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
1bgojman@seas.upenn.edu

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

© 2009 IEEE
A. Technology: Nanowires

Nanowires are the main building block of the NanoPLA. These can be made 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, chemists 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. Stage two and three behave like the AND and OR 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]).

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

Throughout this paper we express the amount of variation as a percentage equal to $\sigma/\mu$; we will refer to this simply as $\sigma$. Though other works also report variation as a percent it is worth noting that many, including the ITRS [30], tend to report $3\sigma$ variation while we label our variation points by $\sigma$. 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
To calculate the variation

\[ \tau_{\text{cycle}} = \tau_{\text{phase1}} + \tau_{\text{phase2}} + \tau_{\text{phase3}} \]

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, \( \tau_{\text{switch}} \), is defined as the time it takes the slowest used NAND-term to switch. Similarly the precharge leak time, \( \tau_{\text{leak}} \), is the time it takes the leakiest used NAND-term to lose its percharged 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:

\[ \max_{\text{planes}} \left( \frac{\tau_{\text{switch}}}{\text{plane}} \right) \leq \tau_{\text{phase}} \leq \min_{\text{planes}} \left( \frac{\tau_{\text{leak}}}{\text{plane}} \right) \]

To provide adequate noise margins we demand at least two orders of magnitude separation between the worst case \( \tau_{\text{switch}} \) and \( \tau_{\text{leak}} \). 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:

\[ 100 \cdot \max_{\text{planes}} \left( \frac{\tau_{\text{switch}}}{\text{plane}} \right) \leq \min_{\text{planes}} \left( \frac{\tau_{\text{leak}}}{\text{plane}} \right) \]

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:

\[ \tau_{\text{switch}} = (R_{\text{contact}} + R_{\text{onFET}} + 0.5R_{\text{inWire}}) \times (C_{\text{inWire}} + \sum_{\text{fanout}} C_{\text{outWire}}) \]

\[ \tau_{\text{leak}} = (R_{\text{contact}} + R_{\text{offFET}} + 0.5R_{\text{inWire}}) \times (C_{\text{inWire}} + \sum_{\text{fanout}} C_{\text{outWire}}) + (R_{\text{diode}} + 0.5R_{\text{outWire}}) \cdot C_{\text{outWire}} \]

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, \( \sum_{\text{fanout}} C_{\text{outWire}} \).

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, \( R_{\text{onFET}} \) and \( R_{\text{offFET}} \). 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
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 \( \tau_{\text{switch}} \) and \( \tau_{\text{leak}} \) (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 \( \tau_{\text{switch}} \) and \( \tau_{\text{leak}} \) (Sec. IV-E).

A. Variation-Oblivious Mapper

At high levels of variation, the distribution of \( \tau_{\text{switch}} \) and \( \tau_{\text{leak}} \) 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 \( 100 \times \tau_{\text{switch}} \) and of \( \tau_{\text{leak}} \) 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 \( \tau_{\text{switch}} \) and \( \tau_{\text{leak}} \) 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 \( 100 \cdot \tau_{\text{switch}} \lesssim \tau_{\text{leak}} \). Since the only difference between \( \tau_{\text{switch}} \) and \( \tau_{\text{leak}} \) is the state of the transistor being on and off respectively (see Eq. 3 and 4), for correct operation \( R_{\text{offFET}} \) must be the dominant term in \( \tau_{\text{leak}} \). If this were not the case and one of the other terms in Eq. 4 dominated, there would be nearly no difference between \( \tau_{\text{switch}} \) and \( \tau_{\text{leak}} \) and, as such, correct operation would be impossible regardless of how the design is mapped.

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

\[
I_{\text{sat}} = Wv_{\text{sat}}C_{\text{ox}}(V_{\text{gs}} - V_{\text{th}} - 0.5 \cdot V_{\text{d, sat}}) \quad (5)
\]

IV-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 \( \tau_{\text{leak}} \), 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 \( \tau_{\text{switch}} \) and \( \tau_{\text{leak}} \) 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 \( \tau_{\text{leak}} \) distribution (Fig. 2) that is below \( 6 \times 10^{-4s} \). 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 \( \tau_{\text{leak}} \) 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 no 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 \( AB + AC'D + BE + AF \) to a block in the NanoPLA, three terms in the AND-plane will use input signal \( A \) (\( AB, AC'D, \) 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 \( C_{\text{out,wire}} \) 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 \( \tau_{\text{leak}} \) 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 \( R_{\text{off,FET}} \) 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 (\( \tau_{\text{cycle}} \)), 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 \( R_{\text{off,FET}} \) by carefully matching a high-fanout term with a low \( R_{\text{off,FET}} \) 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 why this works by examining how the \( \tau_{\text{leak}} \) distribution changes based on how each of the three algorithms uses the \( R_{\text{off,FET}} \) 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 \( R_{\text{off,FET}} \) variation leading to the twelve orders of magnitude range of \( \tau_{\text{leak}} \). The defect avoiding algorithm limits \( \tau_{\text{leak}} \)'s range by directly limiting the range of \( R_{\text{off,FET}} \) values used, but this must discard almost half of the resources.

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. Nonetheless, 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 \( \max(\tau_{\text{switch}}) \) and maintain performance, we map the lowest (highest fanout) logical NAND-term to the fastest (lowest \( R_{\text{off,FET}} \)) physical NAND-term and use the resulting \( \tau_{\text{leak}} \) as a minimum target, \( \tau_{\text{target}} \), 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 \( \tau_{\text{target}} \) is calculated as described above. Each logical function in the queue is mapped to the physical NAND-term with the lowest \( R_{\text{off,FET}} \) that does not cause it to have a \( \tau_{\text{leak}} \) lower than \( \tau_{\text{target}} \). 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 \( \tau_{\text{target}} \) to be that NAND-term's \( \tau_{\text{leak}} \).

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 \( \max(\tau_{\text{switch}}) \) and only requires 3% extra resources over MinC.

To build intuition for this success, Fig. 6 presents the distribution of the used \( R_{\text{off,FETs}} \) 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 $L \in \{ \text{logic} \}$ 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)
                Target := NT.Tleak()
        end
    end
end

Algorithm 1: VMATCH

---

Fig. 5: Distribution of $\tau_{\text{leak}}$ and $100 \times \tau_{\text{switch}}$ of VMATCH mapping. Benchmark sla 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

---

![Graph showing the distribution of $\tau_{\text{leak}}$ and $100 \times \tau_{\text{switch}}$ of VMATCH mapping.](image)

---

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%.
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 \( R_{\text{off}} \) 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.

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.

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 “nanscale 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

<table>
<thead>
<tr>
<th>Net</th>
<th>% Extra Channels Mean</th>
<th>St.Dev.</th>
<th>Delay St.Dev. Mean</th>
<th>Energy St.Dev. Mean</th>
<th>Area St.Dev. Mean</th>
<th>Delay St.Dev. Mean</th>
<th>Energy St.Dev. Mean</th>
<th>Area St.Dev. Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>misex</td>
<td>12%  2%</td>
<td>170 34</td>
<td>2.1 0.32</td>
<td>0.05 0.08</td>
<td>1.3 0.23</td>
<td>0.03 0.02</td>
<td>1.2 0.13</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>apex</td>
<td>13%  3%</td>
<td>192 34</td>
<td>3.5 0.43</td>
<td>0.09 0.08</td>
<td>1.7 0.31</td>
<td>0.05 0.02</td>
<td>1.6 0.13</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>sigkey</td>
<td>12%  3%</td>
<td>164 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>dime</td>
<td>8%  3%</td>
<td>174 48</td>
<td>1.8 0.24</td>
<td>0.14 0.08</td>
<td>1.5 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>top</td>
<td>12%  2%</td>
<td>186 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>towel</td>
<td>12%  2%</td>
<td>186 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>misex2</td>
<td>10%  3%</td>
<td>164 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>pack</td>
<td>9%  2%</td>
<td>192 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>tk78</td>
<td>9%  2%</td>
<td>186 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>smokel</td>
<td>8%  2%</td>
<td>164 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>toge</td>
<td>8%  2%</td>
<td>186 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
<tr>
<td>mean</td>
<td>10%  3%</td>
<td>192 34</td>
<td>3.0 0.32</td>
<td>0.14 0.08</td>
<td>1.8 0.34</td>
<td>0.10 0.02</td>
<td>1.8 0.14</td>
<td>0.02 0.01</td>
</tr>
</tbody>
</table>

TABLE I: Tolerable Variation and Overheads Required for Toronto 20 Benchmark Set
VMATCH depends on the ability to characterize $R_{off,FET}$ 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 $R_{in,Wire}$ will dominate the transistor resistance $R_{on,FET}$. 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

\[ R_{wire}(V_{test}) = \frac{R_{ref} \cdot V_{source}}{V_{measure}(V_{test})} - R_{ref} \]  

Each $R_{wire}(V_{test})$ will be composed of three resistances. The series combination of the actual wire resistance, $R_{in,Wire}$ and the transistor resistance, $R_{FET}$ (either on or off, depending
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 50 $\mu$s, the total time to make the full set of measurements will be under 1ms. Limiting measurement time to 50 $\mu$s allows us to accurately measure resistances up to 1G$\Omega$. This provides an adequate range for measurements since 1G$\Omega$ is 1000 times the 1M$\Omega$ 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 20×20 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 bringing the time back to seconds.