Algorithms and Techniques for Conquering Extreme Physical Variation in Bottom-Up Nanoscale Systems

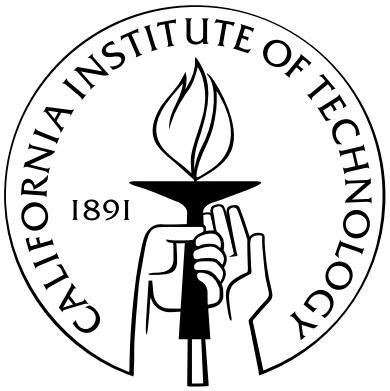
Thesis by

Benjamin Gojman

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science



California Institute of Technology

Pasadena, California

2010

(Submitted April 5, 2010)

Acknowledgements

This work would not have been possible without the constant support and motivation from my advisor, André DeHon. His patience and guidance are invaluable to me. André, it is because of your dedication that I successfully completed this work. Thank you for all your help.

Raphael Rubin and Nikil Mehta were instrumental in the development of this thesis. Rafi, I am grateful for both the insightful discussions we had about the technical aspects of this work as well as the uncountable amount of infrastructure support you provided. Nick, without your in-depth knowledge, I would have been lost trying to understand all the low level details of the NanoPLA. I also want to thank the other members of the IC Group, Nachiket Kapre, Michael deLorimier and Corey Waxman, for their advice and encouragement.

Emily Traver has been with me through this whole process, delighting in the ups and never failing to be there when things got rough. I am grateful that you were with me every step of the way.

Finally, I want to thank my family Marcos and Karen, Mauricio and Monica Gojman for their love and constant support and Dita for always believing that this project would come to a successful end.

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 author and do not necessarily reflect the views of the National Science Foundation.

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. Characterization of the complete threshold voltage distribution present in the system can be measured at a rate of 10^8 resources per second by augmenting the system with voltage comparison mechanisms. 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. What's more, there is only a 10% overhead when the measurement precision is limited to ten discrete threshold voltage values.

Contents

Chapter 1

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.

1.1 Overview

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. We present an efficient technique to measure the physical variation and show that although it can provide high precision results, VMATCH only requires moderate precision measurements. 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].

The next chapter reviews the bottom-up technology that enables the manufacturing of the NanoPLA (Section 2.1) along with its architecture (Section 2.2) as introduced in [10]. The chapter concludes by considering the sources of the variation present in the NanoPLA (Section 2.3).

Chapter 3 explains how the NanoPLA functions as well as how it is used. In particular, it introduces the defect model (Section 3.2) which enables, in later chapters, the discussion of why and how the NanoPLA fails due to variation.

In Chapter 4, we motivate and introduce VMATCH, our algorithm to mitigate the negative effects of variation. Specifically, we first recognize that ignoring variation invariably leads to failure (Section 4.1). A partial understanding of the variation in the NanoPLA (Section 4.2), leads to an expensive solution (Section 4.3). Finally, full insight on the variation in the system (Section 4.4), naturally leads to VMATCH (Section 4.5).

VMATCH requires knowledge of the electrical characteristics of the underlying devices in the NanoPLA. Chapter 5 explores how these measurements can be obtained and how the NanoPLA is suited for measuring specifically the characteristics required by VMATCH. After analyzing the circuit model (Section 5.2) we explain how to configure the NanoPLA to make these measurements (Sections 5.3) and how long it takes to characterize a full NanoPLA (Section 5.4). The details of the measurement algorithm are then presented (Section 5.5). We finish with an analysis of the effect of limited measurement precision (Section 5.6).

Chapter 6 provides experimental results that demonstrate the effectiveness of VMATCH by comparing it to other algorithms and to a hypothetical variation free case. This chapter also considers the amount of measurement precision need to provide enough information for VMATCH to produce a successful mapping. Finally, conclusions are drawn in Chapter 7.

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 [11] of the impact of: (a) ignoring variation, (b) treating variation as defects, and (c) using VMATCH to mitigate variation.
- Measurement technique to efficiently characterize the resources in the NanoPLA.
- Quantification of the measurement precision required to extract enough information for VMATCH to successfully map a design.

Chapter 2

Background

The NanoPLA is fabricated through a novel bottom-up process where nanowires are first grown or otherwise manufactured and assembled into regular crossbar arrays. In this chapter we review this bottom-up technology along with the architecture of the NanoPLA itself. Understanding this construction, we examine why it leads to structures with high variation, and how it manifests in the electrical properties of the NanoPLA.

2.1 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 [12], and Au [13]. These wires can be microns long [14] 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 [15, 16] 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 [17, 18].

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 [19, 20]. By using nanowires with insulating shells, the LB technique can tightly pack nanowires without shorting them. These shells can later be selectively etched away [20]. What's more, to reduce the resistivity of the wires, they can be nickel silicide in the region where they do not interact with other wires [21]. When repeated, this process allows for two orthogonal layers to form a densely packed nanowire crossbar [19, 22].

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 [23, 24], amorphous silicon nanowire coatings [25], and nanowires made of switchable species [1]. Some of these programmable switches have diode-like rectification allowing the crossbars to be directional, only letting

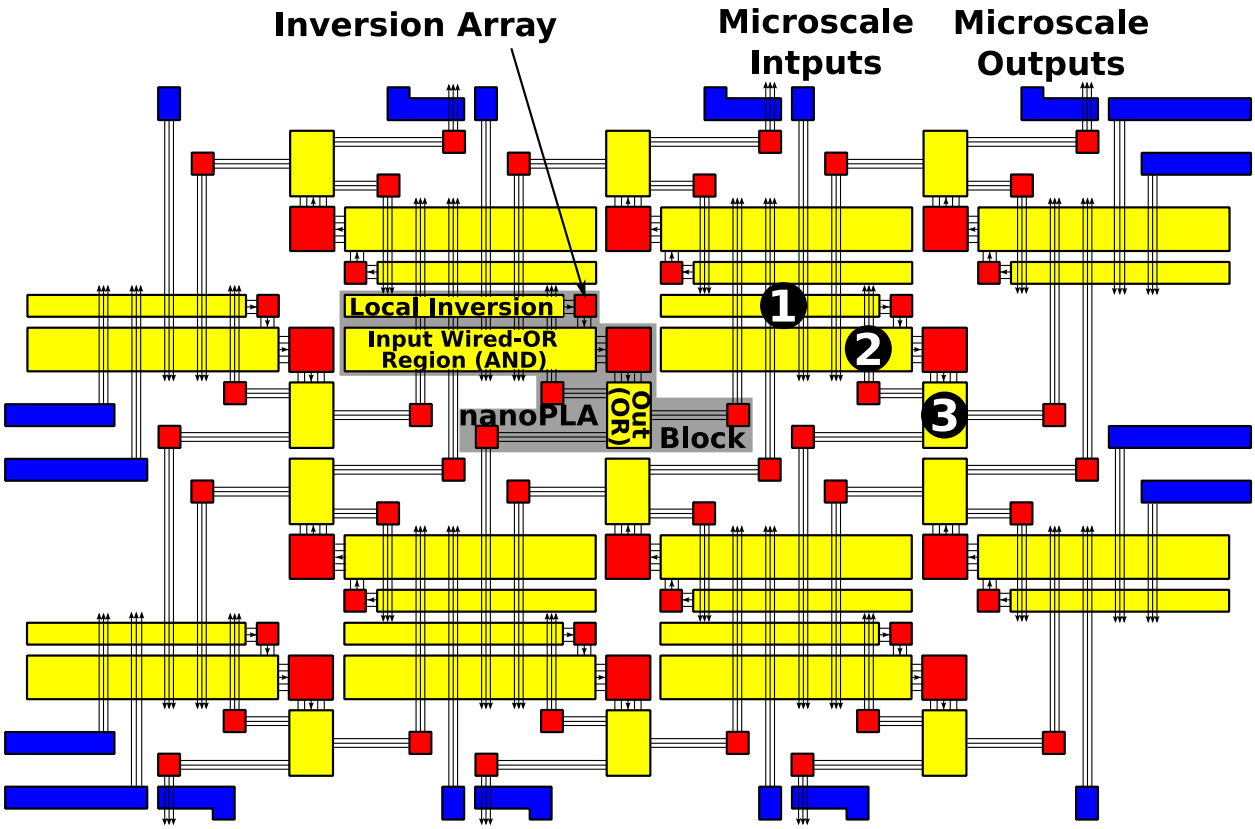


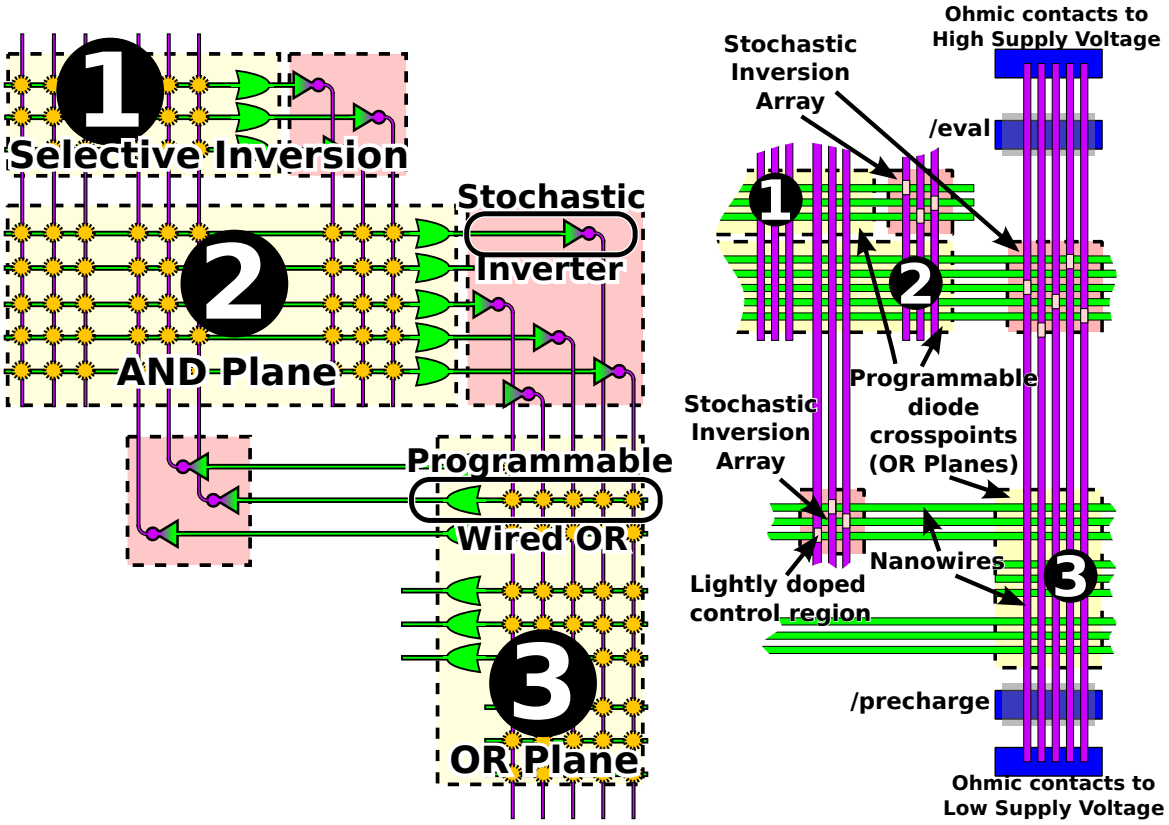
Figure 2.1: NanoPLA Block Tiling

charge flow from the vertical wires to the horizontal wires. This property is essential for correct operation because it allows the crosspoints to implement wired-OR gates, part of the basic unit of computation in the NanoPLA.

2.2 Architecture: NanoPLA

The NanoPLA is organized as shown in Figure 2.1. It consists of tiled logic blocks with overlapping nanowires that enable Manhattan routing while maintaining direct nanoscale-density interconnect among blocks. It is based on the local inversion design presented in [10] and uses amorphous Si switches [25] to improve performance and energy over the design in [4]

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 inputs❶. Stage two and three behave like the AND❷ and OR❸ planes respectively (Figure 2.2). The benefit of having an initial inverting phase is that it avoids the need for non-inverting restoration which [10] shows is a costly design choice, reducing performance and increasing total energy.



(a) Logical representation

(b) Physical nanowire implementation

Figure 2.2: NanoPLA Block

Figure 2.2b shows a physical 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 Figure 5.1) are used to program the diode-like switches. They are built as described in [4] and demonstrated in [16].

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. These connections can be seen in Figure 2.1 as multiple wires passing over a few blocks.

2.3 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.* Local oxide thickness variation [26], statistical dopant variation [27] and dopant placement, line-edge roughness [28], channel length variation [29]) 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 [30] during assembly cause the geometry of the field-effect regions to vary from device to device [31].
- 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 [29] calculates that the 5 nm long transistors we are considering are nearly impossible to manufacture reproducibly. We assume independent Gaussian distributions for these values consistent with the models and experimental results from the literature (*e.g.* [29, 28, 26, 27]).

$$P(x) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right) e^{\left(-\frac{(x-\overline{x})^2}{2\sigma^2}\right)} \tag{2.1}$$

Throughout this work 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 [32], 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 [32].

Chapter 3

System Model

3.1 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. Figure 3.1 shows the physical nanowire implementation and 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.

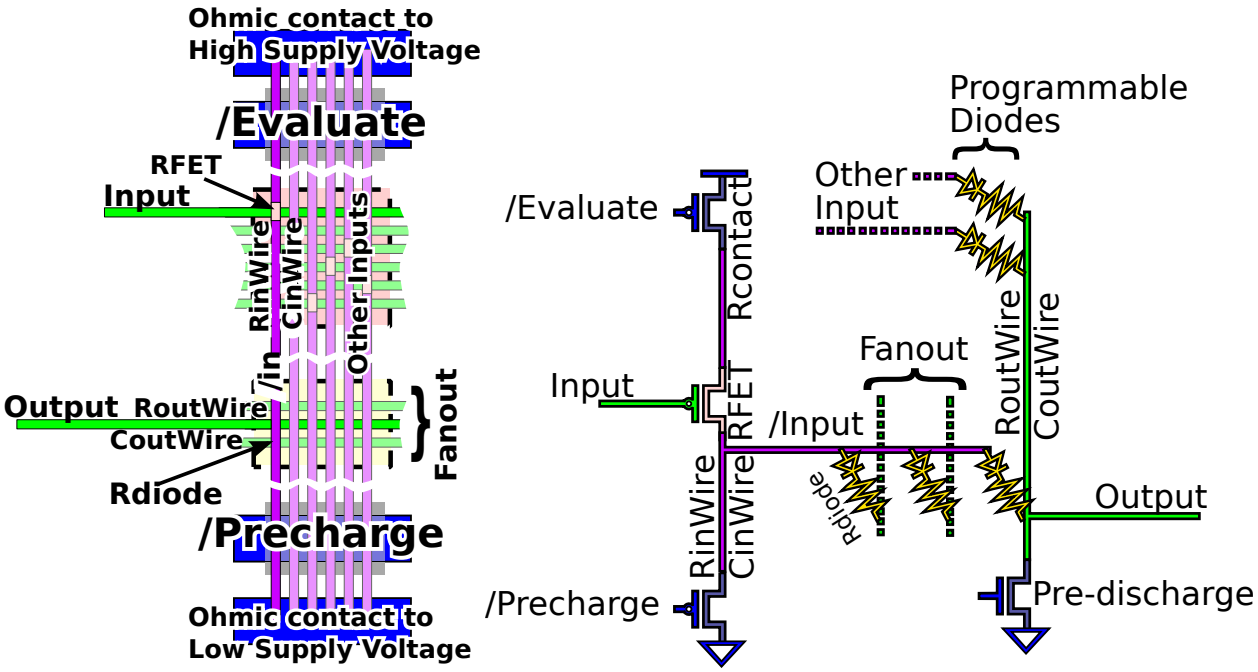
Within each plane, computation is done in a precharge fashion by first pre-discharging the nanowires and then evaluating the inputs. Since each block is composed of three planes, as shown in Figure 2.2, the evaluation scheme demands that we use a three-phased clock to sequence logic in the NanoPLA. At 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.* [33]), allowing for high throughput.

3.2 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 precharge 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:

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

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



(a) Physical Implementations (b) Circuit Diagram

Figure 3.1: NanoPLA NAND-term

therefore leakage current will be less than 1% of drive current across all blocks for a functional NanoPLA. We can state this constraint as:

$$100 \cdot \max_{planes}(\tau_{switch}) \leq \min_{planes}(\tau_{leak}) \tag{3.1}$$

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.*

3.3 Timing Model

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

$$\begin{aligned}\tau_{switch} &= (R_{contact} + R_{onFET} + \frac{1}{2}R_{inWire}) \times (C_{inWire} + \sum_{fanout} C_{outWire}) \\ &\quad + (R_{diode} + \frac{1}{2}R_{outWire}) \cdot C_{outWire}\end{aligned} \tag{3.2}$$

$$\tau_{leak} = (R_{contact} + R_{offFET} + \frac{1}{2}R_{inWire}) \times (C_{inWire} + \sum_{fanout} C_{outWire}) \quad (3.3)$$
$$+(R_{diode} + \frac{1}{2}R_{outWire}) \cdot C_{outWire}$$

Each term in the equation maps to a physical section of the NAND-term as shown in Figure 3.1. 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 (Equation 2.1). Also modeled as a Gaussian distribution is V_{th} variation which is used in Equations 4.1 and 4.2 to calculate the variation of the on and off resistance of the transistor, R_{onFET} and R_{offFET}. Since the dominate variation is random (Section 2.3), we assume independent distributions in this paper.

3.4 NanoPLA CAD Flow

Here we review how logic is mapped on the NanoPLA. Covering and clustering [34] is followed by a block-level placement computed using VPR 4.3 [35]. 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 [36] perform this block-level routing or *global route*. As shown in Figure 3.2, at this point 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. The global route stage also determines the minimum number of wires needed for the design to route. *MinC*, the minimum channel width, is marked in Figure 3.2.

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 chapter explains why this variation-oblivious mapping produces defective chips and introduces a solution.

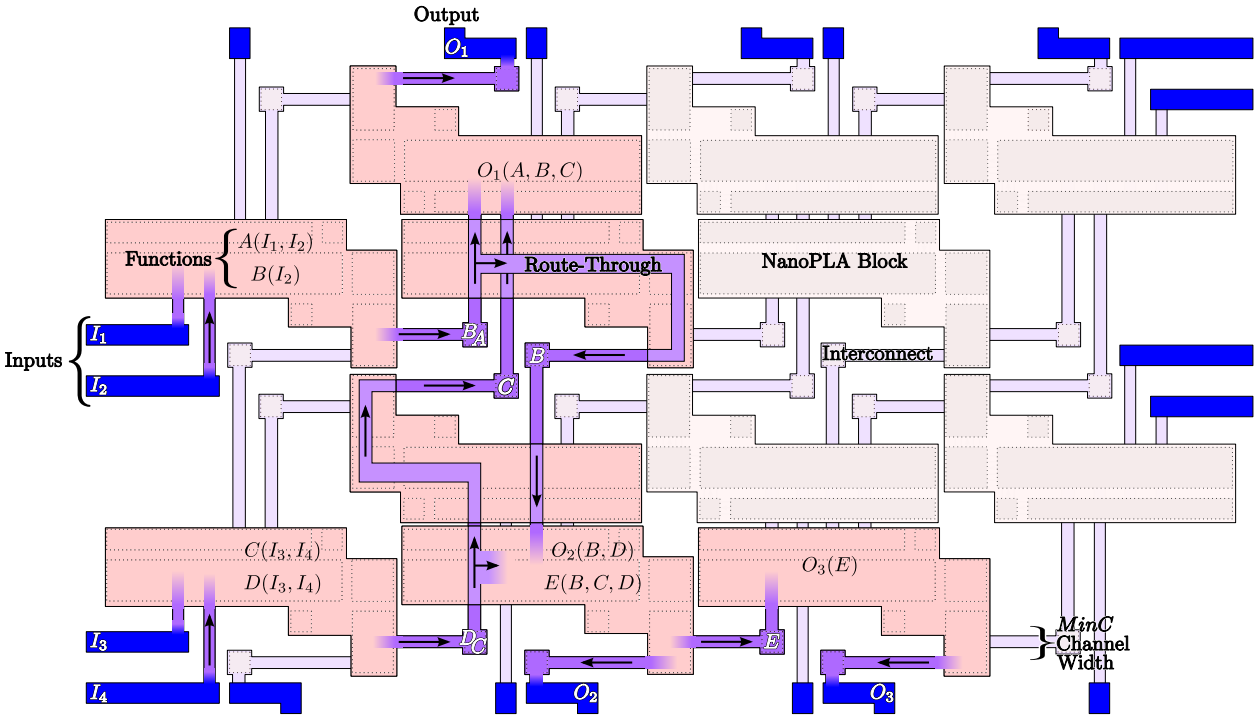


Figure 3.2: NanoPLA After Placement and Global Route: Function and Route-Troughs assigned to Blocks. Minimum Channel Width, $MinC$, Calculated.

Chapter 4

Device Specific Mapping

In this chapter, we illustrate why the mapper must consider the physical variation (Section 4.1). We examine how variation affects τ_{switch} and τ_{leak} (Section 4.2) and introduce a naive solution that satisfied Equation 3.1 but at a high cost (Section 4.3). In Section 4.4 we investigate how to improve on the naive solution. Finally we introduce VMATCH, our algorithm that considers the effects the mapping has on τ_{switch} and τ_{leak} (Section 4.5).

4.1 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 Equation 3.1 is almost zero. Figure 4.1 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 Equation 3.1 does not hold.

4.2 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 Equation 3.1 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 Equation 3.2 and 3.3), 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 Equation 3.3 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 $R_{offFET} = V_{dd}/I_{sub}$ (Equation 4.2). In the on state, the transistor operates in saturation, and we define the value of R_{onFET} as V_{dd}/I_{sat} (Equation 4.1). Since the

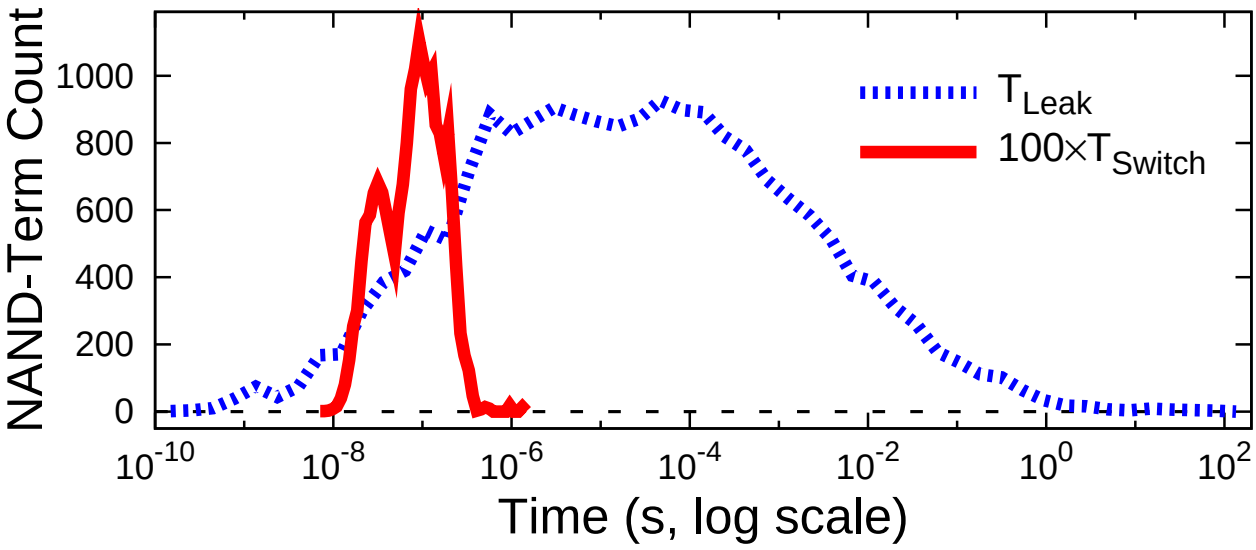


Figure 4.1: Distribution of τ_{leak} and $100 \times \tau_{switch}$ of a delay oblivious-mapping. Benchmark spla at $\sigma = 38\%$

nanowires are still Silicon, we use short-channel P-type MOSFET current equations [37, 38]:

$$I_{sat} = W v_{sat} C_{ox} \left(V_{th} - V_{gs} - 0.5 \cdot V_{d,sat}\right) \tag{4.1}$$

$$I_{sub} = \frac{W}{L} \mu C_{ox} (n-1) \cdot {v_T}^2 e^{\frac{V_{th} - V_{gs}}{n v_T}} \left(1 - e^{-V_{ds} \cdot {v_T}^{-1}}\right) \tag{4.2}$$

We see that saturation current is *linear* in V_{th} and V_{gs} and that sub-threshold current is *exponential* in V_{th} and V_{gs}. 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 $V_{th} = 295\text{mV}$ and $V_{dd} = 0.7\text{V}$, the mean value for R_{onFET} is $7.0 \times 10^4\Omega$ and for R_{offFET} is $1.1 \times 10^{12}\Omega$. At $\sigma = 38\%$, the 3σ V_{th} variation point gives a range for R_{onFET} from $3.2 \times 10^4\Omega$ to $7.1 \times 10^6\Omega$. For R_{offFET} the range is from $1.8 \times 10^7\Omega$ to $7.0 \times 10^{16}\Omega$. While the -3σ R_{offFET} value is larger than the $+3\sigma$ R_{onFET}, they are less than a factor of two apart and hence do not satisfy Equation 3.1. Figure 4.2 shows the full $\pm 3\sigma$ range. Given that all other parameters in Equations 3.2 and 3.3 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 Equation 3.1.

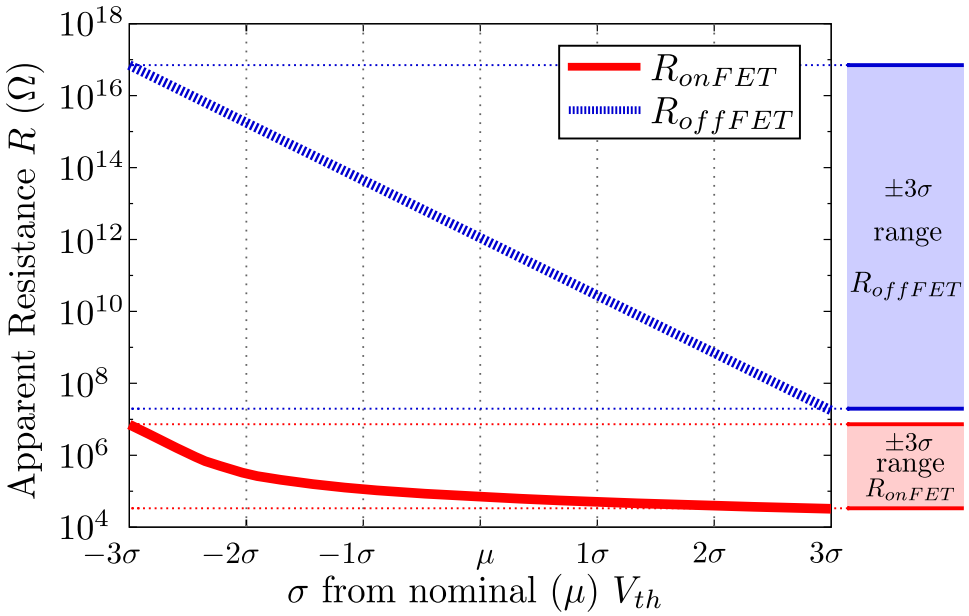


Figure 4.2: R_{onFET} and R_{offFET} ranges over $\pm 3\sigma$ of nominal V_{th}.

4.3 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, 39, 40], 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 Equation 3.3 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.

By avoiding resources in the fast tail of the distribution, the τ_{switch} and τ_{leak} distribution essentially shift towards higher delays. This helps creates the required two orders of magnitude separation because the τ_{switch} distribution shifts by a linear amount while τ_{leak}'s distribution shifts exponentially towards a higher delay.

Figure 4.3 shows the result of mapping the same chip shown in Figure 4.1. 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 (Figure 4.1) that is below 6×10^{-5}s. In Chapter 6 we show that this defect-avoidance algorithm, on average, needs 193% more resources than the variation-free case.

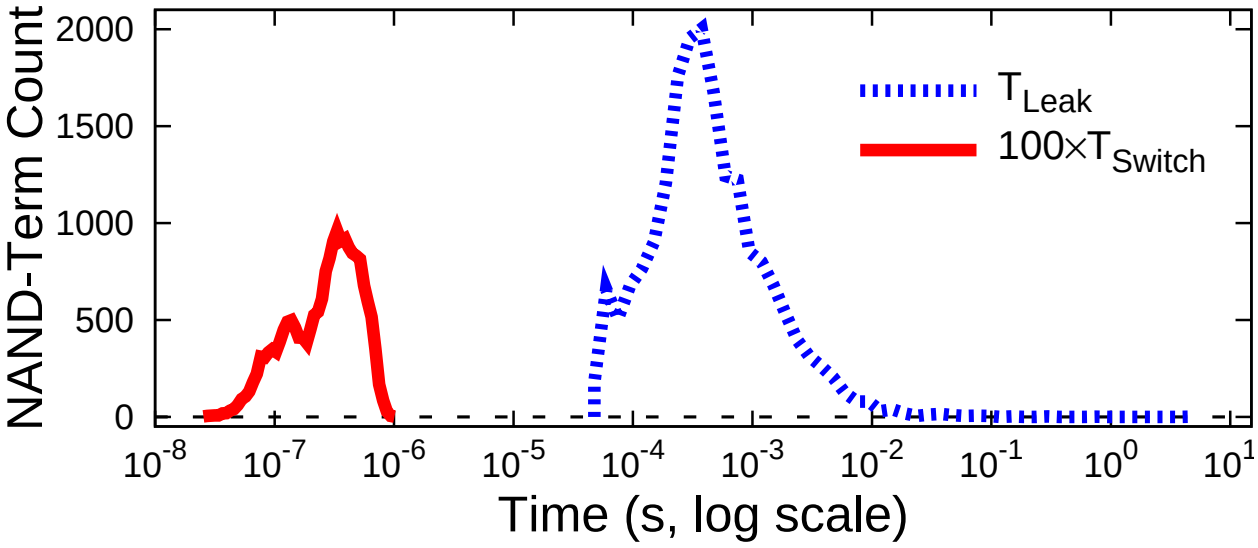


Figure 4.3: Distribution of τ_{leak} and $100 \times \tau_{switch}$ of a Defect-Avoidance mapping. Benchmark spla at $\sigma = 38\%$

4.4 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 Equation 3.3, 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 (Figure 3.1). 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 Figure 2.1 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. Figure 4.4 shows a typical distribution with a

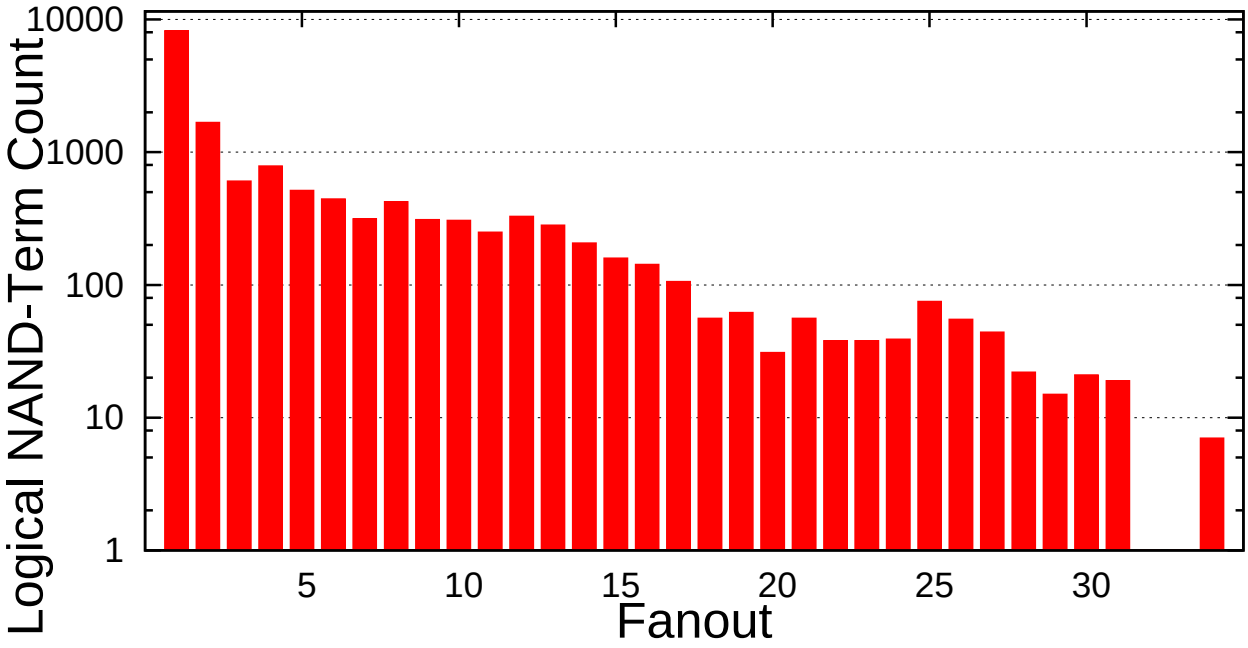


Figure 4.4: Fanout distribution. Benchmark spla

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 Equations 3.2 and 3.3. 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 inputs and AND-terms to reduce the fanout but only at the expense of increasing the total energy, area, and evaluation latency. Figure 4.5 shows the trade-offs between delay and area. Multiple points in the space were explored. The figure highlights the number of inputs but each point represents a unique combination of inputs, AND-terms and outputs. It shows that the best trade-off occurs when the inputs are 16. [10] fully explores this space for the variation free mapping.

While smaller arrays can reduce the clock cycle (τ_{cycle}), they increase the number of blocks in a logical evaluation path. For each benchmark, our design point was chosen from the results presented in [10] so that the overall evaluation time and area are both close to minimum across the array shape parameter space.

4.5 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

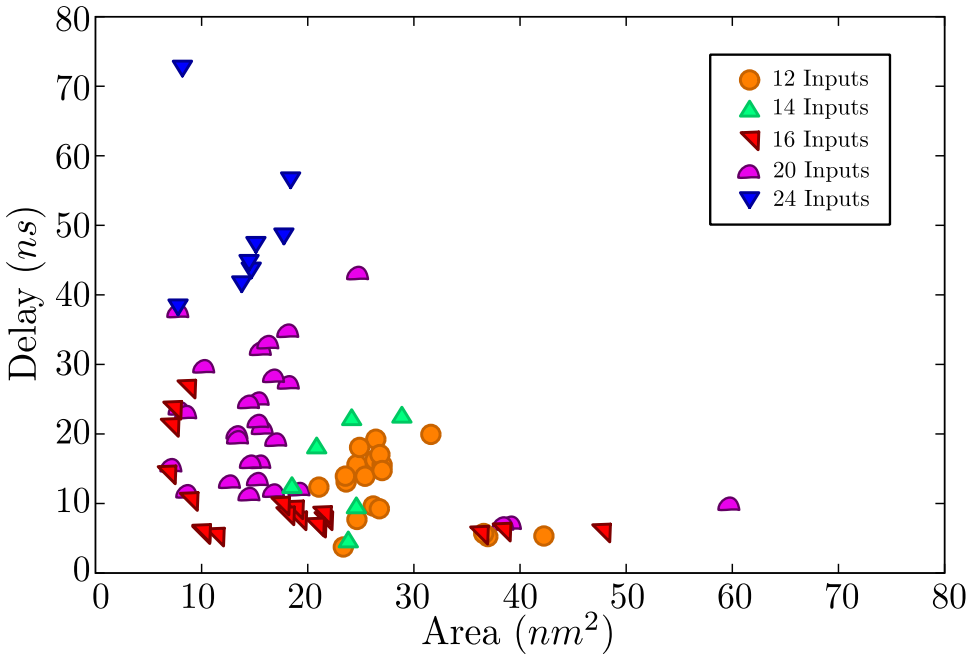
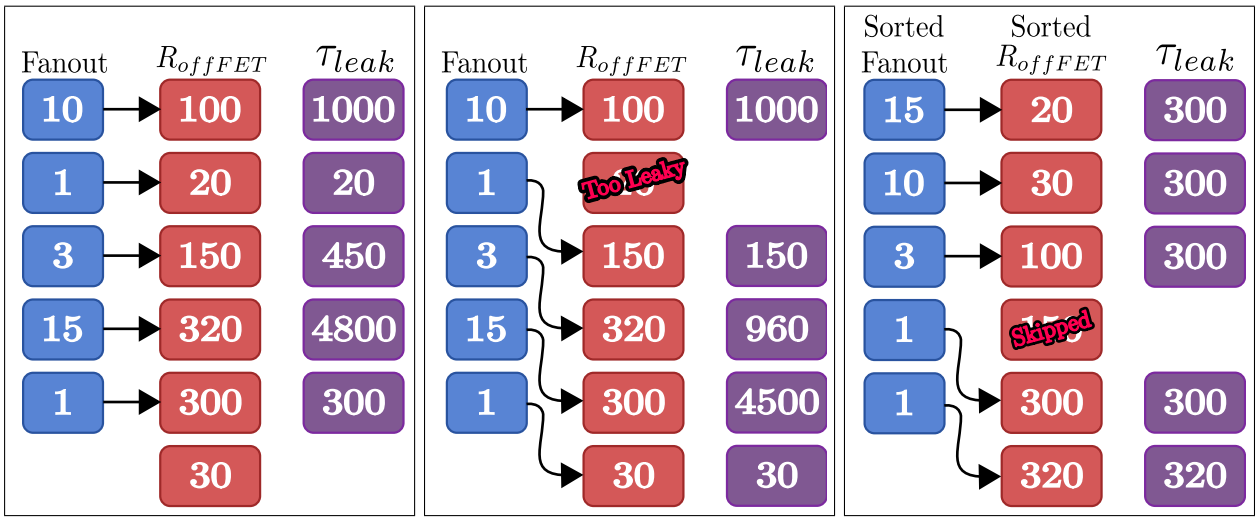


Figure 4.5: Delay-Area trade-off highlighting inputs parameter. Each point represents a unique (inputs, AND-terms, outputs) tuple. Benchmark spla variation free

variation-free case. A limited version of VMATCH was first introduced in [41]. Here we present a more robust version of the algorithm.

We can understand 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 (Figure 4.4) 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 Figure 4.1. 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. Figure 4.6 shows a simplified version of the problem where we clearly see the result of applying the three algorithms, variation oblivious, defect avoiding and VMATCH, to the same problem. As explained, the oblivious algorithm worsens the variation. Avoiding leaky resources helps reduce the spread of τ_{leak} but only slightly. To nearly eliminate all variation in τ_{leak}, we need to use VMATCH.

To perform this variation-aware post-fabrication mapping it is necessary to measure the nanowire transistor threshold voltages. What follows assumes knowledge of these measurements, and explains how VMATCH takes advantage these measurements. Chapter 5 details one way in which these measurements can be made.



(a) Oblivious Algorithm (b) Defect-Avoiding Algorithm (c) VMATCH Algorithm

Figure 4.6: Simple example showing predicted T_{leak} for the three algorithms applied to the same problem.

4.5.1 Algorithm Details

In order to reduce $\max(\tau_{switch})$ and maintain performance, we attempt to map every function (logical NAND-term) to the fastest (lowest R_{offFET}) resource (physical NAND-term) that will not violate Equation 3.1. Before mapping, the slowest functions will be those with high fanout as Equation 3.2 implies. Thus, we make sure to map functions in order of highest to lowest fanout so that the high fanout functions can take advantage of the fastest resources and counteract their high fanout.

The success of the algorithm depends on two conditions. First, within a plane the lowest τ_{leak} must be greater than the highest τ_{switch}. However, it is not enough for every plane to have the required separation between $\min(\tau_{leak})$ and $\max(\tau_{switch})$, this separation must also exist over all planes. The lowest τ_{leak} *over all planes* must be two orders of magnitude above the highest τ_{switch} *over all planes.*

VMATCH, therefore, is a two step algorithm that first coordinates over all planes to find the slowest feasible on delay, $\tau_{switchFeasible}$. It then iterates over each plane matching functions to resources with the goal of bettering, if not at least meeting, this target so that the plane's $\max(\tau_{switch})$ is at or below $\tau_{switchFeasible}$ and its $\min(\tau_{leak})$ is at or above 100 times this target. If all planes meet this condition, the mapping is successful and achieves a delay at least equal to $\tau_{switchFeasible}$ or better. It is later explained why if a plane fails to meet this target, the overall mapping fails.

$\tau_{switchFeasible}$ determines the slowest possible delay for a successful mapping. It is calculated by first finding the slowest mapping for each plane and then choosing the slowest $\max(\tau_{switch})$ over these slow mappings. Within a plane, the slowest mapping is computed by assigning the function with the lowest fanout to the slowest resource, the second lowest fanout function to the second slowest resource and so on.

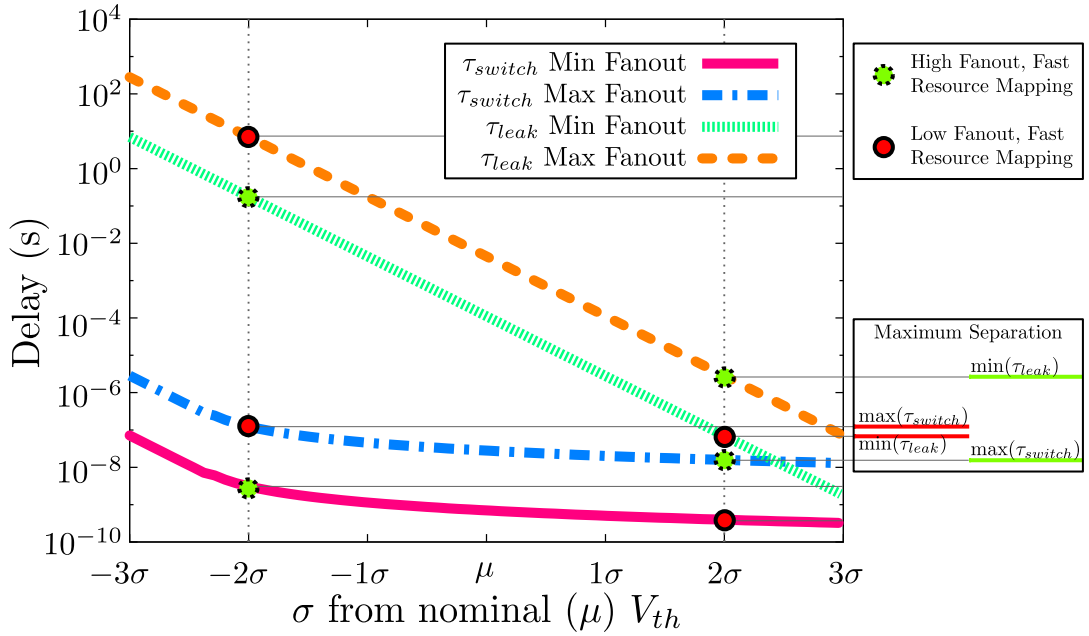


Figure 4.7: τ_{switch} and τ_{leak} ranges over $\pm 3\sigma$ of nominal V_{th} for high fanout and low fanout functions. Two resources highlighted at $\pm 2\sigma$. Green points show the result of mapping high fanout functions to fast resources and low fanout functions to slow resources. Red points show the opposite result, high fanout to slow resources and low fanout to fast resources. Green's separation is over two orders of magnitude while there is no separation for red since $\max(\tau_{switch})$ is above $\min(\tau_{leak})$.

The reason for assigning functions in this order instead of assigning the highest fanout function to the slowest resource (which would give a slower τ_{switch} for the first mapping, Equation 3.2) is so that the mapping does not violate the two orders of magnitude separation required.

To explain this, consider the extreme example shown in Figure 4.7. Here, the plane has two functions, one with $MaxFanout$ and one with $MinFanout$. Also, there are only two resources a fast resource with V_{th} at $+2\sigma$ and a slow resources with V_{th} at -2σ. The green points show the τ_{switch} and τ_{leak} achieved for mapping the high fanout function to the fast resource and the low fanout function to the slow resource. On the right side we see that the separation between $\min(\tau_{leak})$ and $\max(\tau_{switch})$ is over two orders of magnitude, this would be a successful mapping. On the other hand, consider what happens when we map the high fanout function to the slow resource and the low fanout function the fast resource. The red points show this results. Again, looking at the Maximum Separation, we see that there is no separation whatsoever since $\max(\tau_{switch})$ is above $\min(\tau_{leak})$. Therefore, even though mapping a high fanout function to a slow resource gives the highest τ_{switch}, when the remaining low fanout functions use fast resources, the separation actually diminishes. As such, all mappings on the NanoPLA need to occur in a "high fanout function to fast resource" fashion.

Once $\tau_{switchFeasible}$ is computed, each plane can independently compute its mapping. The mapping from

functions to resources is done by creating a bipartite graph between functions and resources where a function is assigned to a resource if and only if the result of mapping the function to that resource is one where the resulting $\tau_{switch} \leq \tau_{switchFeasible}$ and $\tau_{leak} \geq 100 \times \tau_{switchFeasible}$. A mapping on the plane is given by a bipartite matching that assigns each function to a unique resource. One way to solve for this matching is by computing the maximum correspondence maximum weight bipartite matching where the edges between functions and resources are given a weight equal to the negative of τ_{switch} that would resulting from applying the mapping defined by the edge. By assigning negative τ_{switch} as the weights of the edges, we guarantee that the maximum correspondence maximum weight solution returns a mapping with the fastest $\max(\tau_{switch})$ for the functions in that plane. Efficient solutions to the maximum correspondence maximum weight problem are presented in [42]. Nevertheless, we present a more efficient greedy heuristic that produces results comparable to the matching produced by the maximum correspondence maximum weight solution.

The greedy algorithm works by assigning the function with the highest fanout to the fastest resource it can map to as marked in the bipartite graph. Once this mapping has been assigned, any other edge incident to that resource node is removed. Then the second highest fanout function is assigned the mapping to the fastest resource it can use based on the remaining edges in the graph. The process repeats until all functions are assigned to a resource. This greedy heuristic is guaranteed to find a solution because of the way $\tau_{switchFeasible}$ is defined. Since $\tau_{switchFeasible}$ is the slowest τ_{switch} from the slowest mapping over every plane, the algorithm is guaranteed to always at least find this solution. However, by assigning fastest resources first, we can get a solution that is significantly faster than $\tau_{switchFeasible}$ while still maintaining the two orders of magnitude separation.

A further optimization is possible where construction of the bipartite graph is not necessary. By ordering all resources from fastest to slowest, starting with the function with the largest fanout, we iterate over the ordered resources until a resource is found that maintains $\tau_{switch} \leq \tau_{switchFeasible}$ and $\tau_{leak} \geq 100 \times \tau_{switchFeasible}$ or until the number of resources remaining equals the number of functions not yet mapped. Then, the function is assigned to the resource and the algorithm continues searching through the remaining resources with the next highest fanout function. Thus, by ordering the resources and considering each only once, we can find the best possible matching for the given $\tau_{switchFeasible}$.

The reason why all remaining resources do not have to be considered for every function is because once a resource is rejected by a function with fanout f, it will be rejected by any function with fanout $\leq f$ since τ_{leak} will be even faster for a resource with lower fanout. We can see this in Figure 4.8. Assuming now that we have more than 2 resources, distributed over $\pm 3\sigma$, and that only one of the two function has been mapped, the $MaxFanout$ function to the resource at $+2\sigma$. The second function has a fanout of $MinFanout$. If the second function were to use a resource at or above $+2\sigma$, the separation that the fist resource achieved would be reduced. In fact, as highlighted, for a function with $MinFanout$, any resource above $+1\sigma$ will

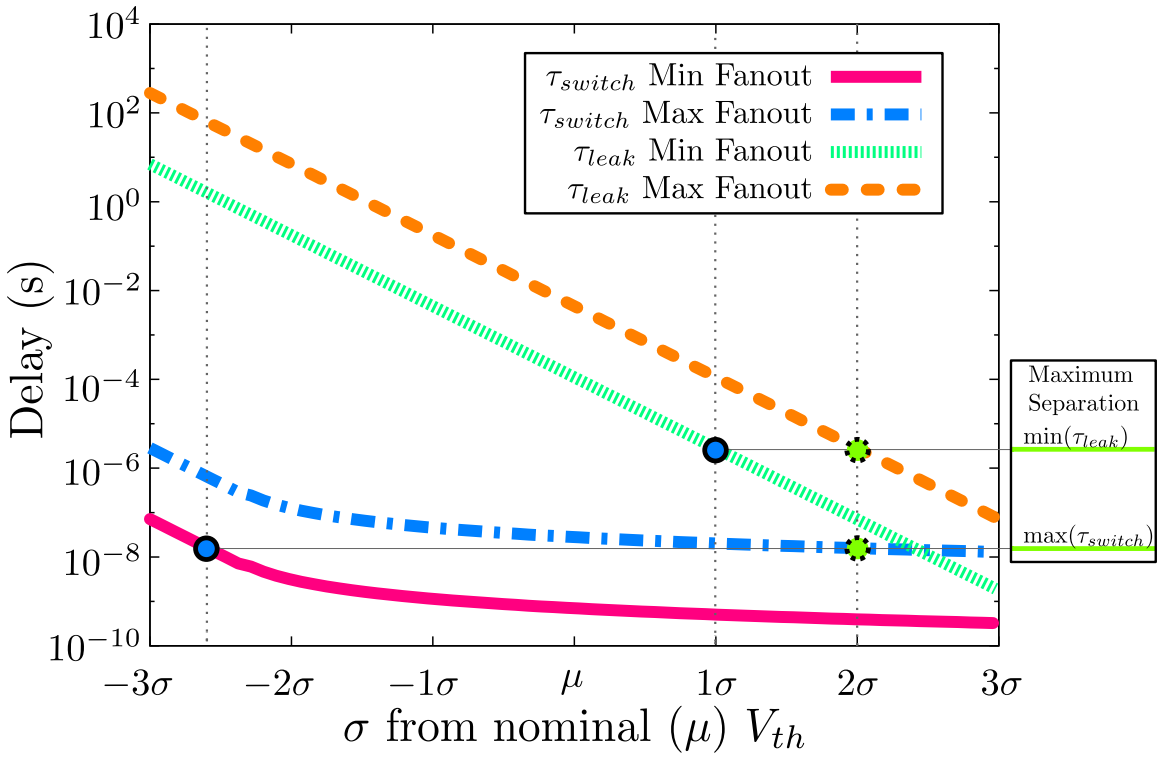


Figure 4.8: τ_{switch} and τ_{leak} ranges over $\pm 3\sigma$ of nominal V_{th} for high fanout and low fanout functions. One resources highlighted at $+2\sigma$, one at $+1\sigma$ and one at -2.7σ. Green points show the result of mapping one high fanout function to a fast resources.

reduce $\min(\tau_{leak})$ and as a consequence, the separation. Thus all the resources the first function had rejected will clearly also not work for a function with a lower fanout. In general, by sorting resources from fastest to slowest and starting with the highest fanout function, we know that if a function rejected a resource, all the functions that still need mapping will also reject that resource. However, observe that if we use a resource that is too slow, we can also reduce the separation. As highlighted in Figure 4.8, if that second function uses a resource below approximately -2.7σ, $\max(\tau_{switch})$ will increase, which is one reason why we map functions to the fastest remaining resource. Finally, by forcing a mapping when the number of resources remaining equals the number of function remaining, we guarantee that at least we find the slowest solution as was computed for finding $\tau_{switchFeasible}$ initially.

This heuristic is guaranteed to find the fastest matching. The delay of a mapping is determined by the slowest τ_{switch}. Figure 4.9 shows what τ_{switch} is for a high fanout function and a low fanout function over $\pm 3\sigma$ range. Faster resources are those towards the right of the graph and become monotonically slower towards lower σ. The heuristic above makes sure to always give the fastest resource available to the largest fanout function. To understand why this leads to the fastest mapping, examine the following example. Assume two resources, one at $+2\sigma$ and one at -1σ as highlighted in the figure. Let us consider that instead

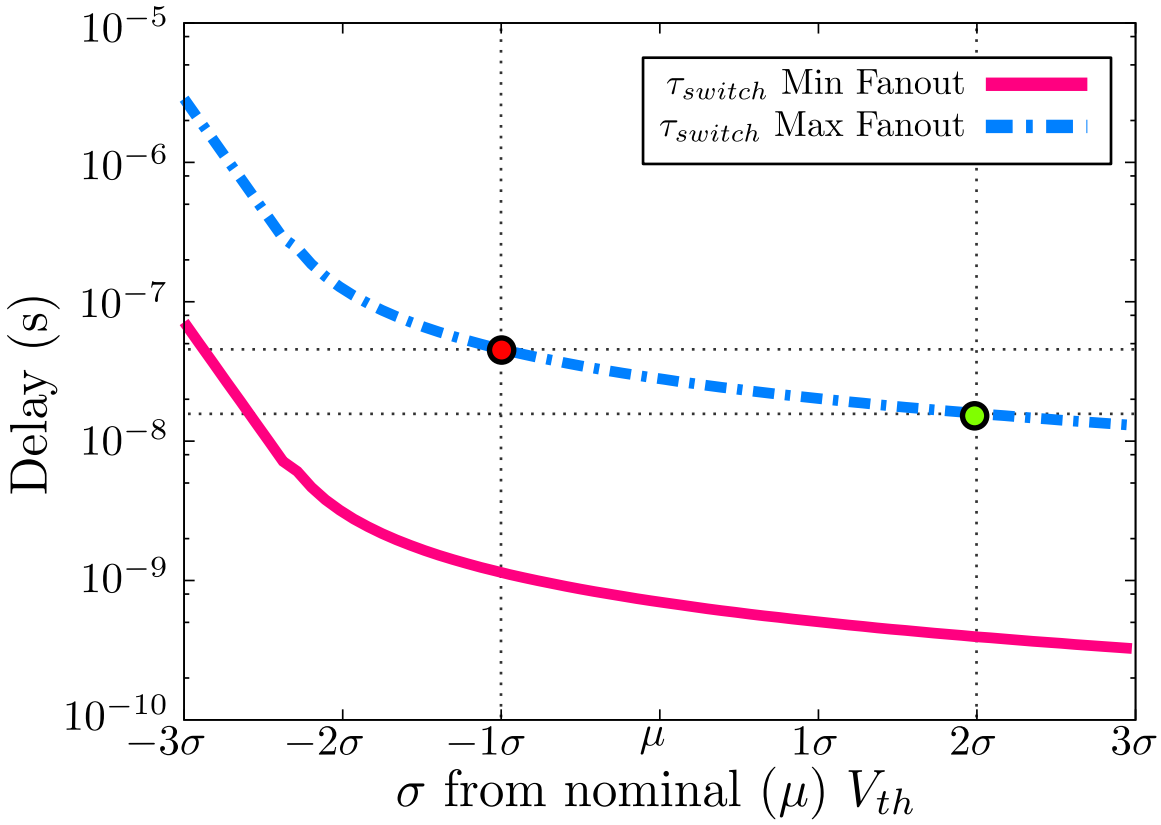


Figure 4.9: τ_{switch} range over $\pm 3\sigma$ of nominal V_{th} for high fanout and low fanout functions.

of following the heuristic, the high fanout is not assigned to the fast resource but instead a function with lower fanout is mapped to this fastest resource. Since the delay increases for slower resources, this mapping will indeed give that function its fastest possible delay. This, however, forces the largest fanout function to the slower resource. Compounded by the large fanout, this leads to a very high delay (Red point in the figure). Following VMATCH leads to mapping the high fanout on the fast resource, which gives a lower delay for that function (Green point). The lower fanout function, when mapped to the slow resource, does not change the maximum delay. At the extreme case, where we have one resource at $+3\sigma$ and one at -3σ, it is possible that a lower fanout function could be forced to use a very slow delay and cause the overall delay to worsen. In this second example, the MinFanout function would force to use the -3σ resource. Nevertheless, consider the alternative of allowing it to use the fast resource. Again, this would force the high fanout to this extremely slow resource, which would result in an even worse delay. Therefore, assigning the fast resources to the high fanout functions grantees that the overall mapping is fastest.

Algorithm 4.1 shows VMATCH in detail. First $\tau_{switchFeasible}$ is calculated in $upperBound()$. Then, for each plane, $mapPlane()$ computes a mapping based on $\tau_{switchFeasible}$. For each plane, $upperBound()$ computes the slowest feasible mapping by mapping the slowest unused resource to the lowest fanout function

until all function have been assigned to a resource. Then the slowest overall τ_{switch} from these mappings is assigned to $\tau_{switchFeasible}$. Within a plane, *mapPlane*() then tries to find a mapping that meets the $\tau_{switchFeasible}$ boundary requirements. Starting with the highest fanout function, it iterates over the resources from fastest to slowest. It assigns the function to the first resource that meets the **if** condition and continues with the next highest fanout function, considering the next resource.

Algorithm 4.1: VMATCH

```
VMATCH()
    τ_switchFeasible = upperBound(Planes)
    clearAllMappings()
    foreach Plane P ∈ Planes do
        P.mapPlane(τ_switchFeasible)
    end

upperBound(Planes)
    foreach Plane P ∈ Planes do
        for i ← 1 to P.numFunctions() do                    /* Compute Slowest Mapping */
            function = P.nextLowestFanoutFunction()
            resource = P.nextSlowestResource()
            P.map(function, resource)
        end
    end
    τ_switchFeasible =Max(τ_switch(Planes)) ;                    /* Find Slowest τ_switch */
    return τ_switchFeasible

mapPlane(τ_switchFeasible)
    function = highestFanoutFunction()
    foreach resource ∈ OrderedResources do          /* Ordered from fastest to slowest */
        τ_switch = onDelay(function, resource)
        τ_leak = offDelay(function, resource)
        if (τ_switch ≤ τ_switchFeasible and τ_leak ≥ 100 × τ_switchFeasible)
          or (numRemainingResource() == numRemainingFunctions()) then
            map(function, resource)
            function = nextHighestFanoutFunction()
        end
        if allFunctionsMapped() then
            return Success
        end
    end
    return Failure
```

The run time for VMATCH is $O(r \log(r))$ where r is the total number of resources in the NanoPLA. For convenience, let f be the total number of functions that will be mapped. *upperBound*() orders all resources and function and then iterates over every function that will be mapped, to find $\tau_{switchFeasible}$. This takes $O(r \log(r) + f \log(f))$ to sort functions and resources and $O(f)$ for the mapping. At worst, every resource is explored once in *mapPlane*(). Since resources and function are also ordered *mapPlane*() takes

$O(r\log(r) + f\log(f) + r)$. Overall this means that VMATCH takes $O(r\log(r) + f\log(f) + r + f)$. However, the number of resources must be greater than or equal to the number of functions, else there would not be enough resources to map all functions. Therefore VMATCH runs in $O(r\log(r))$.

A mapping can be not feasible in two ways, both of which can be detected during the first phase of the algorithm. In a similar way to how we reasoned about why the greedy mapping produces the fastest mapping, we can argue that the slowest mapping, as used to calculate $\tau_{switchFeasible}$, produces a mapping with the widest separation between τ_{switch} and τ_{leak}. Figure 4.10 has three highlighted resources and two functions. It shows the two possible mappings that could happen by mapping slow resource to low fanout functions first. One is represented by the green points and the other one, by the blue points. It is clear that the widest separation is given by the green mapping. This is also the slowest possible mapping. Although $\max(\tau_{switch})$ is faster for the blue mapping, the linear vs exponential nature of τ_{switch} Vs. τ_{leak} means that loosing a small amount of separation due to a slower $\max(\tau_{switch})$ translates into gaining an exponential increase in separation due to the higher $\min(\tau_{leak})$. Thus, slower resource can sacrifice a slower τ_{switch} for a significantly larger τ_{leak}. The result of using the slowest resources, means that the separation will be greatest. Therefore, the slow mappings computed by $upperBound()$ will be the mapping with the largest separation. If this separation is less than two orders of magnitude, then there is no mapping that will produce the required separation, since any other mapping would have to use faster resources and Figure 4.10 shows that faster resources have smaller separations. Thus the mapping will not be feasible if the slowest mapping of any plane does not have the required separation.

Even if every plane has at least one mapping that achieves a separation of 100, it might still be impossible to map the design as placed and global-routed. This occurs when, between two planes, regardless of the mapping used in each, it is never the case that when considered together, $\min(\tau_{leak}) \geq 100 \times \max(\tau_{switch})$. Detecting this kind of problem requires only a little more work than what $mapPlane()$ is already doing. $\tau_{switchFeasible}$ is defined as the slowest achievable τ_{switch} when every plane is mapped to the slowest resources as previously explained. In a similar way we can define $\tau_{leakFeasible}$, the fastest τ_{leak} when every plane is mapped to the slowest resources. This can be computed along side $\tau_{switchFeasible}$ by augmenting $mapPlane()$ to also compute $\tau_{leakFeasible}$ =Min($\tau_{leak}(Planes)$). For the same reason that within a plane the slowest mapping gives the widest separation, the widest separation between all planes is given by the separation between $\tau_{leakFeasible}$ and $\tau_{switchFeasible}$. If this separation is less than 100, then there are at least two planes that are incompatible and will not allow a mapping to occur.

Both within plane failures and between plane failures can be overcome using the same two techniques. The first technique is simply to widen the minimum channel width, $MinC$, calculated by the global route (Figure 3.2), by adding more resources to every plane. By increasing the number of resources, we increase the probability that the mapping will be able to use resources that lead to a solution. This approach is

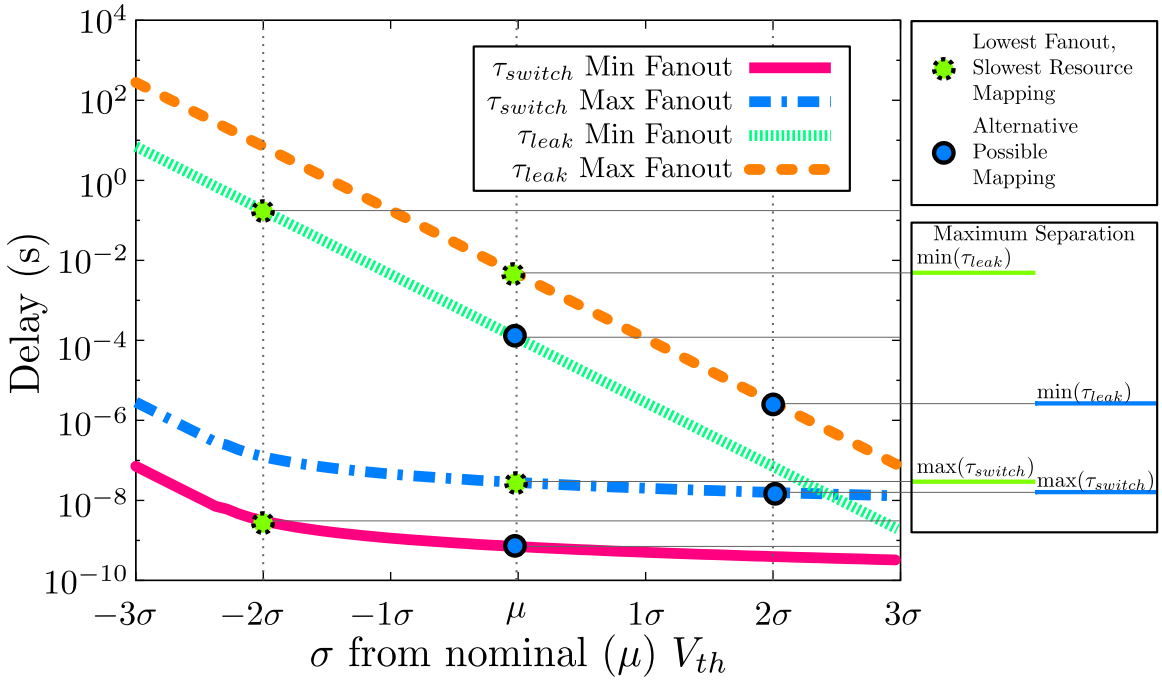


Figure 4.10: τ_{switch} and τ_{leak} ranges over $\pm 3\sigma$ of nominal V_{th} for high fanout and low fanout functions. Three resources highlighted. Green points show the result of the slowest mapping. The only other possible mapping that respects the low fanout to slow resource constraint is shown with blue points.

not without a downside. Adding resources forces the nanowires to be longer so that the crossbars are still complete. Longer wires lead to more area, energy and most importantly longer wire delays. If we add too many wires, the general separation between τ_{switch} and τ_{leak} will decrease. Therefore it is best to add a modest amount of resources, enough to improve the probability of a successful mapping.

The second method involves running higher level CAD tools to reduce the burden on the failing planes by re-placing and/or re-routing functions. Moving functions away from a plane has a positive effect similar to the first technique. In essence, the remaining functions have more resources to choose from. This mechanism does not increase the area but, depending on the way functions are moved, the total energy and delay may change. In this work we focus on the first technique by adding more resources until we reach 100% yield. As shown in Chapter 6, VMATCH only needs a modest amount of extra resources, and this minimally affects the delay energy and area. A full probabilistic analysis of how many resources should be added, as well as the re-placement method of dealing with failures is left as future work.

Figure 4.11 shows the results of using VMATCH to map the same chip shown in Figure 4.1. Clearly Equation 3.1 holds and the mapping is defect free. However, unlike the conservative mapping (Figure 4.3), this mapping achieves a lower max(τ_{switch}) and only requires 3% extra resources over MinC.

To build intuition for this success, Figure 4.12 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

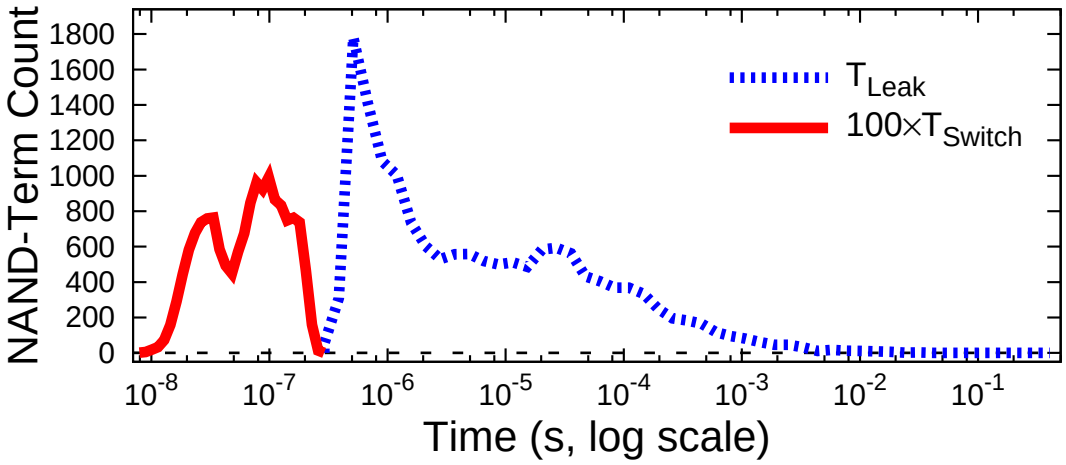


Figure 4.11: Distribution of τ_{leak} and $100 \times \tau_{switch}$ of VMATCH mapping. Benchmark spla at $\sigma = 38\%$

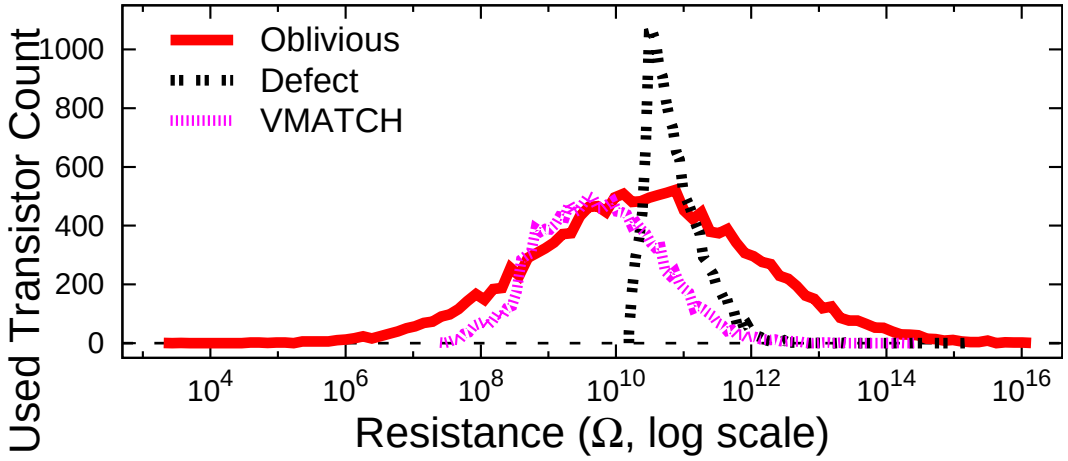


Figure 4.12: Used R_{offFET} distribution for Oblivious, Defect and VMATCH algorithms. Benchmark spla at $\sigma = 38\%$

(Figure 4.1). The defect-avoiding algorithm chooses resources that are above the conservative threshold. Figure 4.3 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 Figures 4.11 and 4.12 demonstrate.

Chapter 5

Device Characterization

VMATCH depends on the ability to characterize R_{FET} of the transistors. In this chapter 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 resistance.

All restore wires are connected to source and ground voltage terminals. (*Source Voltage* and *Measurement Voltage* in Figure 5.2). 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 Figure 5.2, 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 (Figure 5.2). Compute wires are encoded with a k-hot code [31] 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*, Figure 5.2). We can turn all restore transistors off by selecting all compute wires at once and applying a high voltage through them. The charge used to turn off the transistors can be stored by isolating it between the decoder on one end and the precharge transistor on the opposite end (Figure 5.2). 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 Section 3.2, under certain variation conditions, an off transistor can leak faster than an on transistor can switch. Second, the nanowires have high resistance. When

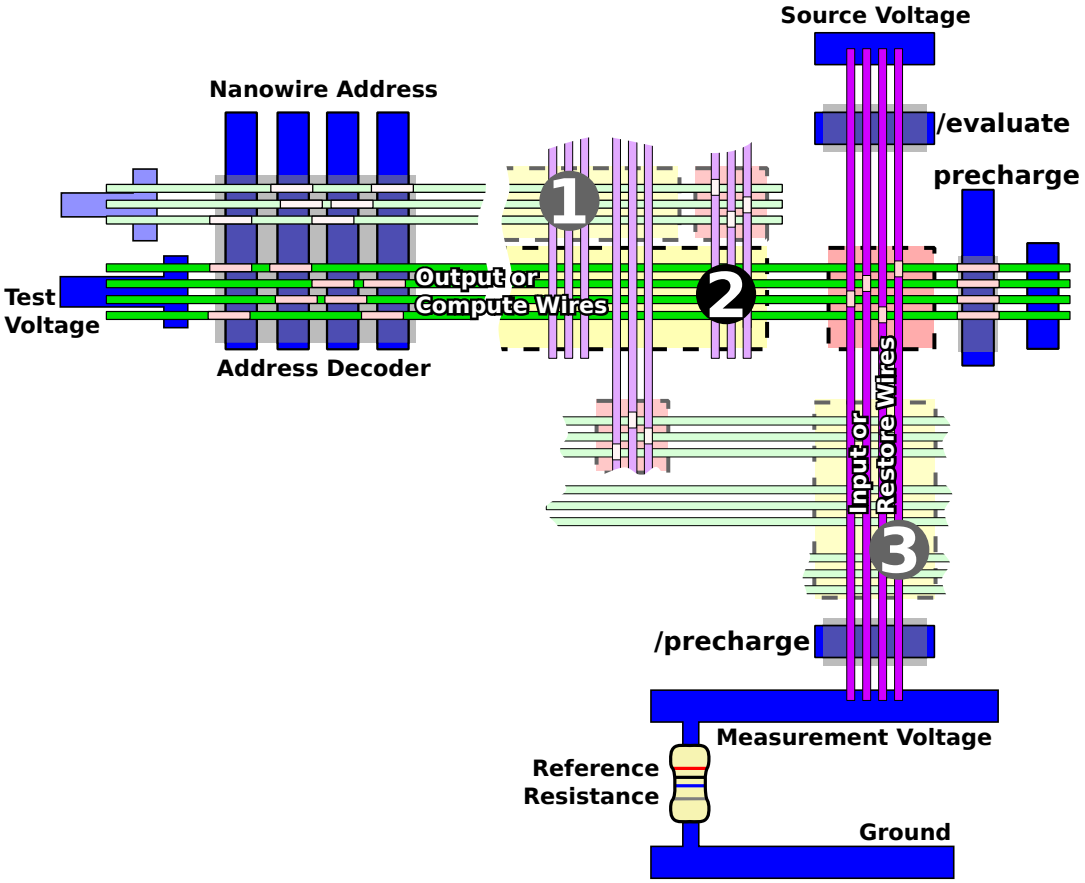


Figure 5.1: NanoPLA Block with Address Decoder

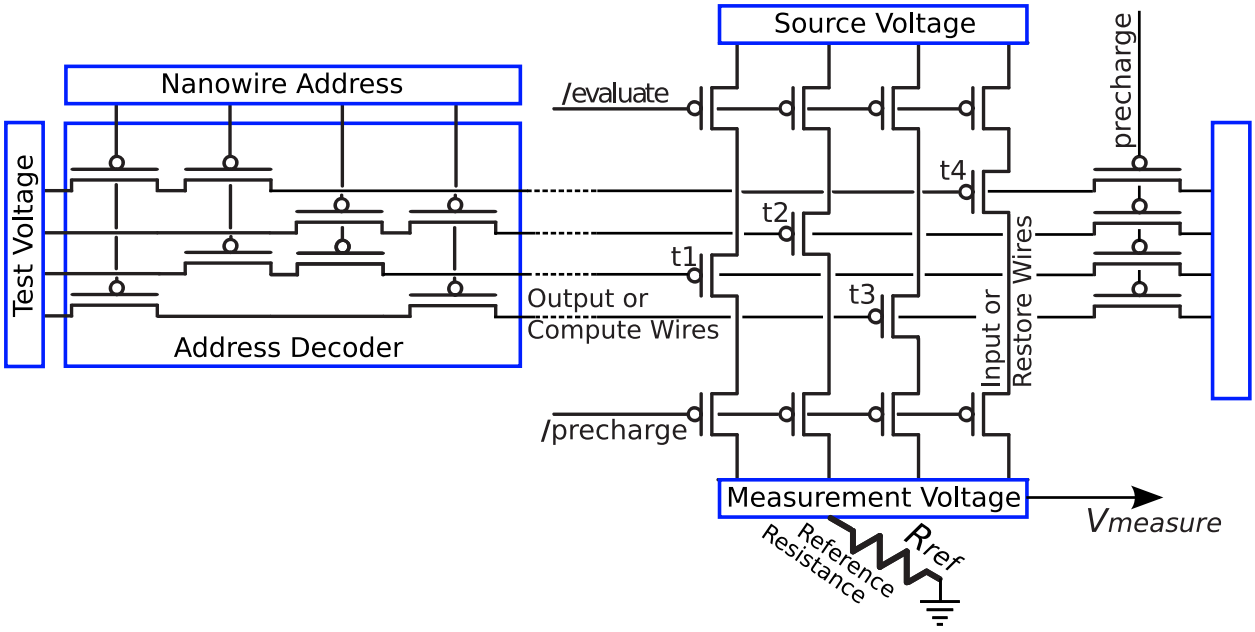


Figure 5.2: NanoPLA Inversion Array Circuit with Address Decoder

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 do not provide the information we need for VMATCH.

To deal with the first problem, we strongly turn off all transistors. Equation 4.2 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. Performing a binary search on the Test Voltage terminal (Figure 5.2) reveals the test voltage that charges the output to a predefined percent of the source voltage. This value has a direct relationship to the actual V_{th} of the transistor, producing the sufficient information for VMATCH.

Section 5.1 gives a high level overview of how this measurement technique works. Section 5.2 formally explains the circuit model. We examine how to correctly configure a plane in the NanoPLA in Section 5.3. This leads to a discussion on how to set the reference resistance and the time it takes to perform this testing (Section 5.4). The details of the measurement algorithm are explained in Section 5.5. The impact of limited measurement precision is examined in Section 5.6.

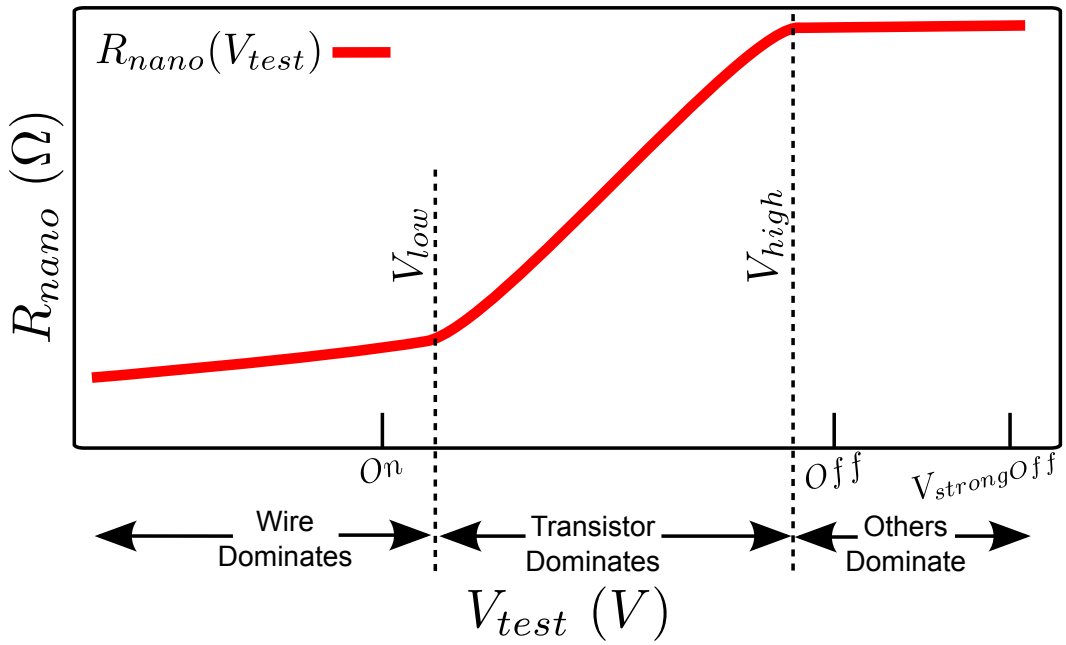


Figure 5.3: Expected R_{nano} as a function of V_{test} showing where R_{FET} dominates and where R_{inWire} dominates

5.1 Overview of measurement steps

Measuring each restore wire requires a series of initialization steps 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.

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 compute nanowires. Then we apply a higher than normal voltage $V_{test} = V_{strongOff}$ so that the transistors on the restore wires are *strongly* turned off. The charge is then isolate between the precharge transistors and the decoder so that the wires remain strongly turned off during the following measurement step. Section 5.3.2 explains how to choose $V_{strongOff}$ so that it is 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, and use the test voltage terminal to search for the value of V_{test} that causes $V_{measure}(V_{test})$ to reach a predefined percent of V_{dd}. This occurs at $R_{nano}(V_{test} = V_{settled})$, where $V_{settled}$ is defined as the aforementioned V_{test}. Section 5.5 explains how once $V_{settled}$ is determined, we can directly find the V_{th} of the transistor.

$R_{nano}(V_{setteld})$ will be composed of three resistances. The series combination of the actual wire resistance,

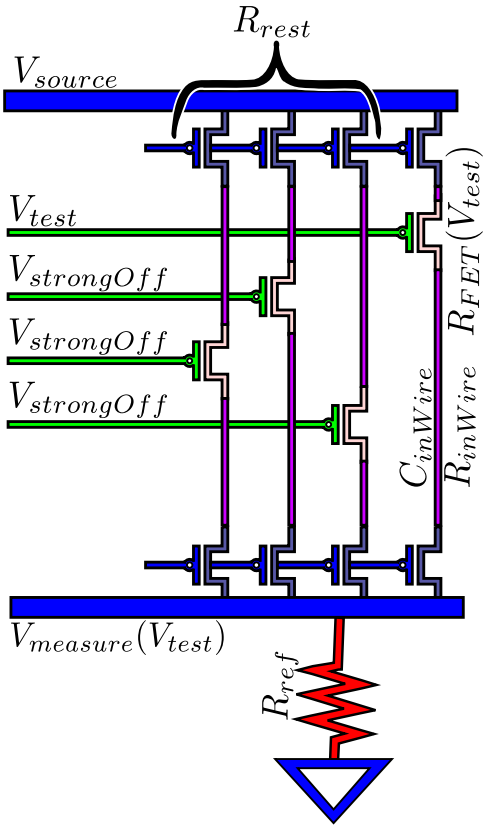


Figure 5.4: Details of the NanoPLA Voltage Divider Measurement Circuit

R_{inWire} and the transistor resistance, R_{FET} (either on or off, depending on the value of V_{test}) in parallel with the parallel resistance of all the other wires not being tested, R_{rest} (Figure 5.4).

$$\frac{1}{R_{rest}} = \sum_{i \neq \text{test wire}} \left(\frac{1}{R_{nano_i}(V_{strongOff})} \right) \tag{5.1}$$

Equation 5.2 shows the composition of R_{nano}, the series resistances of the wire followed by the transistor all together in parallel with the resistance of the other wires.

$$\frac{1}{R_{nano}(V_{test})} = \frac{1}{R_{FET}(V_{test}) + R_{inWire}} + \frac{1}{R_{rest}} \tag{5.2}$$

Plotting the measured R_{nano} as a function of V_{test} would look similar to Figure 5.3. At $V_{test} = V_{strongOff}$ the leakage from the other wires may dominate. As V_{test} decreases, the transistor resistance dominates R_{nano} 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_{nano}. The value of $V_{setlled}$ is guaranteed to be in the section of Figure 5.3 where the transistor dominates, which is why it can be used to estimate V_{th}.

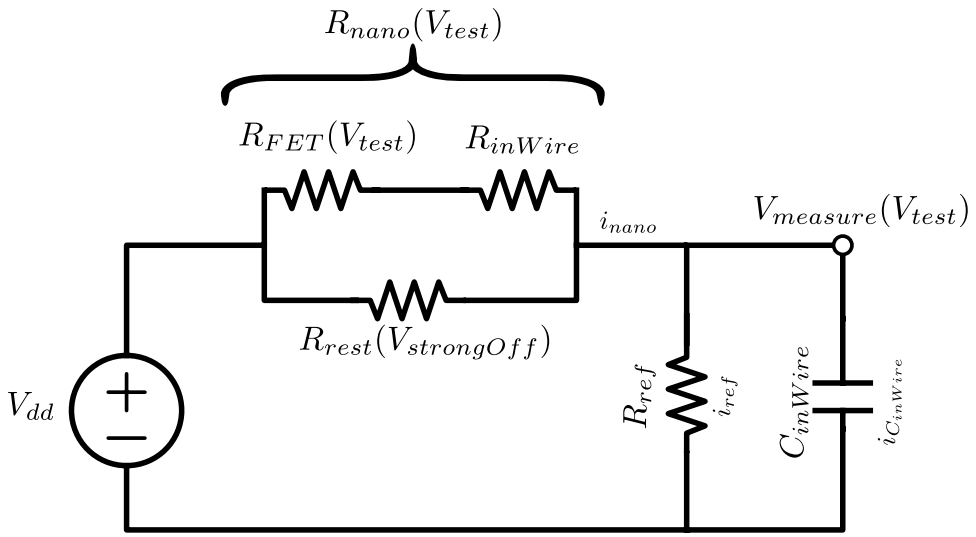


Figure 5.5: NanoPLA Voltage Divider Circuit Diagram

5.2 Circuit Model

Once all wires have been strongly turned off, we can take measurements at multiple V_{test}. To better frame the discussion of how this technique gives us the measurement information needed, we take a formal look at what the test circuit looks like. Figure 5.5 shows the equivalent circuit diagram for the proposed NanoPLA test circuit. As shown, we let $R_{nano}(V_{test})$ be the equivalent resistance of the nanowire components, including R_{FET}, R_{inWire} and R_{rest}. In order to solve for $V_{measure}(V_{test}, t_{measure})$, we set up the following linear circuit equation:

$$i_{nano} = i_{ref} + i_{C_{inWire}}$$

$$\frac{V_{dd} - V_{measure}(V_{test})}{R_{nano}(V_{test})} = \frac{V_{measure}(V_{test})}{R_{ref}} + C_{inWire} \cdot \frac{d}{dt} V_{measure}(V_{test})$$

Solving this equation yields an answer for $V_{measure}(V_{test}, t_{measure})$ shown in Equation 5.3

$$V_{measure}(V_{test}, t_{measure}) = \frac{R_{ref}}{R_{nano} + R_{ref}} \cdot V_{dd}(1 - e^{-\frac{t_{measure}}{RC}}) \tag{5.3}$$

where RC is defined as

$$RC = \frac{R_{nano} \cdot R_{ref} \cdot C_{inWire}}{R_{nano} + R_{ref}} \tag{5.4}$$

and R_{nano} is

$$R_{nano}(V_{test}) = \frac{(R_{FET}(V_{test}) + R_{inWire}) \cdot R_{rest}}{R_{FET}(V_{test}) + R_{inWire} + R_{rest}} \tag{5.5}$$

To fully grasp what these equations imply, we will first focus on the resistance of the voltage divider connected to the source voltage, *i.e.* the resistance from the nanowires being measured. That analysis will then help determine what the reference resistance, R_{ref}, should be.

5.3 Upper Resistance: NanoPLA Plane Resistance

Our goal is to measure $R_{FET}(V_{test})$ and from that, compute what the transistor's V_{th} is. From Figure 5.5 it is evident that if R_{rest} can be significantly greater than $R_{FET}(V_{test}) + R_{inWire}$, then $R_{FET}(V_{test}) + R_{inWire}$ will be the dominant resistance. Furthermore, choosing a V_{test} that causes $R_{FET}(V_{test})$ to be greater than R_{inWire}, but still less than R_{rest}, results in R_{FET} dominating R_{nano} and thus a measurement of R_{nano} will be nearly equal to measuring R_{FET}. This sets the following constraints on the values R_{FET} should take in order for it to dominate R_{nano}.

$$R_{rest} \geq \quad R_{FET}(V_{test}) > \quad R_{inWire} \tag{5.6}$$

The region where R_{FET} dominates is bounded by $V_{test} = V_{low}$, defined by the boundary between the wire dominating and the transistor dominating, and $V_{test} = V_{high}$, above which the other wires start to dominate (Figure 5.3). We are free to define what V_{high} is since R_{rest} can be adjusted by raising or lowering $V_{strongOff}$. V_{low} however, is fixed for each resource and determined completely by the variation in that resource. Since V_{low} cannot be controlled, we first explain what V_{low} is and then focus on how to define V_{high} and set $V_{strongOff}$ so that there is a wide region over which R_{FET} dominates.

5.3.1 Understanding V_{low}

When $R_{FET}(V_{test}) > R_{inWire}$ in Equation 5.5, the transistor's resistance dominates, and when $R_{FET}(V_{test}) < R_{inWire}$, the wire dominates. Thus V_{low} will be the value of V_{test} that makes $R_{FET}(V_{test}) = R_{inWire}$. This can happen when the transistor is in the subthreshold region or when it is in the saturation region, thus we need to consider both cases.

$$R_{inWire} = \begin{cases} \dfrac{V_{dd}}{I_{sat}(V_{test})} & \text{Transistor in saturation region} \\ \\ \dfrac{V_{dd}}{I_{sub}(V_{test})} & \text{Transistor in subthreshold region} \end{cases} \tag{5.7}$$

To simplify the analysis, we rewrite the saturation and subthreshold current equations for P-type transistors (Equations 4.1 and 4.2) as follows:

$$I_{sat} = I_0 \cdot (V_{th} - V_{gs}) - I_1 \tag{5.8}$$

$$I_{sub} = I_2 \cdot 10^{\frac{V_{th} - V_{gs}}{S}} \tag{5.9}$$

Substituting Equations 5.8 and 5.9 into Equation 5.7,

$$R_{inWire} = \begin{cases} \dfrac{V_{dd}}{I_0 \cdot (V_{th} - V_{test}) - I_1} & \text{Transistor in saturation region} \\ \\ \dfrac{V_{dd}}{I_2 \cdot 10^{\left(\frac{V_{th} - V_{test}}{S}\right)}} & \text{Transistor in subthreshold region} \end{cases} \tag{5.10}$$

and solving for V_{test} leads to:

$$V_{test} = \begin{cases} V_{th} - \dfrac{V_{dd}}{I_0 \cdot R_{inWire}} - \dfrac{I_1}{I_0} & \text{Transistor in saturation region} \\ \\ V_{th} + S \cdot \log\left(\dfrac{V_{dd}}{I_2 \cdot R_{inWire}}\right) & \text{Transistor in subthreshold region} \end{cases} \tag{5.11}$$

When the wire's resistance is comparable to the resistance of the transistor when the transistor is in the saturation region, then, the V_{test} at which this occurs will be strictly less than the threshold voltage V_{th} of the transistor, as shown in Equation 5.11. Thus, for the saturation case, V_{low} is bounded by V_{th}. This is consistent with the definition of saturation for a PMOS transistor.

If on the other hand, the wire resistance equals the transistor resistance when V_{test} causes the transistor to be in the subthreshold region, then, Equation 5.11, again in accordance with the definition, shows that V_{test} will be greater than V_{th}. To be able to bound V_{low} by V_{th}, R_{inWire} must be less than or equal to $R_{FET}(V_{th})$. This only happens when the resistance of the wire is on the order of the resistance of the transistor in the saturation region.

For the technology being considered, $R_{FET}(V_{th}) = 2M\Omega$, thus as long as $R_{inWire} \leq 2M\Omega$ then V_{low} will be bounded by V_{th}. The value of R_{inWire} is defined, as shown in Equation 5.12, by the length and cross-sectional area of the wire, l and a, the resistivity, ρ, of the material, and the variation of that wire $n_{R_{inWire}} \cdot \sigma_{R_{inWire}}$.

$$R_{inWire} = \frac{l \cdot \rho}{a} + n_{R_{inWire}} \cdot \sigma_{R_{inWire}} \tag{5.12}$$

As explained in section 2.1, we examine a technology where the wire is nickel silicide in the regions where it does not interact with perpendicular wires and silicon otherwise. This means that ρ changes throughout the length of the wire. The length is determined by the architecture and channel width, a conservatively long wire is on the order of 30μm. The cross-sectional area is that of a wire with a diameter of 5nm. Finally, we consider $\sigma_{R_{inWire}} = 38\%\mu_{R_{inWwire}}$, *i.e.* 38% of the mean wire resistance. These parameters lead to a nominal $\mu_{R_{inWire}} = 54k\Omega$. From this and Equation 5.12, we conclude that in order for the wire resistance to dominate the transistor resistance when $V_{test} = V_{th}$, the wire must be 95 standard deviations above the mean. The probability of this event is so infinitesimal that, for this technology, we can safely bound V_{low} by the threshold voltage V_{th} of the transistor we are trying to measure. Equation 5.13 formally defines V_{low} in terms of the nominal V_{th}, the amount of V_{th} variation of the transistor, n_{wire}, and $\sigma_{V_{th}}$, the standard deviation of the V_{th} distribution.

$$V_{low} = V_{th_{nominal}} + n_{wire} \cdot \sigma_{V_{th}} \tag{5.13}$$

From Equation 5.13, it is clear that although we can bound V_{low}, since V_{th} varies from transistor to transistor, V_{low} will be different for every wire. Fundamentally, this still leaves the original challenge of measuring V_{th}. However, it does helps frame the discussion below.

5.3.2 Defining V_{high} and Setting $V_{strongOff}$

V_{high} defines the boundary above which the other wires in the NanoPLA plane dominate and below which the transistor of the wire being measured dominates (Figure 5.3). $V_{strongOff}$ can be used to control where this boundary lays. To make sure we can measure most of the transistors, $V_{strongOff}$ must be high enough so that $R_{rest}(V_{strongOff})$ is large even for a transistor with a V_{th} that is n sigma below the nominal V_{th}. It should still be the case that its V_{high} satisfies the constraint $R_{rest}(V_{strongOff}) \geq R_{-n\sigma FET}(V_{high})$. In fact, it is the case that for the lowest V_{high} expected over all transistors, $V_{strongOff}$ should be set so that $R_{rest}(V_{strongOff})$ equals $R_{-n\sigma FET}(V_{high})$. The value of $V_{strongOff}$ comes directly from this relationship, but first we must determine what the lowest expected V_{high} will be.

The lowest expected V_{high} will occur in the wire with the lowest expected V_{th} which will be a wire that has a V_{th} n sigma below the nominal V_{th}. In the previous section we conservatively defined V_{low} to be equal to the V_{th} being measured, this implies that for the transistor with the lowest expected V_{high}, its V_{low} will equal $V_{thNominal} - n \cdot \sigma_{V_{th}}$. Since the transistor dominates Equation 5.5 between V_{low} and V_{high}, the lowest expected V_{high} has to be greater than $V_{thNominal} - n \cdot \sigma_{V_{th}}$; however, it is not immediately clear how much greater. The fact that we are trying to measure R_{offFET} will help clarify how much greater. R_{offFET} is equivalent to the value of the transistor when $V_{test} = V_{dd}$. Thus, the transistor does not have to dominate Equation 5.5 when V_{test} is above V_{dd} and we can define V_{high} for the transistor n sigma below nominal to

be V_{dd}.

With the lowest expected V_{high} established, we can calculate what $V_{strongOff}$ is. Remember that R_{rest} is the parallel sum of all wires not being measured, with their transistors strongly turned off. If R_{rest} is composed of m wires, then we can bound the value of R_{rest} below by the value of the leakiest resource in R_{rest} divided by m and above, simply by the leakiest resource. The leakiest resource will be the resource with the transistor that is n sigma above the nominal V_{th}. Equation 5.14 formally shows this bound for measuring wire j.

$$\frac{1}{R_{+n\sigma FET}(V_{strongOff})} \leq \sum_{i \neq j}^{m} \frac{1}{R_{FET_i}(V_{strongOff})} \leq \frac{m}{R_{+n\sigma FET}(V_{strongOff})} \tag{5.14}$$

Combining this bound with the fact that the V_{high} of the wire with the transistor n sigma below nominal is V_{dd}, we can now solve for $V_{strongOff}$.

$$R_{+n\sigma FET}(V_{strongOff}) \geq R_{-n\sigma FET}(V_{test} = V_{dd}) \geq \frac{R_{+n\sigma FET}(V_{strongOff})}{m} \tag{5.15}$$

Since both V_{high} and $V_{strongOff}$ will be greater than V_{th}, we can substitute the subthreshold Equation 5.9

$$I_2 \cdot 10^{\frac{(V_{th} + n\sigma_{V_{th}}) - V_{strongOff}}{S}} \leq I_2 \cdot 10^{\frac{(V_{th} - n\sigma_{V_{th}}) - V_{dd}}{S}} \leq m \cdot I_2 \cdot 10^{\frac{(V_{th} + n\sigma_{V_{th}}) - V_{strongOff}}{S}} \tag{5.16}$$

Which gives a value for $V_{strongOff}$

$$2 \cdot n\sigma_{V_{th}} + V_{dd} \leq V_{strongOff} \leq 2 \cdot n\sigma_{V_{th}} + V_{dd} + S\log(m) \tag{5.17}$$

Meaning that in order to guarantee that a transistor that is n sigma above the mean V_{th} in another wire will not dominate a transistor that is n sigma below nominal when $V_{test} \leq V_{dd}$, $V_{strongOff}$ needs to be at least $2n$ times the variation of V_{th} plus V_{dd} but does not need to be greater than that value plus $S\log(m)$ where S is the subthreshold decay per decade and m is the number of wires composing R_{rest}. In practice m is less than that since only a handful of wires per plane are expected to be very leaky and dominate R_{rest}.

Having $V_{strongOff}$, we can now calculate what V_{high} is for any wire. As defined, V_{high} is the value of V_{test} where $R_{FET}(V_{test} = V_{high}) = R_{rest}(V_{strongOff})$. Both V_{high} and $V_{strongOff}$ are greater than V_{th}, therefore, we can substitute the subthreshold current equation to get:

$$I_2 \cdot 10^{\frac{(V_{th} + n_{wire}\sigma_{V_{th}}) - V_{high}}{S}} = I_2 \cdot 10^{\frac{(V_{th} + n\sigma_{V_{th}}) - V_{strongOff}}{S}} \tag{5.18}$$

Solving for V_{high} yields:

$$V_{high} = V_{strongOff} + (n_{wire} - n) \cdot \sigma_{V_{th}} \tag{5.19}$$

Equations 5.13, 5.19 and 5.17 describe the values for V_{low}, V_{high} and $V_{strongOff}$ respectively. Together, they completely define the behavior of R_{nano} in terms of the V_{th} variation of the transistor, n_{wire}, and the most extreme variation expected at $\pm n\sigma_{V_{th}}$. They guarantee that we can isolate one transistor and define the size of the V_{test} range over which the transistor dominates R_{nano} as $V_{high} - V_{low}$ which is equal to $V_{strongOff} - V_{th_{nominal}} - n \cdot \sigma_{V_{th}}$. Understanding how R_{nano} operates helps to direct how the reference resistance R_{ref} should be set, as explained in the following section.

5.4 Lower Resistance: R_{ref}

For VMATCH to work, every transistor in the NanoPLA must be characterized. Although the NanoPLA provides the ability to measure multiple transistors in parallel, the discussion above exposed the fact that within a NanoPLA plane, wires are individually measured. Thus, we want to perform an individual measurement as quickly as possible. The time to measure a wire is determined by the time it takes $V_{measure}(V_{test})$ to settle and this is determined by the value of RC in Equation 5.3. RC is defined in Equation 5.4 which is re-written here in a way that will simplify the discussion.

$$RC = \frac{1}{\frac{1}{R_{nano}} + \frac{1}{R_{ref}}} \cdot C_{inWire} \tag{5.20}$$

Since the settling time of $V_{measure}(V_{test})$ is determined by RC, we want to consider setting R_{ref} so that RC is minimized. Equation 5.20 shows that RC is minimized when R_{ref} is significantly smaller than R_{nano}. In fact, if it is small enough, RC will approach $R_{ref} \cdot C_{inWire}$. Having a small R_{ref} with respect to R_{nano} has two advantages but one significant problem. The first advantage is clear, the settling time will be small and thus the measurement of the NanoPLA will be fast. Furthermore, the settling time is nearly independent of the value of R_{nano}, it will not change regardless of what wire we are measuring nor what V_{test} is set to. The disadvantage is seen in the values that $V_{measure}(V_{test})$ can take as a function of V_{dd}. If $V_{measure}(V_{test})$ is allowed to settle, Equation 5.3 behaves essentially like Equation 5.21.

$$V_{measure}(V_{test}, t_{measure}) = \frac{R_{ref}}{R_{nano} + R_{ref}} \cdot V_{dd} \tag{5.21}$$

In this case, the voltage measured is a fraction of the source voltage. This fraction is set by the value of R_{ref} and R_{nano}. Constraining R_{ref} to be small compared to R_{nano}, implies that the largest value $V_{measure}(V_{test})$ can take is below $\frac{1}{2}V_{dd}$, since it is a half when R_{ref} equals R_{nano}. The largest value for R_{ref} that is less than or equal to any expected R_{nano} is $R_{nano}(V_{low})$. Therefore, if $R_{ref} = R_{nano}(V_{low})$, $V_{measure}(V_{test})$ will only reach half of V_{dd} when $V_{test} \leq V_{low}$ at which point the transistor no longer dominates

R_{nano} and the measurement is pointless.

In the region where the transistor dominates, R_{nano} will range almost exactly between $R_{FET}(V_{low})$ and $R_{FET}(V_{high})$, a region exclusively in the subthreshold region of the transistor since V_{low} is bounded by V_{th}. As explained in Section 4.2, in this region, the resistance will be exponential in V_{test}. Thus as we linearly sweep V_{test} from V_{low} to V_{high}, the fraction of V_{dd} that will be measured at $V_{measure}$ will exponentially decrease. As previously stated, for the technology being explored here, $R_{FET}(V_{low})$ will be $2\times10^6\Omega$. Setting $V_{strongOff}$ as explained in Section 5.3.2, with $V_{dd} = 0.7V$, $V_{th} = 295mV$ and assuming 100 wires and $n = 3$, $R_{FET}(V_{high})$ will be $7 \times 10^{18}\Omega$. Using Equations 5.13 and 5.19, we can calculate that at the midpoint between V_{low} and V_{high}, $V_{mid} = n_{wire} \cdot \sigma_{V_{th}} + \frac{V_{strongOff}+V_{th_{nominal}}-n\sigma_{V_{th}}}{2}$. At this point $R_{nano}(V_{mid}) = 3.6 \times 10^{12}\Omega$, if $R_{ref} = 2 \times 10^4\Omega$, a value below the lowest expected R_{nano}, then the maximum value that $V_{measure}$ can achieve is approximately $5.5\times10^{-7}\%$ of V_{dd} which is $3.8\times10^{-9}V$. As such, half of all voltage measurements between V_{low} and V_{high} will be at or below $3.8 \times 10^{-9}V$. These are unreasonably small voltages which we don't expect to be able to accurately detect or measure. Therefore, as appealing as a low settling time is, setting R_{ref} low is not a feasible solution.

Let us consider the opposite case, setting R_{ref} higher than the highest expected R_{nano} (when $V_{test} = V_{high}$). $V_{measure}(V_{test})$ will be on the order of V_{dd} since $\frac{R_{ref}}{R_{nano}+R_{ref}}$ now approaches one and is never less than $\frac{1}{2}$ which makes detecting the voltage easier. However, now RC is approximately equal to $R_{nano}\cdot C_{inWire}$. As such, the settling time will be determined by the worst case RC which is when R is equal to the highest expected R_{nano}. Considering the parameters of the NanoPLA as stated above, and assuming $C_{inWire} = 3.6 \times 10^{-16}F$, the worst case expected settling delay is $3.6 \times 10^{-16}F \cdot 7 \times 10^{18}\Omega = 2520$s. Characterizing a whole NanoPLA when one measurement takes 2520s might appear to also makes this solution not feasible. However, careful consideration of what happens when a measurement is done before the worst case settling time, reveals that this is a reasonable solution since we do not need to measure at the highest expected resistance and can therefore measure a resource in a fraction of the time.

As long as we can detect a settled voltage somewhere above $V_{test} = V_{low}$, then we can calculate the V_{th} of the transistor. It is this fact that makes setting R_{ref} equal to or greater than the highest expected R_{nano} a good solution. First, consider what happens when $V_{measure}$ is sampled before it has had time to settle. As expressed in Equation 5.3, the voltage will reach $s\%$ of saturation after a time $t_{measure}$ equal to $-RC\cdot\ln(1-s)$. When $t_{measure} = RC$, the voltage will reach 63% of its settling value. The V_{test} that causes RC to equal $t_{measure}$ is $V_{settled}$. That is to say, for a chosen measure time $t_{measure}$, when V_{test} is below $V_{settled}$, the value of $V_{measure}$ will be at or above 63% of settled. Since the settled value will never be less than $\frac{1}{2}V_{dd}$, when measuring at $V_{settled}$, the lowest possible $V_{measure}$ will be $63\% \times \frac{1}{2}V_{dd}$. When V_{test} is above $V_{settled}$, $V_{measure}$ will not be settled and will be orders of magnitude less than V_{dd}. Figure 5.6 demonstrates the result of measuring the output voltage when V_{test} is above and below $V_{settled}$. Above, $V_{measure}$ is orders

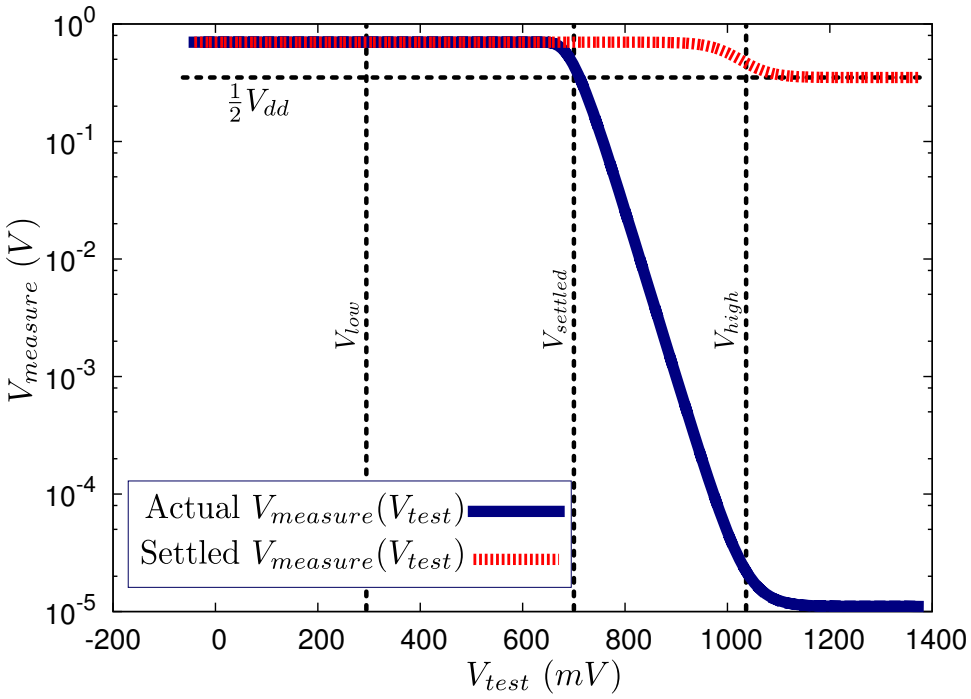


Figure 5.6: Effect on $V_{measure}$ when $t_{measure}$ is too small (Actual) and when $t_{measure}$ is "large enough" (Settled)

of magnitude lower than the voltage it will eventually settle to. While, as expected, measuring below $V_{settled}$ gives a value that is at least 63% of the fully settled voltage and approaches the fully settled voltage as V_{test} gets lower. For clarity of discussion, from now on, when talking of a settled value, we refer to a value that is at least 63% from the fully settled voltage.

The result above that calculated $t_{measure}$ to be 2520s incorrectly assumed that to measure V_{th} we need to be able to measure a settled $V_{measure}(V_{test})$ when V_{test} is anywhere in the range between V_{low} and V_{high}. It arrived at 2520s by estimating what the highest possible R_{FET} is when V_{test} is between V_{low} and V_{high} and concluded that it is when $V_{test} = V_{high}$. This assumption is flawed because as long as V_{test} is within the range of V_{low} to V_{high}, only one settled measurement of $V_{measure}(V_{test})$ is needed to calculate V_{th}. We present the intuition of why one measurement suffices here and leave the full development for the next section where the measurement algorithm is detailed. Figure 5.7 shows $V_{measure}(V_{test})$ for two resources, one with a nominal V_{th} and one where the V_{th} is 3σ above nominal. The time at which a measurement is taken, $t_{measure}$, is significantly less than the previously calculated 2520s but for every single measurement, it is the same. $V_{settled}$ for both resources is highlighted. The figure clearly illustrates the fact that as long as $t_{measure}$ remains constant, two resources with different V_{th}s have different $V_{settled}$. It is this fact that allows one measurement to suffice in order to calculate V_{th}. If we can find the V_{test} at which $V_{measure}(V_{test})$ is 63% of V_{dd} (*i.e.* find $V_{settled}$) we can determine what V_{th} is for that resource. $V_{settled}$ is defined by

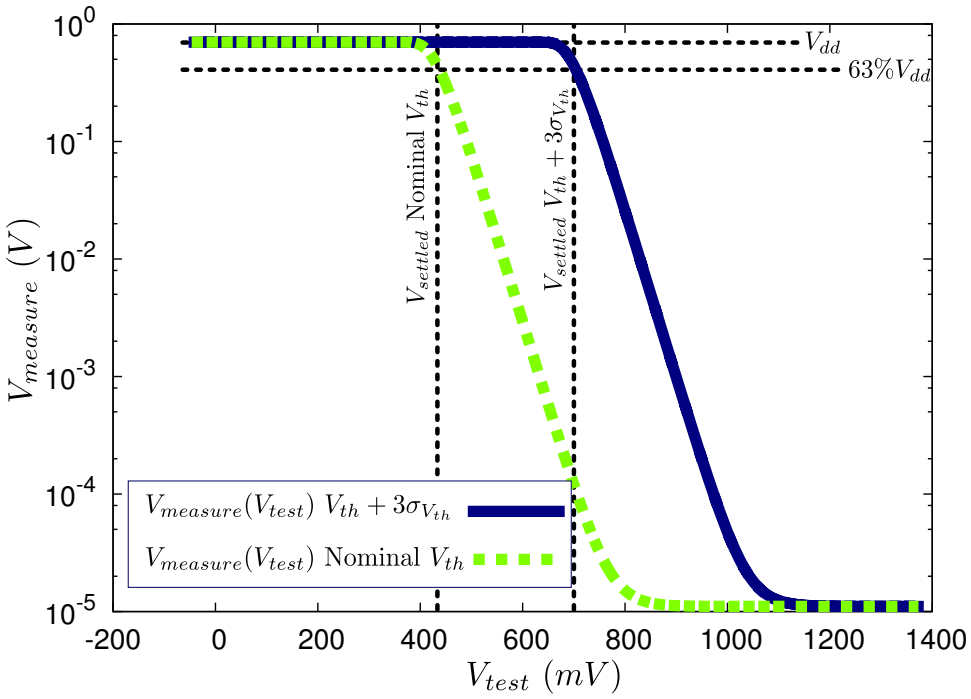


Figure 5.7: Actual $V_{measure}$ for two resources, one with a transistor at nominal V_{th} and one with a transistor 3σ above nominal V_{th}. Time $t_{measure}$ is constant over all measurements.

Equation 5.22. Section 5.5 exposes the full relationship between $t_{measure}$ and $V_{settled}$ but it is clear that as $t_{measure}$ decreases, $R(V_{settled})$ must decrease which occurs when $V_{settled}$ also decreases. Therefore, making $t_{measure}$ smaller will move $V_{settled}$ towards V_{low}. Since we only care about finding where $V_{settled}$ is when measuring a resource, we can safely decrease $t_{measure}$ and still be able to calculate V_{th} with the technique explained above.

$$R(V_{settled}) \cdot C = t_{measure} \tag{5.22}$$

$t_{measure}$ cannot be made arbitrarily fast, however. Lowering $t_{measure}$ drives $V_{setteled}$ towards V_{low}. If $t_{measure}$ is too fast, then $V_{settled}$ will be below V_{low} and therefore no longer useful to calculate V_{th}. Thus, the fastest $t_{measure}$ will be equal to the value RC takes when $V_{test} = V_{low}$. Since the R of RC is dominated by R_{nano} then $R = R_{nano}(V_{low}) = 2 \times 10^6\Omega$. Using the previously defined value of $C_{inWire} = 3.6 \times 10^{-16}F$ leads to a minimum $t_{measure} = RC = 7.2 \times 10^{-10}$s. We can round this to 1ns, which is a significantly more appealing measurement time than 2050s.

Potentially, we could measure a transistor in 1.0ns. However, before characterizing a transistor, we need to strongly turn off all other wires in the plane. The resistance of these strongly turned off wires will, by

definition, be greater then that of any being measured. If we demand that these wires settle before taking a measurement, the settling time will be greater than that of the worst case resistance of a transistor which was previously calculated to have a settling time of 2520s. However, this is not a problem because we can take advantage of the fact that if we do not allow them to settle, the apparent resistance of R_{rest} is significantly greater which has the positive effect of having the transistor being measured dominating R_{nano} even more. Therefore, we can conservatively bound the measurement time of one resource by 5ns which means that in one second we can serially measure on the order of 10^8 resources or if we are even more conservative, approximately 50,000 NanoPLA blocks per second.

As discussed, we no longer need to make sure that $V_{measure}$ is settled for the highest possible resistance, it only needs to be settled for any value of $V_{measure}$ when $V_{test} \leq V_{settled}$. Therefore, R_{ref} no longer needs to be larger than the largest possible R_{FET}. It simply needs to be equal to or slightly larger than $R_{FET}(V_{settled})$. Thus, as long as we are willing to measure at or above t_{min}, we can set R_{ref} equal to $R_{FET}(V_{settled})$. For the expected parameters of the NanoPLA, this sets $R_{ref} = 2.5M\Omega$.

5.5 Algorithm to Characterize the NanoPLA Resources

With the value of R_{ref} and $V_{strongOff}$ determined, and the minimum time to measure a wire calculated, we can focus on the measurement algorithm. A possible algorithm is as follows: After the initial setup of turning all resources in the plane strongly off, we can measure one resource by selecting it with the address decoder (Figure 5.2), charging the gate to V_{test} and measuring the voltage at the output of the voltage divider, $V_{measure}$. Assuming this measurement was taken after the voltage settled, we can use Equation 5.21 to calculate what R_{nano} is. Further assuming that V_{test} is in the region where R_{FET} dominates R_{nano}, with the value of V_{test} and the calculated R_{nano}, we can use the subthreshold Equation 5.9 to work out the threshold voltage for this transistor. Finally, once we know V_{th}, we can calculate R_{offFET}.

Although the steps suggested above will give the desired result of discovering R_{offFET}, this is not a practical algorithm for many reasons. First, it depends on being able to measure $V_{measure}$ very precisely. Even though we know that $V_{measure}$ will be on the order of V_{dd}, we still need a highly accurate measure to calculate V_{th}. The second problem is the assumption that V_{test} and $t_{measure}$ are such that $V_{measure}$ was settled when recorded. Since we don't know what V_{th} is, there is nothing to guide what V_{test} should be. For the same reason, we cannot assume that V_{test} will be in the region where the transistor's resistance dominates. Instead, consider the following algorithm which does a binary search over an interval of V_{test} to determine what $V_{setteld}$ is and from that, what V_{th} is.

The measurement algorithm works by doing a search over V_{test} to find the V_{test} at which $V_{measure} = 63\%V_{dd}$. Then it can directly calculate what V_{th} is based on this V_{test}. The reason why this works is the fact that when $t_{measure} = RC$ in Equation 5.3, $V_{measure}$ will be 63% of V_{dd}. If the measurement time $t_{measure}$ is

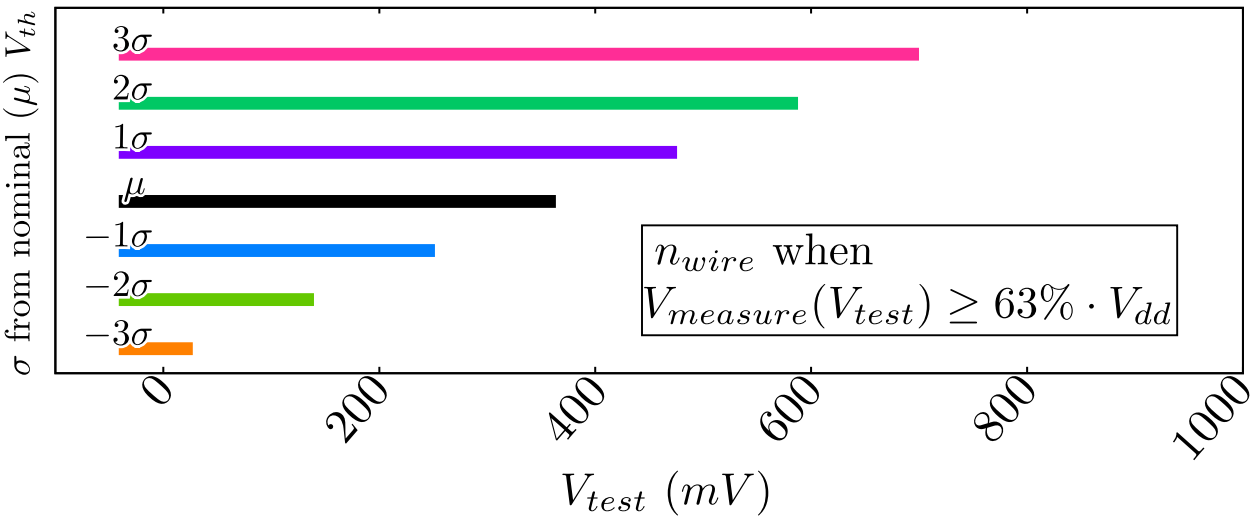


Figure 5.8: V_{test} Interval where $V_{measure} \geq 63\%\ V_{dd}$. 7 intervals, one for each resource from 3 sigma below nominal V_{th} to 3 sigma above nominal V_{th}.

held constant, then RC will be different for each wire and consequently the V_{test} that causes $t_{measure} = RC$ will be different depending on the variation of that transistor. As previously defined, the V_{test} that causes this is $V_{settled}$. Figure 5.8 shows the V_{test} interval over which $V_{measure} \geq 63\% V_{dd}$ for a constant $t_{measure}$. The highest V_{test} of the interval is $V_{settled}$, a different V_{test} for each wire. In fact, it is apparent from Figure 5.8 that we can directly map a value of $V_{settled}$ to the corresponding V_{th} as long as $t_{measure}$ does not change for each measurement. The following formally develop this correspondence.

$V_{settled}$ occurs when $RC = t_{measure}$, therefore from Equations 5.3 and 5.4, $R_{nano}(V_{settled})$ is:

$$R_{nano}(V_{settled}) = \frac{R_{ref} \cdot t_{measure}}{C_{inWire} R_{ref} - t_{measure}}$$

Substituting the subthreshold equation and solving for $V_{settled}$:

$$V_{settled} = V_{thNominal} + n_{wire}\sigma_{V_{th}} - S \log\left(\frac{C_{inWire} \cdot V_{dd}}{I_2 \cdot t_{measure}} - \frac{V_{dd}}{I_2 \cdot R_{ref}}\right) \tag{5.23}$$

Finally, solving for the variation of the wire, n_{wire}:

$$n_{wire} = \frac{V_{settled} - V_{thNominal} + S \log\left(\frac{C_{inWire} \cdot V_{dd}}{I_2 \cdot t_{measure}} - \frac{V_{dd}}{I_2 \cdot R_{ref}}\right)}{\sigma_{V_{th}}} \tag{5.24}$$

If we have $V_{settled}$, by using Equation 5.24, we can directly calculate, for the transistor being measured, what its V_{th} variation is relative to nominal V_{th}. The measurement no longer requires high precision on the value of $V_{measure}$ but a simple comparator between $V_{measure}$ and V_{dd}. As long as every measurement is compared

to 63% of V_{dd} after the same amount of time $t_{measure}$, this technique will give a unique answer for each variation level. However, this still requires that we find $V_{settled}$. Comparing $V_{measure}$ to V_{dd} at multiple V_{test} will help find $V_{settled}$. The precision to which $V_{settled}$ can be determined will thus be set by the degree of precision we have over V_{test} as well as the number of measurements we are willing to make to find one $V_{settled}$.

Comparing to 63% of V_{dd} is a simplifying convention that comes from Equation 5.3 when $t_{measure} = RC$. However, there is no reason why $V_{settled}$ has to be defined at 63%. If we define $V_{settled}$ to be the value of V_{test} that causes $t_{measure} = 10RC$, $V_{settled}$ will occur when $V_{measure}$ is at 99% of V_{dd}. In fact, as long as the value of $V_{measure}$ is high enough for a comparator to detect, we can compare to any percent of V_{dd} as long as every single measurement is compared to exactly the same percent. In Figure 5.7 we see that increasing or decreasing the percent of V_{dd} to which V_{setled} is defined will shift $V_{settled}$ equally for both resources. Thus, changing the percent of V_{dd} being compared will have a constant linear shift on the boundary where a resource is above that percent and where it is below, but it will have no effect on the relative separation between this boundary for two resources and consequently no ill effect on the measurement algorithm.

5.6 Measurement Precision

To significantly reduce the number of measurements, we use a binary search between the smallest and largest V_{test} expected. If the comparator says that $V_{measure}$ is too small, we try a lower V_{test}. On the other hand, when the comparator finds that $V_{measure}$ is greater than or equal to the established percent of V_{dd}, a higher V_{test} is used on the next measurement. The range over which V_{test} will have to be searched can easily be calculated. The lowest expected V_{test} will be the lowest expected V_{low} *i.e.* a V_{th} that is n sigma below the nominal V_{th}. The highest $V_{settled}$ will determine the maximum V_{test}. This will occur for a wire that is n sigma above the nominal V_{th} and is calculated as shown in Equation 5.23 when $n_{wire} = +n$. By knowing the maximum interval over which V_{test} should be explored, $Range(V_{test})$, and combining it with the degree of precision to which V_{test} can be controlled, $Precision(V_{test})$, we can calculate the number of measurements needed to get $V_{settled}$ to the highest precision achievable.

$$measurments = \left\lceil \log_2 \left(\frac{Range(V_{test})}{Precision(V_{test})} \right) \right\rceil \tag{5.25}$$

This number is minimized when $t_{measure}$ is the minimum time interval since the largest $V_{settled}$ will be minimized when $t_{measure}$ is minimized. For the parameters expected for the NanoPLA, when the maximum and minimum wires expected are within 3σ ($n = 3$) and $t_{measure}$ is minimum, the range for V_{test} will be $750mV$. If V_{test} can be controlled to within $1mV$, finding $V_{settled}$ to within $1mV$ will require 10

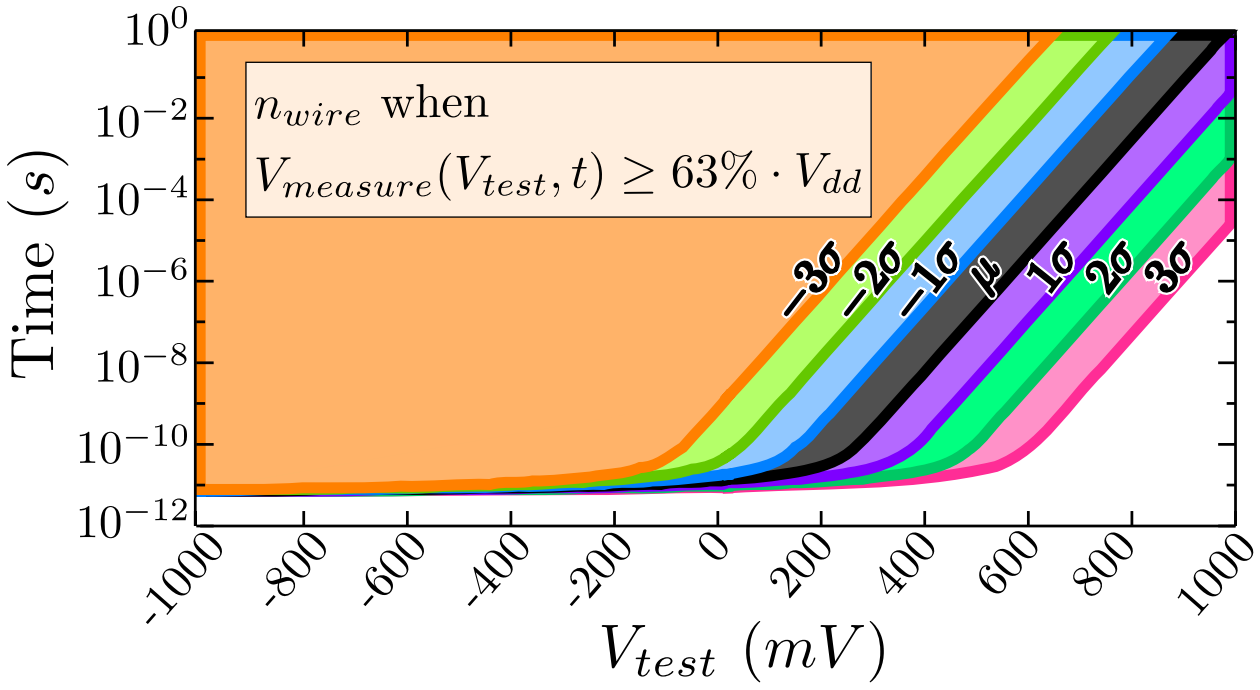


Figure 5.9: Region where $V_{measure} \geq 63\%\ V_{dd}$ as a function of V_{test} and time $t_{measure}$. 7 regions, one for each resource from 3 sigma below nominal V_{th} to 3 sigma above nominal V_{th}.

measurements, each taking 5ns (as we conservatively estimated before).

If the precision over V_{test} is not adequate, we can increase it by using precision in time. To explain this, consider what happens to n_{wire} (Equation 5.24) when we hold everything, including V_{test} constant but vary $t_{measure}$. n_{wire} will linearly decrease as $t_{measure}$ increases exponentially. Therefore, by comparing $V_{measure}$ to V_{dd} at multiple points in time, we can increase the precision with which we can measure V_{th}; however, this comes at a cost of slowing down the whole measurement processes. Figure 5.9 shows the space where for a given V_{test} and time $t_{measure}$, $V_{measure}$ will be at or above 63% of V_{dd} (63% at the strongly marked edge, above in the shaded region). We see that the region is smallest for a transistor with a V_{th} 3σ below nominal and largest for one at $+3\sigma$. If V_{test} is held constant at $400mV$, after 1ns, a transistor with a V_{th} variation of $+1\sigma_{V_{th}}$ will have reached a $V_{measure}$ greater than $63\%V_{dd}$ but one with $-1\sigma_{V_{th}}$ variation will only reach $63\%V_{dd}$ after approximately 100ns. Hence, we can calculate n_{wire} by either changing V_{test} or $t_{measure}$. If neither of these dimensions provides the required precision, we can increase the overall measurement precision by jointly changing V_{test} and $t_{measure}$. Figure 5.10 shows the precision with which n_{wire} can be measured as a function of V_{test} precision and $t_{measure}$ precision. Even though we can get high precision at the cost of time, in fact, high precision is not necessary for VMATCH to function. Section 6.4 shows that at 10% overhead, we can maintain 100% yield even when the measurement precision of V_{th} is 45mV.

Reduced precision has the effect of discretizing the V_{th}s into a handful of distinct values and the advantage

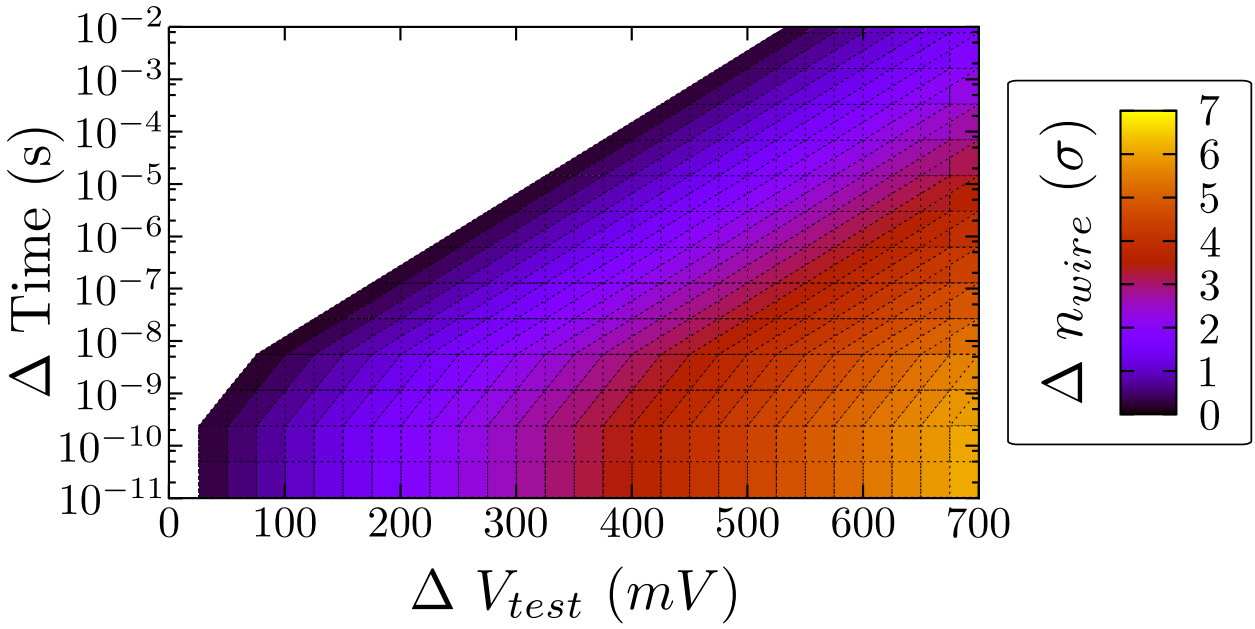


Figure 5.10: Precision of n_{wire} as a function of available V_{test} and Time precision.

that fewer measurements are needed per resource. There are two ways to reduce precision. First, we can directly reduce the precision by reducing the precision over V_{test}. Reducing the number of measurements done during the binary search also reduces the apparent precision. Both techniques however, have the same effect of linearly discretizing the values of V_{th}. Since we limit the range over which V_{test} needs to be search, any resource that "lands" above or below this region is qualitatively marked as being extremely leaky or extremely slow. This discretization of V_{th} has the effect of exponentially discretizing R_{offFET} since R_{offFET} is exponential in V_{th} (Equation 5.9).

In conclusion, by correctly configuring the NanoPLA, in a reasonable amount of time, we can completely characterize the variation of every transistor in the system. Furthermore, we can carefully adjust the precision of these measurements by adjusting the precision of the test voltage and measurement time.

Chapter 6

Results

6.1 Experimental Setup

We simulated a NanoPLA using 5 nm pitch wires with crosspoints implemented in Amorphous-Si technology [25] 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 $10k\Omega$.

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 [11] 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. Finally, we simulated the effects the measurement technique presented in Chapter 5 has on VMATCH as we decreased the measurement precision. Variation on all components was modeled as independent Gaussian distributions as previously defined in Section 2.3.

6.2 Achievable Yield

Figure 6.1 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 Equation 3.1. 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

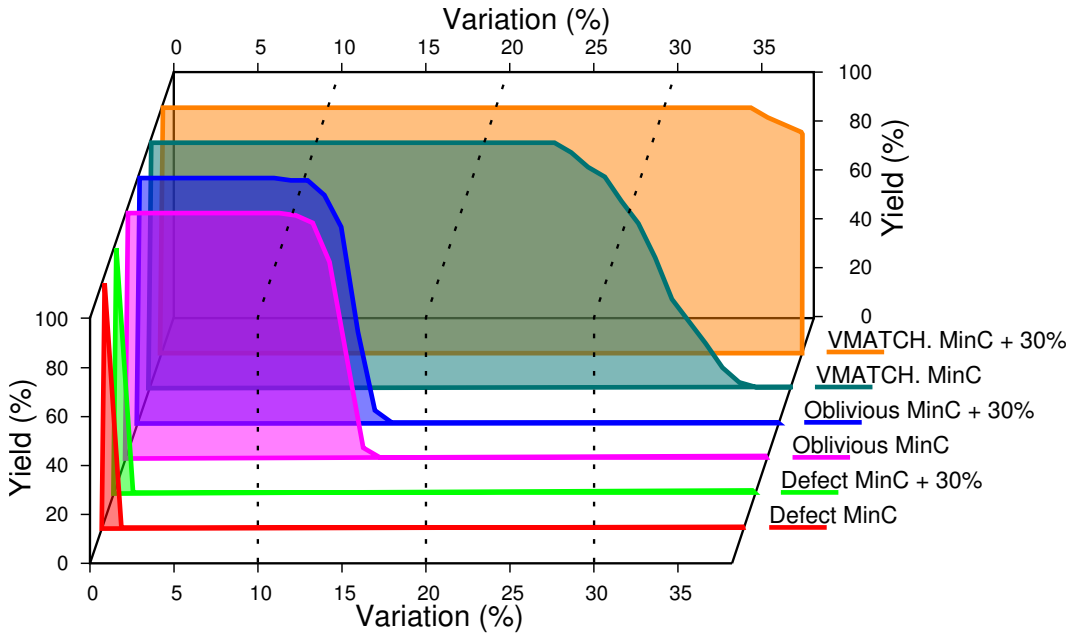


Figure 6.1: Yield curves for Oblivious, Defect and VMATCH mapping. Benchmark spla 100 chips

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 width 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.

Table 6.1 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%.

6.3 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

Net	Highest Variation That Achieves 100% Yield at 30% Extra Channels	
	Oblivious	VMATCH
alu4	10%	32%
apex2	11%	25%
apex4	11%	32%
bigkey	12%	33%
clma	10%	29%
des	9%	30%
diffeq	12%	29%
dsip	12%	32%
elliptic	10%	29%
ex1010	11%	26%
ex5p	10%	35%
frisc	10%	26%
misex3	10%	28%
pdc	9%	32%
s298	9%	28%
s38417	8%	29%
s38584.1	11%	30%
seq	9%	34%
spla	8%	35%
tseng	12%	29%
Mean	10%	30%

Table 6.1: Tolerable Variation for Toronto 20 Benchmark Set With 30% Wider Channel Over the Minimum Channel Width. Unlimited Measurement Precision.

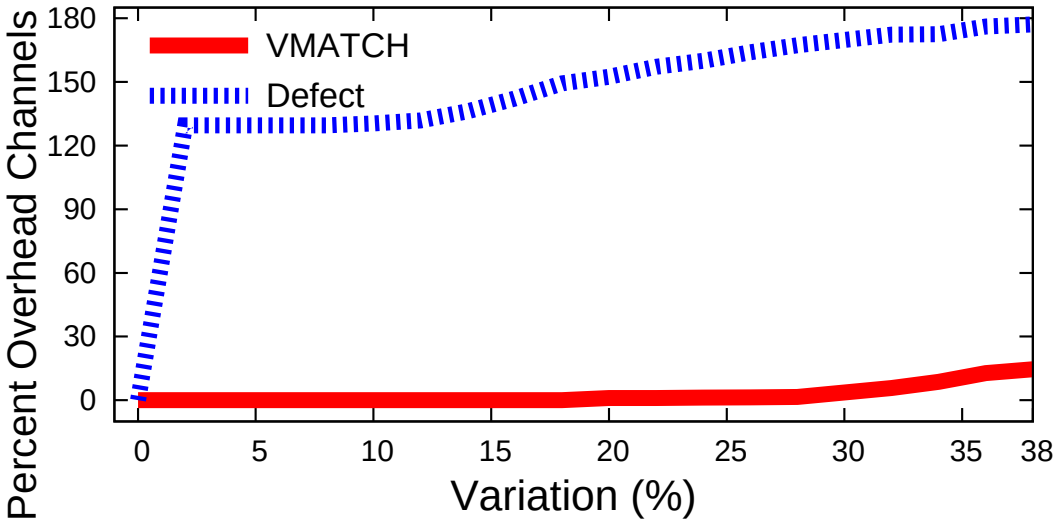


Figure 6.2: Average minimum number of channels required to get 100% yield. Benchmark spla 100 chips

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, Figure 6.2 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.

Figure 6.3 shows how these channel widths (Figure 6.2) 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 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.

Tables 6.2 and 6.3 show 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.

	Defect at 38% σ and 100% Yield							
	% Extra Channels		Ratio to Nominal					
			Delay		Energy		Area	
Net	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**
alu4	174	19	4.3	0.73	2.4	0.16	3.3	0.35
apex2	192	14	6.5	0.75	2.6	0.13	4.0	0.31
apex4	192	25	4.5	0.62	2.4	0.19	3.6	0.47
bigkey	164	14	8.5	1.2	2.5	0.15	4.2	0.44
clma	183	15	7.5	0.80	2.7	0.15	4.9	0.46
des	178	14	7.7	19	2.6	0.14	4.5	0.41
diffeq	241	24	7.5	1.3	2.6	0.15	3.5	0.31
dsip	202	17	9.9	1.3	2.8	0.15	4.3	0.43
elliptic	162	12	6.7	0.79	2.6	0.14	4.6	0.42
ex1010	260	24	6.4	0.80	2.8	0.17	4.2	0.41
ex5p	167	39	4.0	0.87	2.2	0.30	3.2	0.70
frisc	214	12	7.2	0.77	2.8	0.11	4.6	0.31
misex3	204	27	4.4	0.61	2.6	0.21	4.0	0.56
pdc	191	12	5.9	0.43	2.7	0.11	4.2	0.31
s298	165	12	9.8	1.1	2.6	0.13	4.4	0.39
s38417	184	22	7.3	1.3	2.8	0.24	4.9	0.77
s38584.1	228	13	6.6	0.59	2.7	0.11	4.2	0.24
seq	168	13	5.0	0.74	2.5	0.13	4.1	0.38
spla	177	21	5.1	0.59	2.6	0.19	3.8	0.45
tseng	260	29	6.8	0.95	2.6	0.18	3.5	0.37
Mean	193	17	6.4	0.96	2.6	0.18	4.1	0.41

Table 6.2: Overheads Required to Maintain 100% Yield When Mapping the Toronto 20 Benchmark Set at $\sigma = 38\%$ Using the Defect-Avoiding Algorithm. Unlimited Measurement Precision.

	VMATCH at 38% σ and 100% Yield							
	% Extra Channels		Ratio to Nominal					
			Delay		Energy		Area	
Net	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**	**Mean**	**St.Dev.**
alu4	17	12	1.5	0.20	1.2	0.10	1.2	0.12
apex2	26	21	1.9	0.50	1.3	0.17	1.3	0.29
apex4	17	13	1.6	0.35	1.2	0.09	1.3	0.13
bigkey	17	13	2.0	0.47	1.2	0.12	1.3	0.23
clma	34	5.9	2.1	0.11	1.3	0.04	1.5	0.10
des	26	18	1.9	0.51	1.2	0.15	1.4	0.30
diffeq	23	21	2.0	0.41	1.2	0.13	1.3	0.18
dsip	28	16	2.3	0.56	1.3	0.13	1.3	0.23
elliptic	27	26	1.9	0.79	1.3	0.24	1.5	0.51
ex1010	43	22	2.0	0.39	1.4	0.15	1.4	0.24
ex5p	6.7	12	1.4	0.14	1.1	0.07	1.2	0.14
frisc	42	17	2.0	0.39	1.4	0.13	1.5	0.24
misex3	26	11	1.4	0.21	1.2	0.09	1.4	0.10
pdc	30	16	1.9	0.3	1.3	0.13	1.4	0.21
s298	23	32	2.4	1.4	1.3	0.29	1.5	0.62
s38417	26	5.1	1.9	0.19	1.3	0.04	1.4	0.08
s38584.1	33	6.4	1.9	0.31	1.3	0.05	1.4	0.07
seq	24	14	1.6	0.26	1.2	0.12	1.3	0.21
spla	15	9.0	1.7	0.16	1.2	0.08	1.2	0.10
tseng	24	11	1.9	0.61	1.2	0.06	1.3	0.08
Mean	24	17	1.9	0.34	1.3	0.10	1.4	0.20

Table 6.3: Overheads Required to Maintain 100% Yield When Mapping the Toronto 20 Benchmark Set at $\sigma = 38\%$ Using VMATCH. Unlimited Measurement Precision.

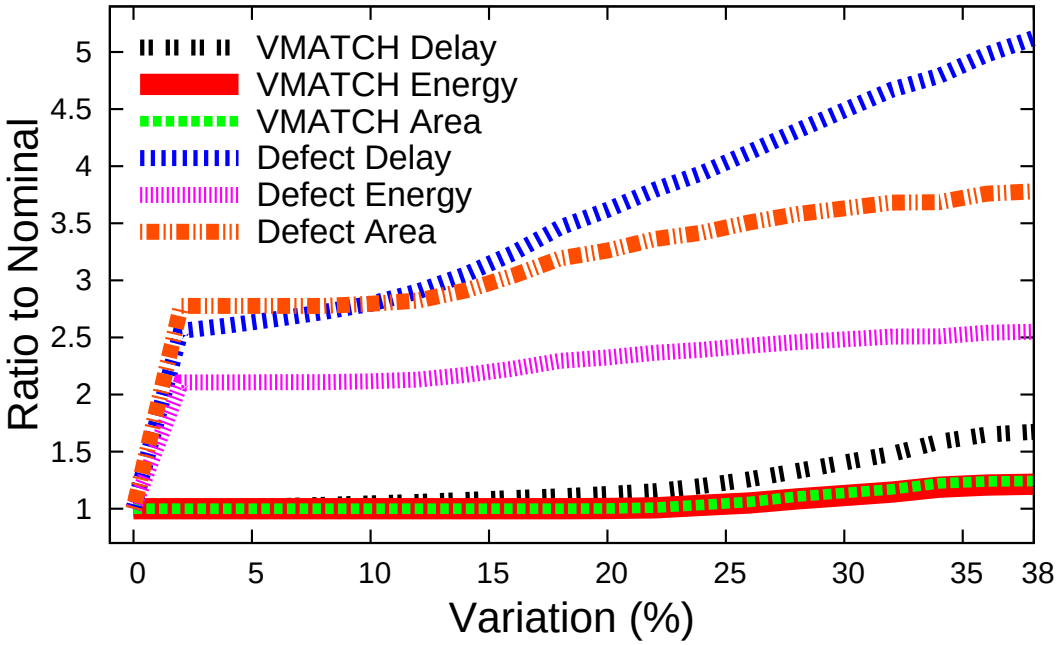


Figure 6.3: Ratio to variation free Nominal of delay, energy and area at 100% yield. Benchmark spla 100 chips

6.4 Measurement Precision

VMATCH needs to know the value of R_{offFET} for each transistor in order to produce a good mapping. Chapter 5 shows how to measure this information and explains how the precision of the measurement affects the R_{offFET} distribution. We simulated this measurement technique and assigned a value of positive (negative) infinity to the R_{offFET} of any resource that was above (below) the measurement boundaries as defined in Section 5.6.

Figure 6.4 shows the number of extra channels needed to maintain 100% yield as precision decreases. Precision is marked on the lower axis in mV and refers to the smallest precision at which the voltage can be measured. The upper axis shows the number of unique values of V_{th} the precision produces. From this graph, it is clear that as long as we add enough resources, we can maintain 100% yield at very low precision.

Adding extra resources has a cost, the total delay, energy and area increases. Figure 6.5 shows how the extra channels needed to maintain 100% yield alter these values as compared to the infinite precision case. It is interesting to note that even when the precision is limited to $40mV$, there is nearly no negative effect on the total delay, energy and area. This is equivalent to only having 10 discrete values for V_{th} as the upper axis shows. Table 6.4 shows the lowest precision required to maintain the parameters to within 10% of infinite precision for the Toronto 20 Benchmarks. On average at 10% overhead we only need $45mV$ precision.

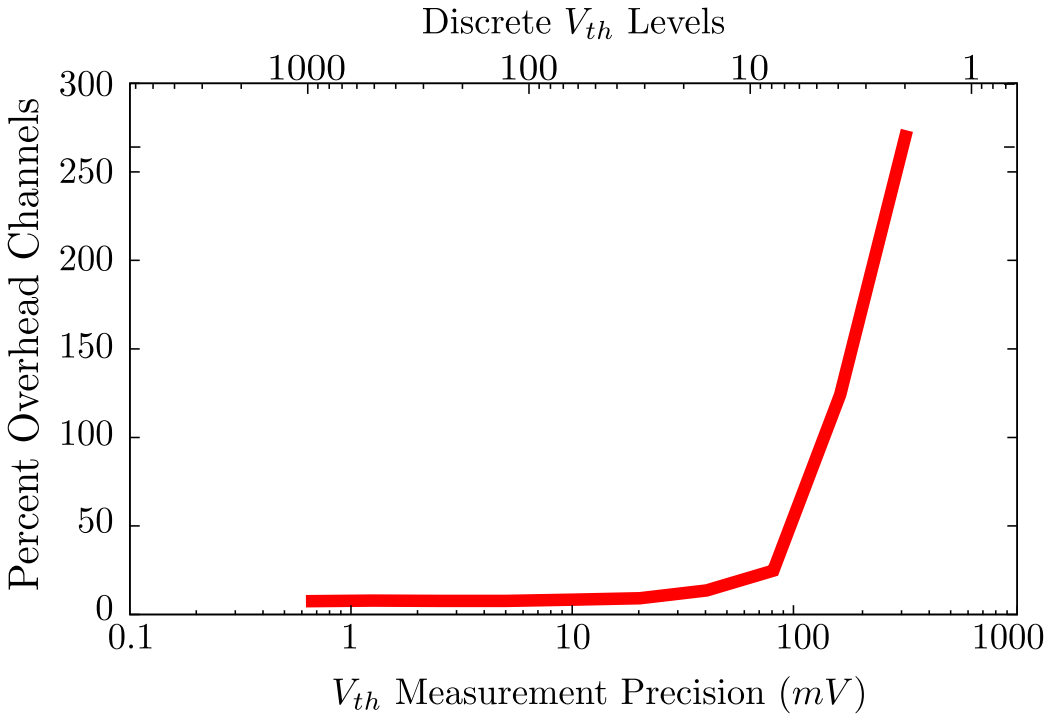


Figure 6.4: Percent Extra Channels Needed for VMATCH to Maintain 100% Yield as Precision (Number of Discrete V_{th} Levels) Decreases. Benchmark spla at $\sigma = 38\%$ 100 Chips

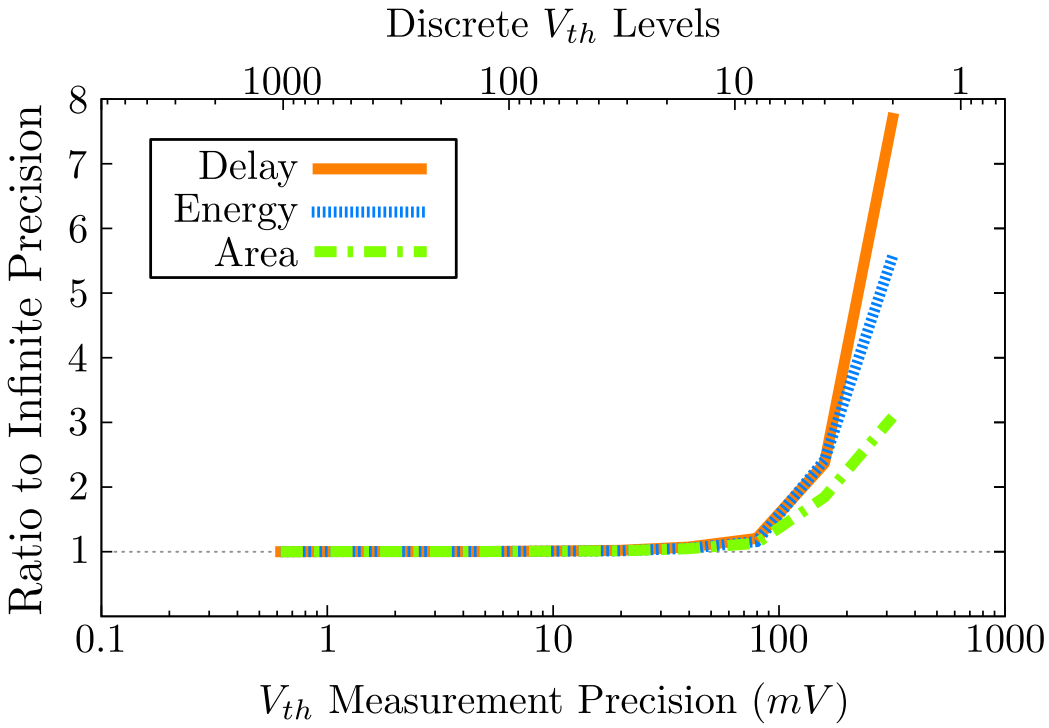


Figure 6.5: Effect of Precision (Number of Discrete V_{th} Levels) on Delay, Energy and Area as a Ratio to Infinite Precision. VMATCH, Benchmark spla at $\sigma = 38\%$ 100 Chips

Net	Minimum Precision (mV) Delay	Energy	Area
alu4	40	80	40
apex2	20	80	40
apex4	40	40	40
bigkey	20	160	40
clma	40	40	40
des	40	80	80
diffeq	20	160	40
dsip	80	160	160
elliptic	40	80	80
ex1010	40	40	40
ex5p	80	80	80
frisc	80	80	80
misex3	40	80	40
pdc	40	40	40
s298	20	80	40
s38417	40	80	40
s38584.1	40	80	20
seq	40	80	40
spla	40	40	40
tseng	20	160	160
Mean	34	70	46

Table 6.4: Minimum Precision Required to Maintain Parameters within 10% of Unlimited Precision Values at 100% Yield When Mapping the Toronto 20 Benchmark Set at $\sigma = 38\%$

Chapter 7

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. Furthermore, we explain how to characterize the transistor in the NanoPLA and observe that limited measurement precision does not affect the quality of the results achieved by VMATCH. This shows that "nanoscale 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.

Chapter 8

Future Work

Although our results are encouraging, there are still many questions left unanswered and techniques unexplored. One important area to understand further is the effects different technology assumptions have on VMATCH. Also, we believe that we can improve performance and reduce power if we can engineer the fanout of the net being mapped to match the variation in the NanoPLA. Finally, despite the fact that we have shown how to characterize the full NanoPLA, a technique that works well with only partial knowledge of the variation would go a long way to extending the lifetime of the NanoPLA.

VMATCH depends on R_{offFET} being the only dominating source of variation. Our technology assumptions follow the current literature; nevertheless, since the field of sub-lithographic circuits is still young, it is important to understand if and how VMATCH can cope with technologies that might arise with different variation properties than the ones we are currently exploring. It is possible that some technology might exist that prevents VMATCH from achieving good results; still, we believe that using one type of variation to reduce another will be applicable under many circumstances.

VMATCH performs well considering the extreme variation it has to deal with. However, if we could better shape the logical variation to match the physical variation, our results would improve even further. There are many ways this can be done, in particular functions within a NanoPLA plane can be duplicated or moved to another plane, thus readjusting the fanout of the function. Also, it is not clear how the size of the NanoPLA blocks, *i.e.* number of inputs, AND-terms and outputs, can be adjusted to better fit the expected variation. [10] characterized this for the variation free case, but the question still remains as to how variation changes their results.

Finally, we are also interested in techniques that avoid the need for full knowledge of the physical variation but rather allows for a process of variation discovery only as necessary. Being able to incrementally fix a mapping would be beneficial for both the initial mapping and for its lifetime operation.

Bibliography

[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.

[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] B. Gojman, H. Manem, G. S. Rose, and A. DeHon, "Inversion Schemes for Sublithographic Programmable Logic Arrays," *IET CDT*, vol. 3, no. 6, pp. 625–642, November 2009.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] Y. Wu, J. Xiang, C. Yang, W. Lu, and C. M. Lieber, "Single-crystal metallic nanowires and metal/semiconductor nanowire heterostructures," *Nature*, vol. 430, pp. 61–64, July 1 2004.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] ——, "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.

[28] 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.

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.

[35] 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.

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

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

[38] 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.

[39] 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.

[40] 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.

[41] B. Gojman and A. DeHon, "VMATCH: Using Logical Variation to Counteract Physical Variation in Bottom-Up, Nanoscale Systems," in *ICFPT*. IEEE, December 2009, pp. 78–87.

[42] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," *ACM Computing Surveys*, vol. 18, no. 1, pp. 23–38, 1986.