering, University of Pennsylvania*
200 S. 33rd Street, Philadelphia, PA, USA 19104
[2] andre@ieee.org

***Abstract*—Nanowire building blocks provide a promising path to small feature size and thus the ability to more densely pack logic. However, the small feature size and novel, bottom-up manufacturing process will exhibit extreme variation and produce many devices that operate outside acceptable operating ranges. One-mapping-fits-all, prefabrication assignment of logical functions to physical transistors that exhibit high threshold variation will not work—combining the wide range of physical variation in transistor threshold voltage with the wide range of fanouts in the design produces an unworkably large composite range of possible delays. Nonetheless, by carefully matching the fanout of each net to the physical threshold voltages of devices after fabrication, it is possible to reduce the net range of path delays sufficiently to achieve high system yield. By adding a modest amount of extra resources, we achieve 100% yield for systems built out of devices with 38% variation, the ITRS prediction for threshold variation in 5 nm transistors. Moreover, for these systems, we maintain delay, energy and area close to the variation-free nominal case.**

I. Introduction

As device feature sizes scale below optical wave length scales, manufacturing reliable systems using lithographic technologies is increasingly challenging. As a consequence, researchers have been exploring bottom-up manufacturing methods that avoid lithography for defining the smallest feature size. Though still in its infancy, one such technology is catalyst-grown nanowires. Researchers have demonstrated components built out of nanowires with diode and FET-like behaviors [1], [2], [3]. Others have proposed how to build integrated reconfigurable systems using these components [4], [5], [6]. While encouraging, this bottom-up technology is not without its challenges; high among them is extreme levels of random variation in the nanoscale components.

Variation in these systems comes both from the independent manufacturing of each component and the stochastic assembly process this technology requires. Components are built out of individually grown wires, and although scientist have demonstrated impressive control of this growth process [7], [8], atomic-scale dimensions mean that small differences among wires manifests as greatly varying component characteristics.

Due to threshold voltage variation of 5 nm length transistors, transistor on current (I_{on}) will range an order of magnitude above and one below its nominal value, and transistor off current (I_{off}) will range five orders of magnitude below and five above its nominal value. Unmitigated, this variation will produce highly defective, "inherently irreproducible" [6] devices, and both fixed and programmable systems built out of nanowires will be inoperable.

We present VMATCH, an algorithm that takes advantage of post-fabrication characterization of devices along with the reconfigurable nature of the NanoPLA, to use highly varying devices more effectively. It successfully maps designs by exploiting the *fanout-variation* introduced by the architecture and logical netlist to counteract *physical variation* of the threshold voltage, V_{th}, in the transistors. We show that our algorithm solves the problem of mapping to systems with extreme variation while maintaining yield, performance, energy and area close to variation-free systems. This leads to reproducible systems built out of irreproducible devices, resulting in a more efficient variant of Von Neumann's vision of reliable systems built out of unreliable components [9].

Sec. II introduces our variant on the NanoPLA from [4] as well as its sources of variation. We then present the operating model for our NanoPLA (Sec. III) and motivate and introduce VMATCH, our algorithm to mitigate the negative effects of variation in Sec. IV. Experimental setup and results are shown in Sec. V. We conclude in Sec. VI.

The novel contributions of this work are:

- Introduction of VMATCH, a post-fabrication mapping algorithm that matches the fanout of logical nets with physical transistor threshold voltages to effectively exploit nanoscale transistors with extreme V_{th} variation.
- Quantification for the Toronto 20 benchmark set [10] of the impact of: (a) ignoring variation, (b) treating variation as defects, and (c) using VMATCH to mitigate variation.

II. Background

To appreciate the sources of variation, we first review the technology and architecture of the NanoPLA.

A. Technology: Nanowires

Nanowires are the main building block of the NanoPLA. These can be grown out of many different materials including doped Si [7], GaAS, GaN [11], and Au [12]. These wires can be microns long [13] and their diameters can be precisely controlled using seed catalysts [7]. Moreover, during the growth process the doping of the nanowire can be varied along its length [14], [15] allowing components such as field-effect transistors to be embedded in the wire. Finally, insulating core shells can be radially grown over the entire length of the wire creating a separation between conducting wires as well as between gate and control wires in a FET [16], [17].

Due to their small features and limited assembly techniques, regular structures are easier to build out of these components than arbitrary topologies. Langmuir-Blodgett (LB) flow techniques are used to align nanowires into large-scale parallel arrays [18], [19]. By using nanowires with insulating shells, the LB technique can tightly pack nanowires without shorting them. These shells can later be selectively etched away [19]. When repeated, this process allows for two orthogonal layers to form a densely packed nanowire crossbar [18], [20].

Furthermore, chemist have demonstrated a number of techniques for placing hysteretic switches into the crosspoints between orthogonal nanowire layers. These include layers of bi-stable molecules [21], [22], amorphous silicon nanowire coatings [23], and nanowires made of switchable species [1]. Some of these programmable switches have diode-like rectification, an essential property for correct operation.

B. Architecture: NanoPLA

The NanoPLA is organized as shown in Fig. 1a. It consists of tiled logic blocks with overlapping nanowires that enable Manhattan routing while maintaining direct nanoscale-density interconnect among blocks. It is a modification on DeHon's [4] nanoPLA blocks and uses amorphous Si switches [23] to improve performance and energy.

The NanoPLA block is composed of three logic stages. As in a conventional PLA, the first stage or input stage is used to selectively invert the inputs ➊. Stage two and three behave like the AND ➋ and OR ➌ planes respectively. Adding the initial inverting phase allows us to avoid the need for non-inverting restoration as used in [4] and thus improve the overall performance and reduce the total energy used.

Fig. 1b shows a detailed view of a NanoPLA block. Using the bottom-up assembly discussed above, small diameter nanowires are arranged into tight-pitch parallel arrays. Though logically each plane performs a different function (Invert, AND and OR) physically all three planes are identical and are made up of a diode-programmable, wired-OR stage built using the switches previously described, followed by an inversion stage where lightly doped regions of the nanowire behave like field-effect gates and provide restoration. During assembly, etching is used to differentiate the three stages. Decoders built into the nanowires (See Fig. 10a) are used to program the diode-like switches. They are built as described in [4] and demonstrated in [15].

The NanoPLA is similar to conventional FPGAs. Both use Manhattan routing to connect discrete clusters of logic. However, routing in the NanoPLA is done through the blocks rather than using an independent switching network. In order to allow signal routing, the output of the OR-plane of every block connects to itself and four neighboring blocks.

C. Source of Variation

Unlike today's technology where region-based and systematic variation dominate, in the NanoPLA random variation dominates due to the bottom-up manufacturing process. Along with the variation that affects even today's technology (*e.g.* [24], [25], [26], [27]), the NanoPLA faces additional sources of *random* variation.

- Nanowire geometries and features (*e.g.* length of doped regions, core shell thickness) will vary independently since each nanowire will be individually fabricated.
- Statistical alignment techniques [28] during assembly cause the geometry of the field-effect regions to vary from device to device [29].
- Each programmable diode region will be composed of a small number of elements or bonds, giving them large, random variation from crosspoint to crosspoint.

These sources of variation manifest as differences in the nanowire resistances and capacitances, the diode resistances, and the threshold voltages (V_{th}) for the field-effect restore nanowires. Note that [27] calculates that the 5 nm long transistors we are considering are nearly impossible to manufacture reproducibly. We assume independent Gaussian distributions for these values (*e.g.* [27], [26], [24], [25]).

$$P(x) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right) e^{\left(-\frac{(x-\overline{x})^2}{2\sigma^2}\right)} \tag{1}$$

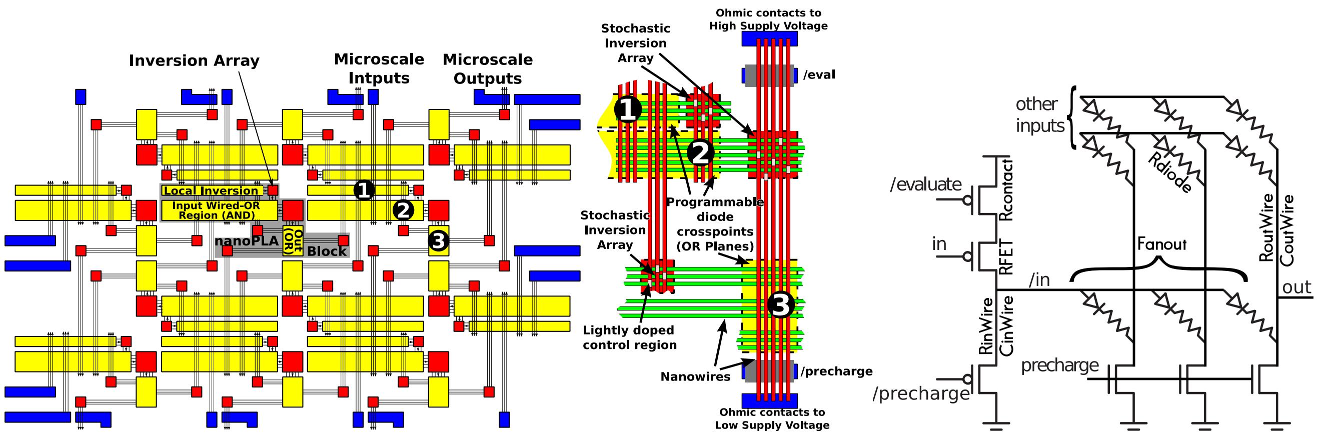
Throughout this paper we express the amount of variation as a percentage equal to σ/μ; we will refer to this simply as σ. Though other works also report variation as a percent it is worth noting that many, including the ITRS [30], tend to report 3σ variation while we label our variation points by σ. Hence our $\sigma = 38\%$ cases corresponds to the $3\sigma = 112\%$ cases ITRS predicts for 5 nm physical gate lengths (13 nm half-pitch technology) as shown in the DESN9b table in [30].

III. System Model

A. Evaluation Model

The NAND-*term* is the smallest unit of computation of a plane in the NanoPLA. Physically it is composed of a set of inverting, restoring wires followed by a wired-OR section thus computing INVERT-OR or, by DeMorgan's laws, NAND. Fig.1c shows an equivalent circuit-level diagram of a NAND-term in a plane of the NanoPLA block. Each plane is composed of many of these NAND-terms together in parallel.

Within each plane, computation is done in a precharge fashion by first pre-discharging the nanowires and then evaluating the inputs. Since each block is composed of three planes, the evaluation scheme demands that we use a three-phased



(a) NanoPLA Block Tiling (b) NanoPLA Block (c) NanoPLA NAND-term Circuit

Fig. 1: NanoPLA Organization

clock to sequence logic in the NanoPLA. At the level of the PLA block, one clock cycle is defined as the time to evaluate all three planes once, $\tau_{cycle} = \tau_{phase_1} + \tau_{phase_2} + \tau_{phase_3}$. Since interconnect is routed through the NanoPLA blocks, it is effectively pipelined (*e.g.* [31]), allowing for high throughput.

B. Defect Model

The time it takes for a plane in the NanoPLA block to switch during the evaluate phase, τ_{switch}, is defined as the time it takes the slowest *used* NAND-term to switch. Similarly the precharge leak time, τ_{leak}, is the time it takes the leakiest *used* NAND-term to lose its perchared value. As the NanoPLA is pipelined to the level of a plane, we can bound permissible phase times by the slowest plane and worst-case leakage by the fastest leaking plane:

$$\max_{planes}(\tau_{switch}) \le \tau_{phase} \le \min_{planes}(\tau_{leak})$$

To provide adequate noise margins we demand at least two orders of magnitude separation between the worst case τ_{switch} and τ_{leak}. This guarantees leakage will charge the output to less than 1% of V_{dd} and therefore leakage current will be less than 1% of drive current across all blocks for a functional NanoPLA. We can state this constraint as:

$$100 \cdot \max_{planes}(\tau_{switch}) \le \min_{planes}(\tau_{leak}) \qquad (2)$$

If a NanoPLA does not meet this constraint, the NanoPLA does not yield and is called defective. In other words, *to compute correctly, all planes must hold charge long enough to allow all computations to complete.*

C. Timing Model

We use the following Elmore Delay models as a conservative estimate of NAND-term switching and leakage:

$$\begin{aligned}\tau_{switch} &= (R_{contact} + R_{onFET} + 0.5R_{inWire}) \\ &\quad \times (C_{inWire} + \sum_{fanout} C_{outWire}) \\ &\quad + (R_{diode} + 0.5R_{outWire}) \cdot C_{outWire}\end{aligned} \qquad (3)$$

$$\begin{aligned}\tau_{leak} &= (R_{contact} + R_{offFET} + 0.5R_{inWire}) \\ &\quad \times (C_{inWire} + \sum_{fanout} C_{outWire}) \\ &\quad + (R_{diode} + 0.5R_{outWire}) \cdot C_{outWire}\end{aligned} \qquad (4)$$

Each term in the equation maps to a physical section of the NAND-term as shown in Fig. 1c. Since the input wire may be connected to many outputs, we include the effect of this fanout as the sum of the downstream capacitance, $\sum_{fanout} C_{outWire}$.

Variation of the resistances and capacitances of the wires and diodes are directly modeled as Gaussian distributions (Eq. 1). Also modeled as a Gaussian distribution is V_{th} variation which is used in Eqs. 5 and 6 to calculate the variation of the on and off resistance of the transistor, R_{onFET} and R_{offFET}. Since the dominate variation is random (Sec. II-C), we assume independent distributions in this paper.

D. NanoPLA CAD Flow

Here we review how logic is mapped on the NanoPLA. Covering and clustering [32] is followed by a block-level placement computed using VPR 4.3 [33]. Global routing and detailed placement and routing are done by our custom NanoPLA place and route tool, NPR. The architecture of the NanoPLA does not provide a separate switching network but rather uses the connections provided by the blocks themselves to perform routing. Conventional FPGA routing algorithms such as Pathfinder [34] perform this block-level routing or *global route*. This determines *MinC*, the minimum channel width necessary for the design to route. Each block has logic functions assigned to it by VPR's placement and *route-throughs* defining what nets route through the block, computed by the global route. Detailed place and route then performs the final mapping. It first decomposes the functions and route-throughs assigned to each block into three sets of logical NAND-terms, one for each of the three planes in the NanoPLA block. Then, one plane at a time, each logical NAND-term is mapped to a physical NAND-term. Without post-fabrication knowledge, however, the mapper is unable to distinguish

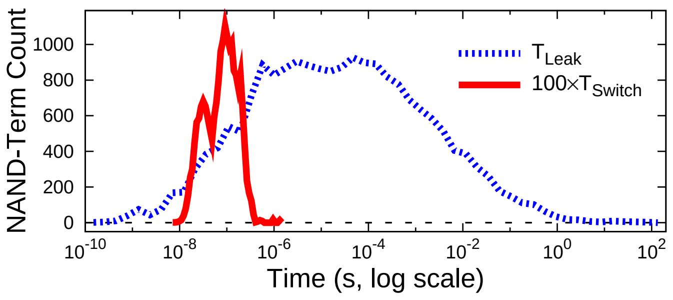


Fig. 2: Distribution of τ_{leak} and $100 \times \tau_{switch}$ of a delay oblivious-mapping. Benchmark spla at $\sigma = 38\%$

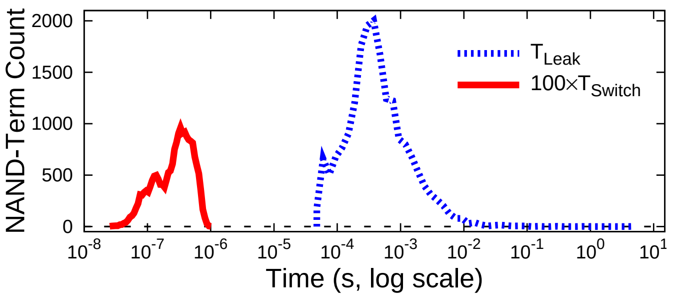


Fig. 3: Distribution of τ_{leak} and $100 \times \tau_{switch}$ of a Defect-Avoidance mapping. Benchmark spla at $\sigma = 38\%$

between physical NAND-terms and must treat them all as having identical characteristics when performing the mapping. It produces a single mapping that is applied obliviously to all chips. The next section explains why this variation-oblivious mapping produces defective chips and introduces a solution.

IV. Device Specific Mapping

In this section we illustrate why the mapper must consider the physical variation (Sec. IV-A). We examine how variation affects τ_{switch} and τ_{leak} (Sec. IV-B) and introduce a naive solution that satisfied Eq. 2 but at a high cost. (Sec. IV-C). Finally we introduce VMATCH, our algorithm that considers the effects the mapping has on τ_{switch} and τ_{leak} (Sec. IV-E).

A. *Variation-Oblivious Mapper*

At high levels of variation, the distribution of τ_{switch} and τ_{leak} is such that, when mapping a design oblivious to the variation in the system, the probability of meeting the constraint set by Eq. 2 is almost zero. Fig. 2 shows the distribution of $100 \times \tau_{switch}$ and of τ_{leak} that results from such an oblivious mapping. Since the curves overlap, it is immediately apparent that Eq. 2 does not hold.

B. *Primary Sources of Variation*

Before exploring how to modify the mapping algorithm, we first look at which sources of variation in τ_{switch} and τ_{leak} are primarily responsible for this yield problem. From Eq. 2 we observe that, for a particular NAND-term to be defect free it must be the case that $100 \cdot \tau_{switch} \leq \tau_{leak}$. Since the only difference between τ_{switch} and τ_{leak} is the state of the transistor being on and off respectively (see Eq. 3 and 4), for correct operation R_{offFET} must be the dominant term in τ_{leak}. If this were not the case and one of the other terms in Eq. 4 dominated, there would be nearly no difference between τ_{switch} and τ_{leak} and, as such, correct operation would be impossible regardless of how the design is mapped.

The difference between R_{offFET} and R_{onFET} comes from the fact that R_{offFET} is the apparent resistance of the transistor in the sub-threshold region or $R_{offFET} = V_{dd}/I_{sub}$ (Eq. 6). In the on state, the transistor operates in saturation, and we define the value of R_{onFET} as V_{dd}/I_{sat} (Eq. 5). Since the nanowires are still Silicon, we use short-channel MOSFET current equations [35], [36]:

$$I_{sat} = W v_{sat} C_{ox} \left(V_{gs} - V_{th} - 0.5 \cdot V_{d,sat}\right) \quad (5)$$

$$I_{sub} = \frac{W}{L} \mu C_{ox} (n-1) \cdot v_T{}^2 e^{\dfrac{V_{gs}-V_{th}}{n v_T}} \left(1 - e^{-V_{ds} \cdot v_T{}^{-1}}\right) \quad (6)$$

We see that saturation current is *linear* in V_{th} and V_{dd} and that sub-threshold current is *exponential* in V_{th} and V_{dd}. Thus a small change due to the variation in V_{th} will cause a *linear change* in the value of R_{onFET} and an *exponential change* in the value of R_{offFET}. Consider that, at $V_{th} = 295$mV and $V_{dd} = 0.7$V, the mean value for R_{onFET} is $51k\Omega$ and for R_{offFET} is $30G\Omega$. At $\sigma = 38\%$, the 3σ V_{th} variation point gives a range for R_{onFET} from $2.8 \times 10^4\Omega$ to $3.2 \times 10^5\Omega$. For R_{offFET} the range is from $4.7 \times 10^5\Omega$ to $1.9 \times 10^{15}\Omega$. While the minimum R_{offFET} value is larger than R_{onFET}, they are less than a factor of two apart and hence do not satisfy Eq. 2. Given that all other parameters in Eqs. 3 and 4 vary linearly based on Gaussian distributions, R_{offFET} varies over the greatest range and therefore is the dominating variation in the system. A successful mapping algorithm must first focus on reducing the range over which R_{offFET} varies to create the separation required by Eq. 2.

C. *Defect-Avoiding Algorithm*

The oblivious algorithm fails because it uses NAND-terms that leak faster than some resources can switch. The Defect-Avoiding algorithm tries to solve this problem by not using the leakiest resources, essentially marking them as defective. Mapping to the remaining resources is arbitrary. The idea of mapping around defective resources has been well studied by many, including [5], [37], [38], and is generally accepted as necessary for nanoscale systems.

A nanowire is marked defective if its off resistance is too low. We determine a conservative threshold for this resistance using Eq. 4 and assuming the wire is driving a single, variation-free output nanowire (*i.e.* fanout of one). Additional fanout will only increase τ_{leak}, so the fanout one case serves as the worst-case possible assignment.

Fig. 3 shows the result of mapping the same chip shown in Fig. 2. Though the separation between τ_{switch} and τ_{leak} is great, this mapping required 167% extra resources above MinC and marked 48% of all NAND-terms as defective; that is, it discards the fraction of the τ_{leak} distribution (Fig. 2) that is below 6×10^{-5}s. In Sec. V we show that this defect-avoidance algorithm, on average, needs 193% more resources than the variation-free case.

D. *Logical Variation: Variation in Fanout*

Though the defect-avoiding algorithm works, it is too conservative and thus loses some of the scaling benefits this sub-lithographic technology affords. A review of Eq. 4, however, shows that physical variation is not the only variation that determines the range of the τ_{leak} distribution. Along with the physical parameters, there is a fanout parameter whose value comes directly from the logical netlist and varies over a significant range. Fanout in the NanoPLA comes from the fact that a NAND-term has non-restoring, diode-like connections (Fig. 1c). If a signal on an input wire is needed by multiple output wires, the input wire must have the associated diodes programmed to connect to the required output wires, and it must charge up all connected wires. Consider an example: When mapping the logical function $A\overline{B}+AC\overline{D}+B\overline{E}+AF$ to a block in the NanoPLA, three terms in the AND-plane will use input signal A ($A\overline{B}$, $AC\overline{D}$, and AF), while signal F is only used once by AF. Even without physical variation, this means that signal A's NAND-term will see three times the $C_{outWire}$ capacitance that F's will.

The maximum fanout of a NAND-term is determined by the architecture of the NanoPLA. Each PLA in an array of PLAs, like the NanoPLA, will have a maximum number of inputs, AND-terms and outputs. This will have a direct effect on the number of output wires each input wire can potentially connect to, and consequently, the maximum fanout a NAND-term can have. For our mappings, we use PLAs with at most 64 AND-terms and 16 inputs that may need inversion; as shown in Fig. 1a routing nanowires are exposed to two AND-planes and two inversion planes. This means the worst-case fanout for a nanowire is $(16+64)\times 2 = 160$. In practice, the maximum fanout is lower. Fig. 4 shows a typical distribution with a maximum fanout of 34. While there are a few high fanout nets, note that most of the nets have fanout one. Mapped obliviously, this adds another two orders of magnitude to the range of the τ_{leak} distribution; this makes fanout the second-most significant source of variation in Eqs. 3 and 4. In the next section we explain how we use this logical variation to counteract the physical variation of R_{offFET} to map designs that maintain acceptable performance, energy and area.

We could architect smaller arrays with fewer AND-terms to reduce the fanout but only at the expense of increasing the total energy, area, and evaluation latency. While smaller arrays can reduce the clock cycle (τ_{cycle}), they increase the number of blocks in a logical evaluation path. Our design point with 64-AND-term arrays was chosen so the overall evaluation time and area were both close to minimum across the array shape parameter space.

E. *VMATCH: NanoPLA Mapping Algorithm*

VMATCH is our variation-aware mapping algorithm, it takes advantage of the fanout variation to counteract the variation in R_{offFET} by carefully matching a high-fanout term with a low R_{offFET} NAND-term and vice versa, achieving a mapping that yields while maintaining performance, energy and area close to the variation-free case. We can understand

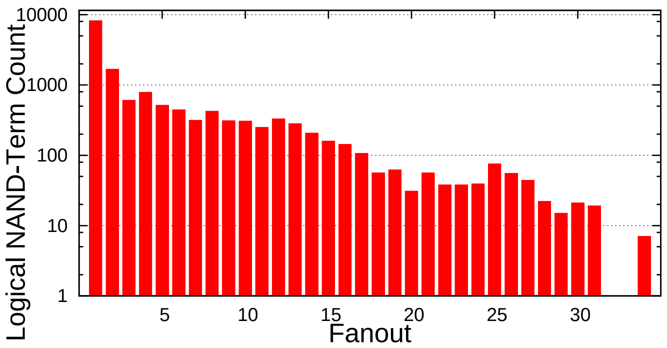


Fig. 4: Fanout distribution. Benchmark spla

why this works by examining how the τ_{leak} distribution changes based on how each of the three algorithms uses the R_{offFET} and fanout variation. In the variation-oblivious mapping, the two orders of magnitude fanout variation (Fig. 4) essentially gets multiplied by the ten orders of magnitude of R_{offFET} variation leading to the twelve orders of magnitude range of τ_{leak} in Fig. 2. The defect avoiding algorithm limits τ_{leak}'s range by directly limiting the range of R_{offFET} values used, but this must discard almost half of the resources. VMATCH, on the other hand, is able to **divide** the magnitude of physical R_{offFET} variation by that of the logical fanout variation, *reducing* the total range of τ_{leak} while using many of the resources the defect avoiding algorithm discarded.

1) Threshold Measurement: To perform this variation-aware post-fabrication mapping it is necessary to measure the nanowire transistor threshold voltages. Appendices A through C sketches how these measurements could be made. Nonethelss, the primary focus of this paper is to characterize the benefits these measurements provide. As the variation-oblivious mapping illustrates, this information is required for correct operation.

2) Algorithm: In order to reduce $\max(\tau_{switch})$ and maintain performance, we map the slowest (highest fanout) logical NAND-term to the fastest (lowest R_{offFET}) physical NAND-term and use the resulting τ_{leak} as a minimum target, τ_{target}, that all other mappings try to match.

Alg. 1 shows VMATCH in detail. First all logical NAND-terms are sorted by fanout in a descending priority queue. Then τ_{target} is calculated as described above. Each logical function in the queue is mapped to the physical NAND-term with the lowest R_{offFET} that does not cause it to have a τ_{leak} lower than τ_{target}. If a mapping cannot find a resource that meets this target, it picks the physical NAND-term that is closest to the target and updates τ_{target} to be that NAND-term's τ_{leak}.

Fig. 5 shows the results of using this algorithm to map the same chip shown in Fig. 2. Clearly Eq. 2 holds and the mapping is defect free. However, unlike the conservative mapping (Fig. 3), this mapping achieves a lower $\max(\tau_{switch})$ and only requires 3% extra resources over MinC.

To build intuition for this success, Fig. 6 presents the distribution of the used R_{offFET}s for each of the three algorithms. For the oblivious algorithm, the distribution spans a very wide range as previously noted (Fig. 2). The defect-avoiding

```
foreach  $Function\ L \in \{logic\}$ do
    // Descending fanout
    $PriorityQueue\ LQ.add(L)$
end
$Target := -1$
while $LQ.hasNext$ do
    $L := LQ.pop()$
    $P := L.physicalPlane$
    if $Target = -1$ then
        $NandTerm\ NT := P.NextMinRoff()$
        $NT.program(L)$
        $NT.commit(L)$
        $Target := NT.Tleak()$
    else
        repeat
            $NandTerm\ NT := P.NextMinRoff()$
            $NT.program(L)$
        until $(NT.Tleak() \geq Target\ ||\ lastNT(NT))$
        $NT.commit(L)$
        if $(lastNT(NT)\ \&\&\ NT.Tleak() < Target)$
        then
            $Target := NT.Tleak()$
        end
    end
end
```

Algorithm 1: VMATCH

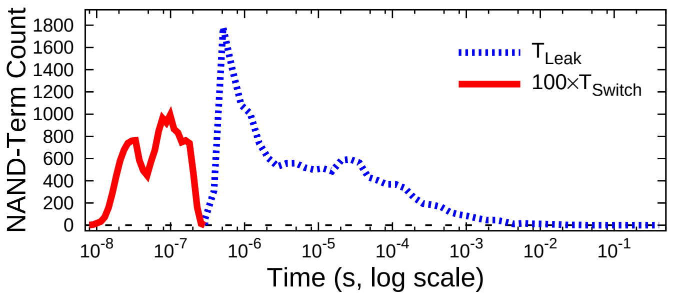


Fig. 5: Distribution of τ_{leak} and $100 \times \tau_{switch}$ of VMATCH mapping. Benchmark spla at $\sigma = 38\%$

algorithm chooses resources that are above the conservative threshold. Fig. 3 shows that this is too cautious and forces the use of slower resources. The distribution of transistors used by VMATCH includes most of the resources that the defect-avoiding algorithm found to be too leaky. By correctly pairing fast resources with high fanout nets, we are able to use faster resources as Figs. 5 and 6 demonstrate.

V. Results

A. Experimental Setup

We simulated a NanoPLA using 5 nm pitch wires with crosspoints implemented in Amorphous-Si technology [23] and transistors with 5 nm channel lengths and 295mV nominal V_{th}. We ran our simulation using 0.7V V_{dd} which produced an effective mean on and off transistor resistance of $51k\Omega$ and $30G\Omega$ respectively. Mean wire resistance and capacitance vary based on design and NanoPLA route channel width but are of the order of $50k\Omega$ and $45fF$ for the input wires, and $1M\Omega$ and $50fF$ for the output wires. Mean crosspoint on resistance, R_{diode}, is $100k\Omega$ and microscale contact resistance, $R_{contact}$, is $10k\Omega$.

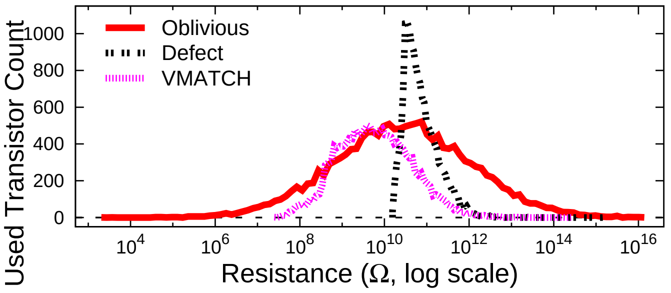


Fig. 6: Used R_{offFET} distribution for Oblivious, Defect and VMATCH algorithms. Benchmark spla at $\sigma = 38\%$

We implemented VMATCH as well as the Defect-Avoiding (*Defect*) and Variation-Oblivious (*Oblivious*) algorithms in our custom NanoPLA router and used them to characterize the benefits VMATCH provides. We routed the Toronto 20 benchmark set [10] on 100 Monte Carlo generated chips; this means we can be 90% confident that the results estimated as 100% yield at least exceed 97.5% yield. Variation on all components was modeled as independent Gaussian distributions as previously defined in Sec. II-C.

B. Achievable Yield

Fig. 7 shows how yield drops as a function of percent variation in the system. We explored the effect of providing extra resources by routing on both the minimum channel width (*MinC*), calculated by the global route, and 30% extra channels on top of MinC (*MinC + 30%*). Yield is based on Eq. 2. The Oblivious algorithm achieves near 100% yield up to 9% variation but drops to no yield by 15% variation. This is true even with extra channels since the Oblivious case makes a fixed mapping to channels that is not affected by the actual characteristics of each wire; the chance of selecting unusable wires is the same even when selecting from a larger resource pool (*i.e.*, more channels). The Defect case does not yield because it discards too many NAND-terms as defective and is unable to fit the design on the remaining resources. It can yield with additional resources as noted in the following section.

The last two curves are for VMATCH at minimum channel with and 30% extra channels. Both maintain 100% yield well past the point where Oblivious fails. For the case with extra channels, because the algorithm carefully chooses what resources to use it can take advantage of the modest increase in channel width and maintain 100% yield up to 35% variation, and it stays above 90% yield at extreme variation.

The first section of Tab. I reports the highest percent variation that achieves 100% yield for both Oblivious and VMATCH assignment at 30% extra channels. On average the Oblivious router maintains 100% yield up to 10% variation compared to VMATCH at 30%.

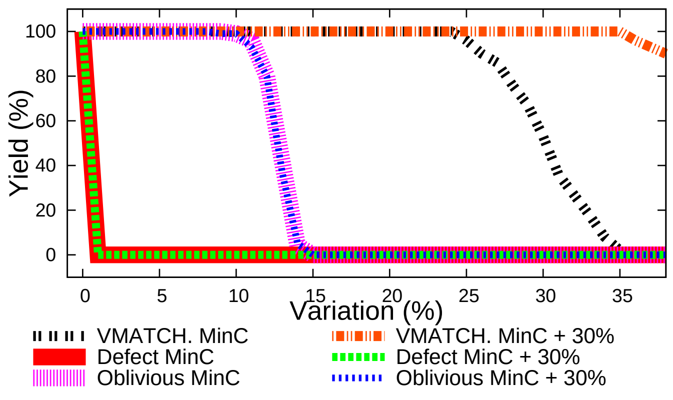


Fig. 7: Yield curves for Oblivious, Defect and VMATCH mapping. Benchmark spla 100 chips

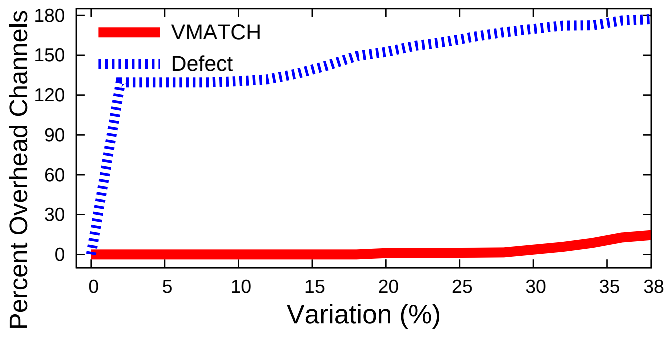


Fig. 8: Average minimum number of channels required to get 100% yield. Benchmark spla 100 chips

C. *Delay, Energy and Area*

Both VMATCH and Defect routing can exploit extra resources; as noted above, Oblivious cannot. In this section we explore the effects extra resources have on delay, energy and area only for the Defect and VMATCH cases.

Adding extra resources, increases the probability that, after the Defect algorithm marks the defective resources, there are enough acceptable resources left to map the design. For VMATCH, extra resources allow it to choose more of the resources that are faster and improve overall delay. For both cases it increases the probability of the design yielding. The disadvantage of extra resources is that, a wider route channel forces longer wires; this increases wire resistance and capacitance, increasing nominal wire delay, total energy, and total area. Therefore, the goal is to use the minimum number of extra channels so that the design yields.

For the 100 Monte Carlo experiments, Fig. 8 shows the average of the minimum number of extra channels, as a percent of the minimum channel width, needed to make the chip yield the design. At low variation VMATCH needs no extra channels, even at 38% variation it only requires 15% extra channels. In contrast, Defect needs 130% extra at very low variation and it only increases.

Fig. 9 shows how these channel widths (Fig. 8) affect the total delay, energy and area. The curves shown are a ratio to the variation free *Nominal* case. Even at 38% variation VMATCH is within 70% of the Nominal delay and within 20% of the

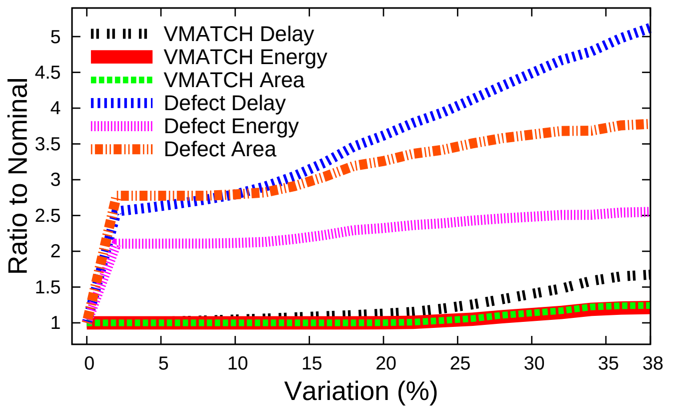


Fig. 9: Ratio to variation free Nominal of delay, energy and area at 100% yield. Benchmark spla 100 chips

Nominal energy and area. Defect is a factor of 5.1 slower, 3.8 larger and uses 2.6 times as much energy as Nominal.

The right side of Tab. I shows the mean and standard deviation for delay, energy and area ratios as well as number of extra channels required for the benchmark set to achieve 100% yield at 38% variation. On average, VMATCH uses 24% more channels than Nominal, and aggregate characteristics stays within 90% of Nominal for delay, 30% for energy and 40% for area, while multiple factors over Nominal are needed for Defect to work. By matching the fanout of the logical NAND-term to the R_{offFET} variation of the physical NAND-term, VMATCH is able to produce a mapping that not only yields but is close to the efficiency of the variation-free case.

VI. Conclusion

We introduced VMATCH, an algorithm for the NanoPLA that can successfully deal with extreme variation. By matching the dominant physical variation to the logical fanout variation we get high yield where an oblivious mapping fails. We can trade a modest amount of extra resources to get performance, energy and area close to what a variation-free device could achieve. This shows that “nansocale field-effect transistors which are inherently irreproducible” [6] (footnote 7) need not prevent the construction of field-programmable components that deliver reproducible design mappings with reasonable energy, delay, and area metrics. Our results also show that, for variation above 10%, component-specific mapping is required to obtain acceptable yield levels.

VII. Acknowledgments

This research was funded in part by National Science Foundation grant CCF-0726602 and CCF-0904577. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Nikil Mehta provided assistance and code for device modeling.

References

[1] Z. Fan, X. Mo, C. Lou, Y. Yao, D. Wang, G. Chen, and J. G. Lu, “Structures and electrical properties for Ag-tetracyanoquinodimetheane organometallic nanowires,” *IEEE Trans. Nanotechnol.*, vol. 4, no. 2, pp. 238–241, March 2005.

	Highest Variation That Achieves 100% Yield at 30% Extra Channels		Defect at 38% σ and 100% Yield								VMATCH at 38% σ and 100% Yield							
			% Extra Channels		Ratio to Nominal						% Extra Channels		Ratio to Nominal					
					Delay		Energy		Area				Delay		Energy		Area	
Net	**Oblivious**	**VMATCH**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**
alu4	10%	32%	174	19	4.3	0.73	2.4	0.16	3.3	0.35	17	12	1.5	0.20	1.2	0.10	1.2	0.12
apex2	11%	25%	192	14	6.5	0.75	2.6	0.13	4.0	0.31	26	21	1.9	0.50	1.3	0.17	1.3	0.29
apex4	11%	32%	192	25	4.5	0.62	2.4	0.19	3.6	0.47	17	13	1.6	0.35	1.2	0.09	1.3	0.13
bigkey	12%	33%	164	14	8.5	1.2	2.5	0.15	4.2	0.44	17	13	2.0	0.47	1.2	0.12	1.3	0.23
clma	10%	29%	183	15	7.5	0.80	2.7	0.15	4.9	0.46	34	5.9	2.1	0.11	1.3	0.04	1.5	0.10
des	9%	30%	178	14	7.7	19	2.6	0.14	4.5	0.41	26	18	1.9	0.51	1.2	0.15	1.4	0.30
diffeq	12%	29%	241	24	7.5	1.3	2.6	0.15	3.5	0.31	23	21	2.0	0.41	1.2	0.13	1.3	0.18
dsip	12%	32%	202	17	9.9	1.3	2.8	0.15	4.3	0.43	28	16	2.3	0.56	1.3	0.13	1.3	0.23
elliptic	10%	29%	162	12	6.7	0.79	2.6	0.14	4.6	0.42	27	26	1.9	0.79	1.3	0.24	1.5	0.51
ex1010	11%	26%	260	24	6.4	0.80	2.8	0.17	4.2	0.41	43	22	2.0	0.39	1.4	0.15	1.4	0.24
ex5p	10%	35%	167	39	4.0	0.87	2.2	0.30	3.2	0.70	6.7	12	1.4	0.14	1.1	0.07	1.2	0.14
frisc	10%	26%	214	12	7.2	0.77	2.8	0.11	4.6	0.31	42	17	2.0	0.39	1.4	0.13	1.5	0.24
misex3	10%	28%	204	27	4.4	0.61	2.6	0.21	4.0	0.56	26	11	1.4	0.21	1.2	0.09	1.4	0.10
pdc	9%	32%	191	12	5.9	0.43	2.7	0.11	4.2	0.31	30	16	1.9	0.3	1.3	0.13	1.4	0.21
s298	9%	28%	165	12	9.8	1.1	2.6	0.13	4.4	0.39	23	32	2.4	1.4	1.3	0.29	1.5	0.62
s38417	8%	29%	184	22	7.3	1.3	2.8	0.24	4.9	0.77	26	5.1	1.9	0.19	1.3	0.04	1.4	0.08
s38584.1	11%	30%	228	13	6.6	0.59	2.7	0.11	4.2	0.24	33	6.4	1.9	0.31	1.3	0.05	1.4	0.07
seq	9%	34%	168	13	5.0	0.74	2.5	0.13	4.1	0.38	24	14	1.6	0.26	1.2	0.12	1.3	0.21
spla	8%	35%	177	21	5.1	0.59	2.6	0.19	3.8	0.45	15	9.0	1.7	0.16	1.2	0.08	1.2	0.10
tseng	12%	29%	260	29	6.8	0.95	2.6	0.18	3.5	0.37	24	11	1.9	0.61	1.2	0.06	1.3	0.08
Mean	10%	30%	193	17	6.4	0.96	2.6	0.18	4.1	0.41	24	17	1.9	0.34	1.3	0.10	1.4	0.20

TABLE I: Tolerable Variation and Overheads Required for Toronto 20 Benchmark Set

[2] J. Brault, M. Saitoh, and T. Hiramoto, "Channel width and length dependence Si nanocrystal memories with ultra-nanoscale channel," *IEEE Trans. Nanotechnol.*, vol. 4, no. 3, pp. 349–354, 2005.

[3] Y. Huang, X. Duan, Y. Cui, L. Lauhon, K. Kim, and C. M. Lieber, "Logic gates and computation from assembled nanowire building blocks," *Science*, vol. 294, pp. 1313–1317, November 9 2001.

[4] A. DeHon, "Nanowire-Based Programmable Architectures," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 1, no. 2, pp. 109–162, 2005.

[5] G. Snider, P. Kuekes, and R. S. Williams, "CMOS-like logic in defective, nanoscale crossbars," *Nanotechnology*, vol. 15, pp. 881–891, June 2004.

[6] D. B. Strukov and K. K. Likahrev, "A reconfigurable architecture for hybrid CMOS/nanodevice circuits," in *FPGA*, 2006, pp. 131–140.

[7] Y. Cui, L. J. Lauhon, M. S. Gudiksen, J. Wang, and C. M. Lieber, "Diameter-controlled synthesis of single crystal silicon nanowires," *Appl. Phys. Let.*, vol. 78, no. 15, pp. 2214–2216, 2001.

[8] N. A. Melosh, A. Boukai, F. Diana, B. Gerardot, A. Badolato, P. M. Petroff, and J. R. Heath, "Ultrahigh-density nanowire lattices and circuits," *Science*, vol. 300, pp. 112–115, April 4 2003.

[9] J. V. Neumann, "Probabilistic logic and the synthesis of reliable organisms from unreliable components," in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton University Press, 1956.

[10] V. Betz and J. Rose, "FPGA Place-and-Route Challenge," <http://www.eecg.toronto.edu/~vaughn/challenge/challenge.html>, 1999.

[11] P. V. Radovanovic, C. J. Barrelet, S. Gradecak, F. Qian, and C. M. Lieber, "General syntehsis of manganese-doped II-VI and III-V semiconductor nanowires," *Nanoletters*, vol. 5, no. 7, pp. 1407–1411, 2005.

[12] C. Wang, Y. Hu, C. M. Lieber, and S. Sun, "Ultrathin Au nanowires and their transport properties," *J. Am. Chem. Soc.*, vol. 130, pp. 8902–8903, 2008.

[13] M. S. Gudiksen, J. Wang, and C. M. Lieber, "Synthetic control of the diameter and length of semiconductor nanowires," *J. of Phys. Chem. B*, vol. 105, pp. 4062–4064, 2001.

[14] M. S. Gudiksen, L. J. Lauhon, J. Wang, D. C. Smith, and C. M. Lieber, "Growth of nanowire superlattice structures for nanoscale photonics and electronics," *Nature*, vol. 415, pp. 617–620, February 7 2002.

[15] C. Yang, Z. Zhong, and C. M. Lieber, "Encoding electronic properties by synthesis of axial modulation-doped silicon nanowires," *Science*, vol. 310, pp. 1304–1307, November 25 2005.

[16] L. J. Lauhon, M. S. Gudiksen, D. Wang, and C. M. Lieber, "Epitaxial core-shell and core-multi-shell nanowire heterostructures," *Nature*, vol. 420, pp. 57–61, 2002.

[17] M. Law, J. Goldberger, and P. Yang, "Semiconductor nanowires and nanotubes," *Annual Review Material Science*, vol. 34, pp. 83–122, August 2004.

[18] Y. Huang, X. Duan, Q. Wei, and C. M. Lieber, "Directed assembly of one-dimensional nanostructures into functional networks," *Science*, vol. 291, pp. 630–633, January 26 2001.

[19] D. Whang, S. Jin, and C. M. Lieber, "Nanolithography using hierarchically assembled nanowire masks," *Nanoletters*, vol. 3, no. 7, pp. 951–954, July 9 2003.

[20] D. Whang, S. Jin, Y. Wu, and C. M. Lieber, "Large-scale hierarchical organization of nanowire arrays for integrated nanosystems," *Nanoletters*, vol. 3, no. 9, pp. 1255–1259, September 2003.

[21] Y. Chen, D. A. A. Ohlberg, X. Li, D. R. Stewart, R. S. Williams, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, D. L. Olynick, and E. Anderson, "Nanoscale molecular-switch devices fabricated by imprint lithography," *Appl. Phys. Let.*, vol. 82, no. 10, pp. 1610–1612, 2003.

[22] Y. Chen, G.-Y. Jung, D. A. A. Ohlberg, X. Li, D. R. Stewart, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, and R. S. Williams, "Nanoscale molecular-switch crossbar circuits," *Nanotechnology*, vol. 14, pp. 462–468, 2003.

[23] Y. Dong, G. Yu, M. C. McAlpine, W. Lu, and C. M. Lieber, "Si/a-Si core/shell nanowires as nonvolatile crossbar switches," *Nanoletters*, vol. 8, no. 2, pp. 386–391, 2008.

[24] A. Asenov, "Intrinsic threshold voltage fluctuations in decanano MOSFETs due to local oxide thickness variation," *IEEE Trans. Electron Devices*, vol. 49, no. 1, pp. 112–119, January 2002.

[25] ——, "Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 μm MOSFET's: A 3-D "atomistic" simulation study," *IEEE Trans. Electron Devices*, vol. 45, no. 12, pp. 2505–2513, December 1998.

[26] A. Asenov, S. Kaya, and A. R. Brown, "Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness," *IEEE Trans. Electron Devices*, vol. 50, no. 5, pp. 1254–1260, May 2003.

[27] V. A. Sverdlov, T. J. Walls, and K. K. Likharev, "Nanoscale silicon MOSFETs: A theoretical study," *IEEE Trans. Electron Devices*, vol. 50, no. 9, pp. 1926–1933, September 2003.

[28] G. Yu, A. Cao, and C. M. Lieber, "Large-area blown bubble films of aligned nanowires and carbon nanotubes," *Nature Nanotechnology*, vol. 2, no. 6, pp. 372–377, Jun 2007.

[29] A. DeHon, P. Lincoln, and J. Savage, "Stochastic Assembly of Sublithographic Nanoscale Interfaces," *IEEE Trans. Nanotechnol.*, vol. 2, no. 3, pp. 165–174, 2003.

[30] "International technology roadmap for semiconductors," <http://www.itrs.net/Links/2008ITRS/Home2008.htm>, 2008.

[31] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. DeHon, "HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array," in *FPGA*, February 1999, pp. 125–134.

[32] D. Chen, J. Cong, M. Ercegovac, and Z. Huang, "Performance-driven mapping for CPLD architectures," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 10, pp. 1424–1431, October 2003.

[33] V. Betz, "VPR and T-VPack: Versatile Packing, Placement and Routing for FPGAs," <http://www.eecg.toronto.edu/~vaughn/vpr/vpr.html>, March 27 1999, version 4.30.

[34] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," in *FPGA*. ACM, February 1995, pp. 111–117.

[35] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, 2nd ed. Prentice Hall, 1999.

[36] S. Hanson, B. Zhai, K. Bernstein, D. Blaauw, A. Bryant, L. Chang, K. K. Das, W. Haensch, E. J. Nowak, and D. M. Sylvester, "Ultralow-voltage, minimum-energy CMOS," *IBM J. Res. and Dev.*, vol. 50, no. 4–5, pp. 469–490, July/September 2006.

[37] A. DeHon and H. Naeimi, "Seven Strategies for Tolerating Highly Defective Fabrication," *IEEE Des. Test. Comput.*, vol. 22, no. 4, pp. 306–315, July–August 2005.

[38] D. B. Strukov and K. K. Likharev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," *Nanotechnology*, vol. 16, no. 6, pp. 888–900, June 2005.

Appendix A
Device Characterization

VMATCH depends on the ability to characterize R_{offFET} of the transistors. In this section we present a testing technique that allows us to perform these measurements. Essentially, we take advantage of the NanoPLA architecture to configure voltage dividers between each input or *restore wire* and a reference resistance to estimate the restore wire's resistance and in turn the transistor's off resistance.

All restore wires are connected to source and ground voltage terminals. (*Source Voltage* and *Measurement Voltage* in Fig. 10). During operation, these columns are used as a restoration mechanism; however, by connecting a reference resistance to the bottom terminal we can create a voltage divider between the restore wire, including the transistor, and the reference resistance. Isolating each restore wire would allow us to measure the resistance of each wire and transistor, giving us all the information needed to run VMATCH. An examination of Fig. 10, however, shows that since all restore wires in a plane are connected to the same Source and Measurement terminals, it is not obvious how to fully isolate a restore wire in order to measure its resistance without measuring the resistance of other wires in the column as well.

Though direct isolation of the restore wires is not possible, it is possible to select one restore wire since programing diodes requires the ability to select the restore and output or *compute wires* connected to the diode being programmed. The mechanism that allows individual wire selection is the Address Decoder (Fig. 10). Compute wires are encoded with a k-hot code [29] that allows the decoder to isolate a particular compute wire. These, in turn, gate the transistor in the restore wires (one of *t1* through *t4*, Fig. 10b). We can turn all transistors off by selecting all compute wires and storing charge on them by isolating the charge between the decoder on one end and the precharge transistor on the opposite end (Fig. 10). Using the decoder we can then select one compute wire and charge it so that it turns the corresponding transistor on. If there was no variation, turning all transistors except one off would successfully isolate one restore wire. Unfortunately there are two problems with this idea. First as we have seen in Sec. III-B, under certain variation conditions, an off transistor can leak faster than an on transistor can switch. Second, the nanowires have high resistance. When the transistor is switched on, the wire resistance R_{inWire} will dominate the transistor resistance R_{onFET}. Thus, it is possible that either a highly leaky wire will charge the measurement terminal of the voltage divider before the restore wire we are interested in measuring does, or we will measure the restore wire's resistance and not the transistor's resistance. Both are results that don't provide the information we need for VMATCH.

To deal with the first problem, we strongly turn off all transistors. Eq. 6 shows that by using a voltage that is significantly greater than the operating voltage, we can counter the expected V_{th} variation and can guarantee that leakage is not a problem. Measuring the transistor resistance, R_{FET}, rather than R_{inWire} requires more work. Once all transistors are strongly off, we use the address decoder to select a wire to test. Sweeping the Test Voltage terminal (Fig. 10) from the on operating voltage to the off operating voltage and measuring at each point the resistance using the voltage divider, we can observe the resistance of the wire plus the resistance of the transistor. Moreover, this test voltage *vs.* resistance curve will clearly show when the wire resistance dominates and when the transistor resistance dominates. Finally, from this and Eqs. 5 and 6 we can characterize V_{th} for this transistor, producing the sufficient information for VMATCH. Appendix B accounts for the steps required for this measurement technique to work. Appendix C considers the time it takes to perform this testing.

Appendix B
Detailed Measurement Steps

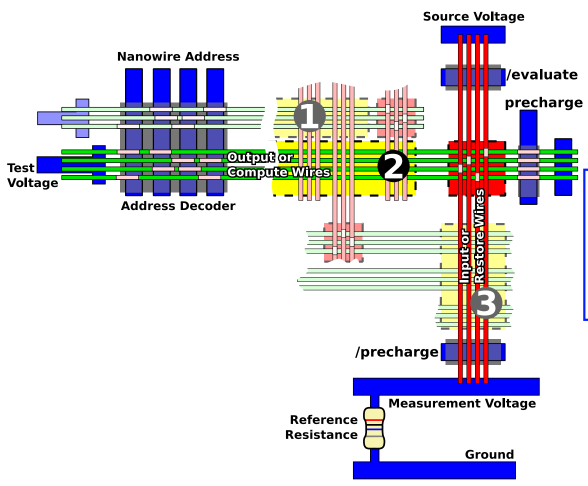
Measuring each restore wire requires a series of initialization steps done at the plane level followed by a set of measurements on the individual wire. These initialization and measurement steps are repeated for each wire that must be measured. What follows is a detailed description of these steps.

Using the decoder, we first select all compute wires at once by setting the reserved all zero nanowire address. This address enables conduction through all the nanowires. Then we isolate them using the precharge transistors and charge them through the test voltage terminal to a higher than normal voltage $V_{test} = strongOff$ so that the transistors on the restore wires are *strongly* turned off. $strongOff$ is chosen to be sufficiently high that it can disable even nanowires with the highest thresholds the variation allows across a chip.

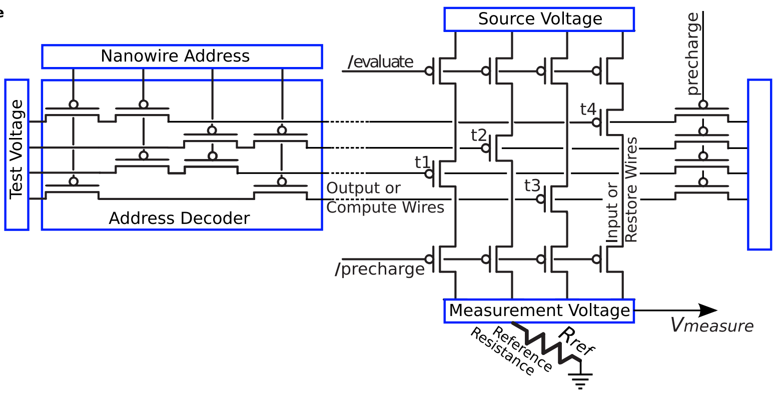
Once this initialization is done, we select one compute wire and perform a set of measurements on the corresponding restore wire. We address the compute wire with the decoder, set the test voltage terminal to the on operating voltage and measure $V_{measure}(V_{test} = on)$. In order to observe the effects of R_{FET} we take several more measurements as we sweep the voltage V_{test} between the on operating voltage and the off operating voltage. This generates a set of measurements $\{V_{measure}(V_{test} = on), V_{measure}(V_{test} = v_1), \ldots, V_{measure}(V_{test} = off)\}$ which we can use to solve for the actual resistances R_{wire} using Eq. 7.

$$R_{wire}(V_{test}) = \frac{R_{ref} \cdot V_{source}}{V_{measure}(V_{test})} - R_{ref} \tag{7}$$

Each $R_{wire}(V_{test})$ will be composed of three resistances. The series combination of the actual wire resistance, R_{inWire} and the transistor resistance, R_{FET} (either on or off, depending



(a) NanoPLA Block with Address Decoder



(b) NanoPLA Inversion Array Circuit with Address Decoder

Fig. 10: NanoPLA Organization with Address Decoder

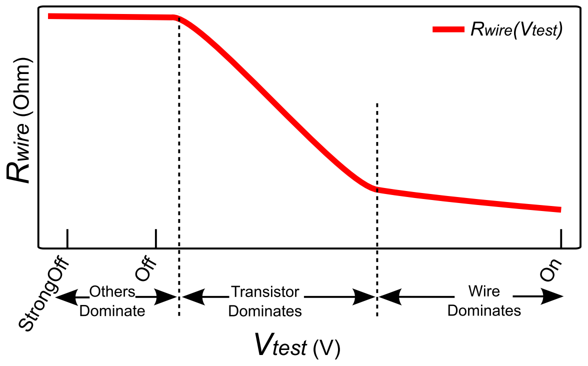


Fig. 11: Expected R_{wire} as a function of V_{test} showing where R_{FET} dominates and where R_{inWire} dominates

on the value of V_{test}) in parallel with the parallel resistance of all the other wires not being tested, R_{rest}.

$$\frac{1}{R_{rest}} = \sum_{i \neq \text{test wire}} \left(\frac{1}{R_{wire_i}\left(strongOff\right)} \right) \quad (8)$$

Eq. 9 shows the composition of R_{wire}, the series resistances of the wire followed by the transistor all together in parallel with the resistance of the other wires.

$$\frac{1}{R_{wire}(V_{test})} = \frac{1}{R_{FET}(V_{test}) + R_{inWire}} + \frac{1}{R_{rest}} \quad (9)$$

Plotting the measured R_{wire} as a function of V_{test} would look similar to Fig. 11. At $V_{test} = strongOff$ the leakage from the other wires may dominate. As V_{test} decreases, the transistor resistance dominates R_{wire} by becoming small compared to R_{rest} while still dominating R_{inWire}. Eventually, however, V_{test} decreases enough that the wire resistance, R_{inWire}, dominates R_{wire}. By taking enough measurements in the section of Fig. 11 where the transistor dominates, we can use Eq. 9 along with Eqs. 5 and 6 to characterize the transistor and estimate V_{th}.

This measurement technique requires that we store the V_{th} for all of the transistors in our NanoPLA which is not ideal. It is explained here as a proof of concept rather than an optimal measurement algorithm. We believe that more efficient measurement techniques are possible and expect to develop them in our future work.

APPENDIX C
TIME TO MEASURE

To estimate the V_{th} of each wire, we need to make a series of measurements as described above. Addressing control operations to precharge and select nanowires takes less than 10ns. When the wire resistance dominates, the RC time constant of wires is around 10ns for the Toronto 20 benchmarks and our technology assumptions. As we increase V_{test} and begin to turn off the transistor, the effective resistance increases (Fig. 11), increasing the RC time constant. Even if we make 10–20 measurements and allow each measurement a settling time around 50μs, the total time to make the full set of measurements will be under 1ms. Limiting measurement time to 50μs allows us to accurately measure resistances up to 1GΩ. This provides an adequate range for measurements since 1GΩ is 1000 times the 1MΩ nominal on resistance of the most resistive nanowires and is comparable to the nominal R_{rest}.

On average, the Toronto 20 benchmarks, use a grid of 20$\times$20 blocks and, accounting for all three planes, each NanoPLA block has 250 restore wires. Using our conservative assumption of 1ms per device, measuring every device in the NanoPLA takes on average 100 seconds. As we explore larger designs than those in the Toronto 20 Benchmarks, the time to measure all devices may grow to hours or days. However, the measurement technique above can be parallelized when the wires are in different blocks brining the time back to seconds.