Fault-Tolerant Nanoscale Microprocessor Design On Semiconductor Nanowire Grids

Teng Wang

University of Massachusetts - Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_1

Part of the Electrical and Computer Engineering Commons

Recommended Citation

https://scholarworks.umass.edu/dissertations_1/67
FAULT-TOLERANT NANOSCALE MICROPROCESSOR DESIGN ON SEMICONDUCTOR NANOWIRE GRIDS

A Dissertation Presented

by

TENG WANG

Submitted to the Graduate School of the University of Massachusetts Amherst in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2009

Electrical and Computer Engineering
FAULT-TOLERANT NANOSCALE MICROPROCESSOR DESIGN ON SEMICONDUCTOR NANOWIRE GRIDS

A Dissertation Presented

by

TENG WANG

Approved as to style and content by:

________________________________________
Csaba Andras Moritz, Chair

________________________________________
Israel Koren, Member

________________________________________
Wayne Burleson, Member

________________________________________
Charles C. Weems, Member

Christopher V. Hollot, Department Head
Electrical and Computer Engineering
DEDICATION

To my parents Yueying and Fugui, for bring me to this beautiful world

To my friends, for making my life vivid

To my wife Jessie, for your endless love and patience

To my daughter Lillian, for the joy you bring to us everyday
ACKNOWLEDGMENTS

I would like to gratefully and sincerely thank Dr. Csaba Andras Moritz for his guidance, understanding and patience throughout my graduate study at Umass Amherst. His mentorship was paramount in providing a well rounded experience consistent to my long-term career goals. He encouraged me to not only grow as a programmer but also as an independent thinker. For everything you’ve done for me, Dr. Moritz, I thank you.

I would also like to thank the former and present members of the NASIC research group, especially Pritish Narayanan, Michael Leuchtenburg, Dr. Yao Guo and Dr. Mahmoud Ben Naser, for their consistent help during my PhD study. The friendship with them will never fade with time.

I would like to thank the Electrical and Computer Engineering Department of Umass Amherst for the support, which turns my dream into reality. In addition, I would also thank all members in my PhD committee, Drs. Csaba Andras Moritz, Israel Koren, Wayne Burleson and Charles C. Wheems, for their significant inputs from dissertation proposal to the doctoral defense.

My wife Jessie Luo, you deserve my special grate for your continuous support, encouragement, patience and unwavering love. I also want to thank my parents, Fugui Wang and Yueying Tong, for their selfless and endless love to me. My little angel, Lillian Wang, although your birth did give me a tough time, I want to give my final appreciation to you. You deliver happiness to me more than anyone else.

Thank you, to all of you!
As CMOS manufacturing technology approaches fundamental limits, researchers are looking for revolutionary technologies beyond the end of the CMOS roadmap. Recent progress on devices, nano-manufacturing, and assembling of nanoscale structures is driving researchers to explore possible new fabrics, circuits and architectures based on nanoscale devices.

Several fabric architectures based on various nanoscale devices have been proposed for nanoscale computation. These show great advantages over conventional CMOS technology but focus on FPGA-style applications. There has been no work shown for nanoscale architectures tuned for a processor application.

This dissertation proposes a novel nanowire-based 2-D fabric referred to as Nanoscale Application-Specific IC (NASIC). Compared with other nanoscale fabric architectures, NASIC designs can be optimized for higher density and performance in an application-specific way (similar to ASIC in this aspect) and used as a fabric for
processors. We present the design of a wire-streaming processor (WISP-0), which exercises many NASIC circuit styles and optimizations.

A major goal of NASIC, and for other nanoscale architectures, is to preserve the density advantage of underlying nanodevices. Topological, doping and interconnect constraints can severely impact the effective density that can be achieved at the system level. To handle these constraints, we propose a comprehensive set of optimizations at both circuit and logic levels. Evaluations show that with combined optimizations, WISP-0 is still 39X denser than the equivalent design in 18nm CMOS technology (expected in 2018 by ITRS).

Another key challenge for nanoscale computing systems is dealing with the unreliable nanodevices. The defect rate of nanodevices is expected to be orders of magnitude higher than what we are accustomed to with conventional CMOS processing based on lithography. In this dissertation, we first investigate various sources of defects/faults in NASIC circuits and analyze their impacts. Then, a hierarchical, multi-layer solution is proposed to tolerate defects/faults. Simulation shows that the yield of WISP-0 is as high as 50% even if as many as 15% transistors are defective.

Estimations of the speed, power consumption of NASIC designs are also presented.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENTS</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xii</td>
</tr>
</tbody>
</table>

## CHAPTER

### 1 INTRODUCTION

1.1 Scaling Limitations of CMOS .................................1
1.2 Nanoscale Computing ................................................3
1.3 Emerging Technologies .............................................4
1.4 Research Goals .......................................................7
1.5 Contributions of Dissertation .................................9
1.6 Organization of Dissertation ..................................12
1.7 List of Publications ..............................................13

### 2 CIRCUIT DESIGN AND MANUFACTURING ...............15

2.1 Introduction ..............................................................15
2.2 Static NASIC Designs .................................................16
   2.2.1 Static Logic Gates on Semiconductor Nanowires ..........16
   2.2.2 Designs of Static NASIC Tiles .................................17
2.3 Proposed Manufacturing Process ..............................19
2.4 Dynamic NASIC Circuits ............................................22
   2.4.1 Challenges in Designing Sequential Circuits in NASIC Fabric ..23
   2.4.2 Proposed Dynamic Circuit Designs, Nano-latches ..........24
2.5 NASIC Designs with Single-type of FETs ....................28
2.6 Combining Logic Families in 2-D NASIC Fabric ............29
   2.6.1 Manufacturing Implications ..................................32
   2.6.2 Fault Tolerance Implications ................................32
   2.6.3 Applicability to Other Types of 2-Level Fabrics ..........33
2.7 Circuit Analysis .......................................................33
   2.7.1 Simulation Setup ..............................................34
   2.7.2 Signal Integrity Considerations .............................36
   2.7.3 Charge Sharing ................................................38
   2.7.4 Capacitive Coupling ..........................................40
   2.7.5 Sensitivity to Parameter Variations .......................40
2.8 Chapter Summary ........................................................................................................42

3 NASIC ARCHITECTURE AND WIRE-STREAMING PROCESSOR ........................................44

3.1 Introduction ..................................................................................................................44
3.2 Pipeline Structure and Multi-tile Design ..................................................................44
3.3 Wire-Streaming Processor .......................................................................................46
3.4 Comparison with Other Nanoscale Computing Architectures ................................52
3.5 Chapter Summary .......................................................................................................54

4 FAULT TOLERANCE ..........................................................................................................55

4.1 Introduction ..................................................................................................................55
4.2 Fault Model in NASICs ..............................................................................................56
  4.2.1 Types and Sources of Errors ..............................................................................56
  4.2.2 Fault Model Assumed .......................................................................................57
4.3 Reconfiguration vs. Built-in Redundancy ................................................................56
4.4 Built-in Fault-Tolerance in NASICs ........................................................................61
  4.4.1 Circuit-Level and Structural Redundancy .......................................................61
  4.4.2 Improving Fault-Tolerance by Interleaving NWs ............................................63
  4.4.3 Adding Weak Pull-up/down NWs ..................................................................65
  4.4.4 Code-based Fault-masking Circuits at Nanoscale ..........................................66
    4.4.4.1 Hamming Distance ..................................................................................66
    4.4.4.2 Error-Correcting Code Background .....................................................67
    4.4.4.3 Code-based Fault-masking in NASICs ...................................................68
    4.4.4.4 One-bit NASIC Adder with Code-based Fault-masking .........................71
    4.4.4.5 Code-based Fault-masking Combined with 2-way Redundancy ..........72
  4.4.5 Adding CMOS Voting .........................................................................................72
4.4.6 Voting at Nanoscale .............................................................................................75
  4.4.6.1 Nanoscale Voting with Unreliable Voting Circuits ..................................75
  4.4.6.2 Nanoscale TMR in AND-OR NASIC Fabric ............................................76
  4.4.6.3 Nanoscale 6MR in AND-OR/NOR NASIC Fabric ....................................77
4.5 Chapter Summary .......................................................................................................79

5 EVALUATIONS OF DENSITY, SPEED, POWER AND YIELD ...................................81

5.1 Introduction ..................................................................................................................81
5.2 Density .........................................................................................................................82
  5.2.1 Comparison with Equivalent CMOS Processor ..............................................82
  5.2.2 Impact of NW Pitch on Density ......................................................................85
5.3 Speed ..........................................................................................................................86
  5.3.1 Parameter Assumptions ....................................................................................86
  5.3.2 Delay Calculations .........................................................................................87
5.4 Power Consumption ....................................................................................................89
  5.4.1 Dynamic Power ...............................................................................................89
5.4.2 Leakage Power ..............................................................................................92
5.5 Yield ..............................................................................................................94
  5.5.1 Yield Evaluation of WISP-0 .................................................................94
  5.5.2 WISP-0 Yield-Density Product Evaluation .......................................97
  5.5.3 NASICs with Clustered Defects .........................................................99
  5.5.4 WISP-0 Yield with Clustered Defects ...............................................99
  5.5.5 Impact of Transient Errors ...............................................................102
5.6 Chapter Summary .......................................................................................103

6 CONCLUSION ...................................................................................................104

BIBLIOGRAPHY .................................................................................................107
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1. Parameter Values for Devices and Circuits</td>
<td>34</td>
</tr>
<tr>
<td>Table 3.1. Comparison of Nanoscale Computing Architectures</td>
<td>53</td>
</tr>
<tr>
<td>Table 4.1. Fault Model for NASIC Fabric</td>
<td>59</td>
</tr>
<tr>
<td>Table 5.1. Parameter Values for Density Estimation</td>
<td>82</td>
</tr>
<tr>
<td>Table 5.2. Parameter Values Used for Speed Estimation</td>
<td>86</td>
</tr>
<tr>
<td>Table 5.3. Capacitive Loading (in aF)</td>
<td>87</td>
</tr>
<tr>
<td>Table 5.4. Delay (ps) – WISP-0 with AND-OR Logic</td>
<td>88</td>
</tr>
<tr>
<td>Table 5.5. Delay (ps) – WISP-0 with NAND-NAND Logic</td>
<td>89</td>
</tr>
<tr>
<td>Table 5.6. Dynamic Power Consumption (μW)</td>
<td>91</td>
</tr>
<tr>
<td>Table 5.7. Leakage Power Consumption (μW)</td>
<td>93</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1: The growing device/power density of Intel’s microprocessors</td>
<td>2</td>
</tr>
<tr>
<td>Figure 2.1: Static AND logic on an n-type nanowire</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.2: Static AND and OR logics on p-type and n-type nanowires</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2.3: A static NASIC tile implementing a 1-bit full adder. The thinner wires are nanowires while the thicker wires are microwires</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2.4: Possible manufacturing steps for NASICs: a) Grow NWs and dope them. b) Align NWs to form a regular array. c) Coarse-grain metallization using a lithographic mask. d) Additional metallization using perpendicular NWs as a fine-grain mask. FETs are formed at the crosspoints indicated by the dashed circles</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2.5: 1-bit NAND flip-flop design on 2-D NASIC fabric. The layout on NASIC fabric is shown on the left. The right side shows the corresponding schematic</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.6: Dynamic circuits implementing AND, NAND, OR, and NOR logic</td>
<td>25</td>
</tr>
<tr>
<td>Figure 2.7: A 1-bit dynamic adder using the AND-OR cascaded logic</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2.8: Waveform for cascaded dynamic AND-OR circuit. The hold phase is added for cascading purpose. The approach might be extended by making the phases asymmetrical to achieve better overall latency</td>
<td>27</td>
</tr>
<tr>
<td>Figure 2.9: n-FET only version of a 1-bit adder using the NAND-NAND cascaded logic</td>
<td>29</td>
</tr>
<tr>
<td>Figure 2.10: a 1-bit adder using H2L logic</td>
<td>31</td>
</tr>
<tr>
<td>Figure 2.11: (a) The layout of a 4-stage cascaded NAND circuit implemented on 2 NASIC nanotiles. (b) Corresponding circuit schematic of the 4-stage cascaded NAND showing all signals</td>
<td>34</td>
</tr>
<tr>
<td>Figure 2.12: Simulation results for 4-stage cascaded dynamic NAND</td>
<td>36</td>
</tr>
</tbody>
</table>
Figure 2.13: Left: Charge sharing in dynamic circuits. Right: Load transistor for alleviating charge sharing effect. ...................................................... 38

Figure 2.14: Noise Margins of a dynamic NAND-NAND circuit achieved with different threshold voltages. ........................................................... 41

Figure 3.1: A pipelined 2-bit carry-ripple adder...................................................... 45

Figure 3.2: The floorplan of WISP-0................................................................. 46

Figure 3.3: The layout (left side) and schematic (right side) of the program counter in WISP-0................................................................. 48

Figure 3.4: The layout (left side) of the instruction ROM in WISP-0. On the right side are the codes stored in the ROM................................. 49

Figure 3.5: The layout of the instruction decoder in WISP-0.............................. 50

Figure 3.6: The layout (left side) and schematic (right side) of the register file in WISP-0................................................................. 50

Figure 3.7: The layout (left side) and schematic (right side) of the ALU in WISP-0................................................................. 51

Figure 4.1: A simple model for clustered defects; shows how defect probabilities are decreasing for FETs and NWs further away from a cluster center. With manufacturing approaches crystallizing, a more accurate defect cluster model would need to be developed........ 59

Figure 4.2: A simple NASIC circuit with built-in redundancy.......................... 62

Figure 4.3: Interleaving NWs and adding weak pull-up/down NWs to reduce hard-to-mask regions. The bottom circuit has interleaved vertical NWs and weak pull-down NW between the AND and OR planes....... 64

Figure 4.4: A simple NASIC circuit: (a) Original design without fault-tolerance. (b) Design with the built-in code-base fault-masking technique................................................................. 69

Figure 4.5: 1-bit NASIC full adder with code-based fault-masking. The circuits in the shadowed area are redundant circuits added for the purpose of error correction......................................................... 71

Figure 4.6: Triple modular redundancy configuration in a pipelined system....... 73

Figure 4.7: The reliabilities of output (R) after the TMR/6MR voting circuits....... 76
Figure 4.8: Nanoscale TMR design in pure AND-OR fabric ......................... 77
Figure 4.9: Nanoscale 6MR design with H2L voters .............................. 77
Figure 4.10: A 1-bit 6MR H2L voter .................................................. 78
Figure 5.1: Normalized WISP-0 density with different defect fault tolerance techniques assuming a 10-nm NW pitch and an equivalent CMOS WISP-0 at various technology nodes ............................................. 83
Figure 5.2: Normalized WISP-0 density with different defect fault tolerance techniques assuming a 20-nm NW pitch at various technology nodes ........................................................................ 86
Figure 5.3: The yield achieved for WISP-0 with different techniques when only considering stuck-on transistors ................................................................. 96
Figure 5.4: The yield achieved for WISP-0 with different techniques when only considering broken NWs ............................................................... 97
Figure 5.5: The yield-density products achieved for WISP-0 when only considering stuck-on FETs ................................................................. 98
Figure 5.6: The yield-density products achieved for WISP-0 when only considering broken NWs ............................................................... 99
Figure 5.7: The yield achieved for WISP-0 for various defect rates assuming clustered defective transistors ................................................................. 100
Figure 5.8: The yield achieved for WISP-0 for various defect rates when considering clustered broken NWs ................................................................. 101
Figure 5.9: The yield-density product achieved for WISP-0 when only considering stuck-on FETs ................................................................. 101
Figure 5.10: The yield-density product achieved for WISP-0 when only considering broken NWs ................................................................. 102
Figure 5.11: The yield achieved for WISP-0 when assuming transient faults .... 103
1.1 Scaling Limitations of CMOS

The great success of CMOS (Complementary Metal-Oxide Semiconductor) technology relies on the continuous improvements of lithographic and process techniques. The scaling theory (by C. Mead [9] and R. H. Dennard [54]) in the 1970s revealed the inherent relation between the electrical properties of MOSFETs (Metal Oxide Semiconductor Field-Effect Transistors) and their geometrical parameters. Since then, the scaling theory and its practical observation, Moore’s Law [23], have been effectively guiding us to build microprocessors with higher density, faster speed and lower power consumption.

This trend, however, is facing challenges and is expected to reach the fundamental barriers of charge-based devices in the next 15 years. MOS devices of 14-nm half-pitch (referred to as technology node) are expected to be available by that time according to the prediction by ITRS (International Technology Roadmap for Semiconductors [31]). However, integrating these MOS devices into reliable large-scale systems may be more challenging:

- As more transistors are integrated into the same die area, the power density of a microprocessor will soon climb beyond the capabilities of any possible cooling techniques in the future (see Figure 1.1).
- The defects and parameter variations during manufacturing are increasing, which will severely impact the functionality and performance of systems.
• The cost of manufacturing is skyrocketing. For example, typical cost for 45nm technology is as high as $1.5M per set [55]. This severely hampers product development.

![Device and Power Density Trends of Intel's Microprocessor](image)

**Figure 1.1: The growing device/power density of Intel’s microprocessors**

It has been pointed out that extending Moore’s Law beyond 10nm technology node with current lithography-based CMOS technology would be extremely difficult, if not unfeasible [20]. On the other hand, high density designs are still desirable in the future; as multi-core processors are developed for higher parallelism. Industry is therefore expecting new devices and new manufacturing technologies that can effectively scale down below 10nm.
1.2 Nanoscale Computing

Among the most promising alternatives to MOS are nanodevices. Nanodevices are different from conventional MOS devices not only in feature sizes, but in their unique structures and working mechanisms as well: the operation of MOS transistors is based on the movement of masses of electrons while nanodevices take advantage of quantum mechanisms which are prominent only at nanoscale. Nanodevices can be downscaled to a few nanometers and are capable of building ultra-dense computing systems ($10^{11}$ to $10^{12}$ devices/cm$^2$ [31]). In addition, nanodevices have the potential to build computing systems with higher performance and lower power consumption than the scaled CMOS technology. Non-conventional manufacturing techniques that are not based on lithography may reduce the manufacturing cost as well.

Designing nanoscale microprocessors is, however, facing many challenges, many of which do not exist in the conventional CMOS world. For example, the defect rate of nanodevices is expected to be around a few percent range [75], much higher than state-of-art CMOS technology (The defect density at 65nm technology node of CMOS logic is 1395 defects/m$^2$, which can be translated to a defect rate of $9 \times 10^{-8}$ per device [31]). This fact makes fault tolerance a critical issue in nanoscale computing systems. Any proposed microprocessor architecture must be able to handle the relatively high defect rates of devices.

Another key aspect is to preserve the density advantage of nanodevices. It is possible that due to manufacturing related layout/doping constraints in nanofabrics, the density advantage of nanodevices could be easily lost when assembled into computing systems, especially after fault tolerance techniques are applied.
Despite these challenges, the promising features of nanodevices are driving researchers to explore possible circuits and architectures that can deal with high defect/fault rates while still keep the density advantage over CMOS technology. In the rest part of this chapter, I will first provide an outlook on major emerging nanodevices that are promising for nanoscale computation and then introduce our proposed nanofabric called NASIC.

1.3 Emerging Technologies

Various nanoscale electronic devices have been demonstrated by researchers. In general they can be divided into two major categories based on their working mechanisms: solid-state and molecular devices. Controlling solid-state electronic devices is typically achieved by certain electrical effects (e.g., electrical/magnetic field, carrier density, and etc.) without changing fabrication materials. Molecular devices, on the contrary, are based on electrical/mechanic behaviors of individual molecules.

Molecular computing systems use a single or a few molecules as a switching device, which can be as small as 1nm. For example, bistable switches based on catenanes [13] and rotaxanes [71] have been demonstrated. The key challenge of molecular computing systems is finding appropriate molecules. Simple molecules are easy to build but do not provide enough functionality. For example, the devices in [13] and [71] show only 4X and 10X ratio of conductance between on and off states respectively, making them not suitable for logic but for memories only [32]. On the other hand, complex molecules such as a DNA strand are often unstable and sensitive to thermal fluctuations, making reliable computation based on these molecules difficult. The interfacing of molecular devices to microscale circuits is also a big challenge.
because of the diffusion of metal atoms into molecules [46]. Therefore, molecular devices do not seem promising for higher density computing systems than MOS devices in the near-term.

By contrast, solid-state nanoelectronic devices may not achieve as high density as molecular devices, but seem more practical with the manufacturing techniques that can be expected in 15 years. Examples of solid-state nanoelectronic devices include Quantum Dots (QDs) [15], Single-Electron Transistors (SETs) [41], resonant tunneling devices [39], and nanowire (NW) / carbon nanotube (CNT) -based devices.

A quantum dot is a semiconductor structure in which electron-hole pairs are confined in all three spatial dimensions. Without any transportation of electrons or holes, quantum dots interact with each other through Coulomb interaction [15]. Quantum Cellular Array (QCA) is perhaps the most promising QD-based computing architecture [15][29]. The main challenges of QD devices include the selective placement of devices and a complex clock design required for coordinating the interactions between QDs [14][42]. Such a clock design may cause severe power dissipation because of radiation and therefore eliminate the power advantage of QDs.

The most extensively studied resonant tunneling devices are Resonant Tunneling Diodes (RTDs) with a double barrier structure [35]. RTDs show a negative differential resistance (NDR) characteristic when the potential applied across the device is in a certain range. Hysteretic devices with multiple states can be built with RTDs [5]. RTDs are promising to build ultra high-speed circuitry for memories or logics. Unfortunately, it has been found that the NDR characteristic of RTDs is very sensitive to the parameters of tunneling barriers [35]. Massive manufacturing of RTDs with
reliable characteristic remains impractical at this time, which indicates that RTDs are not ready for computing systems yet.

Single Electron Transistors (SETs) are three-terminal devices based on Coulomb blockade [41] – a quantum effect where the number of electrons on an island (or dot) is determined by a gate voltage. The main challenge of SET-based systems is that to make the circuits reliable, one must make the circuit work at every low temperature (a few Kelvin) mitigating thermal fluctuation, or make the feature size of SETs below 1nm [41], both of which are difficult to achieve in practice.

The most promising underlying nanodevices today targeting digital applications are perhaps based on semiconductor nanowires and carbon nanotubes (CNTs). In contrast to the above nanodevices, transistors and diodes based on NWs and CNTs follow another type of quantum effect - ballistic transport, i.e., the movement of electrons in material without electrical resistance [21]. Ballistic transport is negligible in bulk devices (e.g., MOSFETs) because of scattering but is prominent when the feature size of devices is around a few nanometers. The lack of resistance provides the NWs/CNTs-based devices with great advantages on performance and power consumption.

Rapid progress is being made for synthesizing and assembling NWs or CNTs into feasible logic devices and circuits. NWs and CNTs of diameters in the order of a few nanometers with controlled doping profiles have been built. With the help of self-assembly manufacturing techniques such as fluidic alignment [23], chemical patterning [74], laterally self-assembled di-block copolymers [64], and Superlattice NAnowire Pattern transfer (SNAP) [45][22], we can construct NWs and CNTs into arrays and
place them perpendicular with each other, forming crossbar structure [74]. FETs and diodes can be formed at the crosspoints of a crossbar [75]. Logic gates have been built using crossed NWs [75] and CNTs [3]. IBM made important progress toward low power logic by developing complementary devices on the same nanotube and demonstrated a CMOS-like nano-inverter [49][55]. Since the electrical characteristics of NWs can be more reliably controlled than those of CNTs [72][80][2], many researchers believe that NW-based devices are easier to assemble into grids and computing systems in general. Our work is therefore mainly focused on nanoscale computing systems based on semiconductor NWs.

1.4 Research Goals

Several nanoscale architectures have been proposed by other researchers but they mainly target FPGA applications. Very little work has been demonstrated for microprocessors. The work presented in this dissertation is part of our efforts to build a high-density defect-tolerant nanoscale architecture that is more geared towards microprocessor designs.

We first identify the challenges we may face when integrating nanodevices into computing systems. In order to design high-density computing systems, we need to understand the characteristics of nanodevices, the technical constraints we need to work around and the corresponding implications at system level. Our research indicates that the manufacturing-related layout/doping constraints may severely impact on the effective system-level density. In addition, many conventional fault tolerance/testing techniques are not suitable for nanoscale designs because they are designed for very low
fault rates and assume arbitrary routing in fabrics. We therefore need new circuit and architecture designs and new strategies for fault tolerance.

The fabric we propose in this dissertation is called NASIC (Nanoscale Application-Specific Integrated Circuit) [11][67] which is based on 2-D tile-based nanogrids. NASIC designs are optimized for specific applications for higher density and better fault-masking. NASIC is a Nano/CMOS hybrid fabric. However, the usage of CMOS in NASIC is minimized for density and performance purposes. To deal with routing constraints, we propose a new type of dynamic circuit which enables efficient cascading and temporary data storage.

A complete microprocessor – WISP-0 (WIre-Streaming Processor) is designed on NASIC fabric for prototyping and evaluation purposes. WISP-0 exercises many design principles and optimization techniques of NASIC. Defect-free WISP-0 can be 39X denser than the equivalent design at 18nm CMOS technology node.

As for fault-tolerance, we first investigate different types of faults that may exist in NASIC fabric including permanent defects and transient faults. Then we propose built-in redundancy techniques at various granularities (e.g., transistor, circuit and architectural levels) to deal with these defects/faults. The goal of built-in redundancy in NASIC is to maintain the NASIC designs functional in the presence of faults, while carefully managing area tradeoffs. A logic simulator was developed to evaluate the impact of defects/faults and the efficiency of our hierarchical fault tolerance strategy. Simulations show that our built-in fault-tolerance techniques can enhance the yield significantly: without fault-tolerance techniques, the yield of WISP-0 is only 5% when the defect rate of transistors is 1%. After our fault-tolerance techniques are applied, the
yield of WISP-0 could be up to 18% when 10% transistors in WISP-0 are defective. We also extend the simulator for transient faults. Simulation shows that our built-in fault-tolerance techniques can effectively mask transient faults as well.

Due to the lack of detailed device information for devices that are assembled into grids, perfectly accurate calculation of speed and power is difficult at this time\(^1\). Nevertheless, this thesis presents preliminary estimations on speed and power consumption based on some of the device work that has been demonstrated. Results indicate that NASIC designs are at least comparable to CMOS designs in terms of speed and power and have potential for being an order of magnitude better depending on device level and defect-rate assumptions as well as materials used.

1.5 Contributions of Dissertation

The primary contributions of this dissertation include:

- A novel nanoscale fabric (NASIC) for microprocessor designs. NASIC designs use nanoscale FETs on 2-D semiconductor nanowires to implement logic functions, with the support of CMOS peripherals. While still based on cascaded 2-level logic style, e.g. AND-OR or NAND-NAND, NASIC designs are optimized towards specific applications for higher density. The fabric is based on simple 2-D grids and well aligned with the expected manufacturing constraints; its projected benefits scale with improvements in nano-manufacturing techniques.

\(^1\) Ongoing effort in the NASIC research group (together with collaborators at UCLA and CHM) is addressing cross-wire device design and manufacturing issues specifically for NASICs.
• A detailed manufacturing flow for NASIC is provided in the dissertation. NASIC manufacturing can be achieved with a combination of self-assembly and more conventional top-down process (e.g. photolithography). While many challenges remain, the key steps of the proposed flow have been demonstrated. We believe that NASIC is one of the most promising fabrics that can be realized in the near future due to its scaling property (wherein the fabric can be made increasingly denser as new nano-manufacturing approaches are developed).

• A novel dynamic circuit design is proposed in the dissertation for high-density digital logic, sequential circuits and efficient pipeline structure. Due to the routing and doping constraints, we found that it would be difficult to implement conventional sequential circuit structures (e.g. flip-flops) on 2-D nanogrids. We propose a new dynamic circuit style that can provide temporary data storage on nanowires. By using the new dynamic circuits, we can design area-efficient sequential circuits and datapaths with pipeline structure.

• A simple but complete prototype microprocessor (WISP-0) is presented in the dissertation. WISP-0 is a stream processor that implements a 5-stage pipelined streaming architecture. WISP-0 exercises many of the NASIC design principles and optimization techniques. With the help of dynamic circuit styles designed for NASICs, data is streamed across tiles with minimum control/feedback paths. Intermediate values during processing are temporarily stored on the nanowires without requiring explicit latching. WISP-0 is also used in this dissertation as a design example to evaluate various capabilities of NASIC fabrics including
density, yield, speed, and power consumption and allow a comparison with equivalent CMOS designs.

- We propose built-in fault-tolerance techniques to sustain the functionality of NASIC designs in the presence of defects/faults. Compared with reconfiguration-based approaches, our built-in fault-tolerance techniques can mask more than just permanent defects; the fabric masks transient faults and errors due to parameter variations in addition to defects. The built-in multi-layered strategy also simplifies the micro-nano interfacing: external access to every crosspoint in the nanogrids is not required (in contrast to reconfiguration based proposals). Our hierarchical built-in fault-tolerance techniques can be applied to NASIC designs at various granularities, such as transistor, circuit and architectural levels, while carefully managing area-yield tradeoffs. These techniques could be supported in future CAD tools.

- We develop a yield simulator to estimate the impacts of various fault sources and to evaluate the efficiency of our built-in fault tolerance techniques. Simulation results show that our built-in fault-tolerance techniques can improve the yields of NASIC designs significantly. For example, the yield of WISP-0 is up to 50% even if 15% transistors are defective (after combined fault tolerance techniques are applied). Simulation data for various fault tolerance scenarios and different fault sources is provided in the dissertation.

- Preliminary estimations on the speed and power consumption of NASIC designs are presented in the dissertation. Our calculations are based on available device characteristics and a lumped resistance-capacitance (RC) model. The results show
that WISP-0 might be able to run at about 57GHz and its power consumption is at least comparable to conventional CMOS technologies at the end of the ITRS roadmap.

1.6 Organization of Dissertation

The rest part of this dissertation is organized as follows:

Chapter 2 describes strategies for assembling NW-based devices into a nanotile – the building blocks of NASIC designs. A novel manufacturing process that can effectively build NASIC circuits within technical constraints is proposed in this chapter. We also identify the challenges of building sequential circuits on 2-D nanogrids and present our novel dynamic circuit designs that can implement sequential circuits efficiently.

Chapter 3 shows how to connect nanotiles into large scale systems. We first describe how to build multi-tile designs using nanowires for local communications between nanotiles. Later we present a complete design of a complete stream microprocessor built on 2-D NASIC fabric – WISP-0. Detailed design of every component of WISP-0 is shown. Then we compare NASIC fabric with other nanoscale computing architectures and highlight the unique features of the NASIC fabric.

Chapter 4 presents our fault-tolerance strategies for NASIC computing systems. First, we identify the main sources of defects/faults in the NASIC fabric and introduce a generic fault model for the NASIC fabric. We describe in detail our built-in fault tolerance techniques. A yield simulator is developed to evaluate various combinations of these techniques. We also compare our built-in fault tolerance techniques with reconfiguration-based approaches.
The density, speed, power consumption and yield of NASIC designs are evaluated in Chapter 5. An equivalent CMOS WISP-0 is designed in Verilog-HDL as the baseline. We scale it down to several future technology nodes predicted by ITRS and compare with the NASIC version. The yields of WISP-0 under various fault tolerance scenarios are presented to show the efficiency of our built-in fault tolerance techniques. The area-yield tradeoffs are also discussed in this chapter.

The dissertation is concluded with a summary of the primary results in Chapter 6.

1.7 List of Publications


CHAPTER 2
CIRCUIT DESIGN AND MANUFACTURING

2.1 Introduction

As mentioned in Chapter 1, semiconductor NWs can be grown and doped to a certain type (p or n type). With the help of self-assembly based manufacturing techniques such as fluidic alignment (e.g., Langmuir-Blodgett technique) [23] or SNAP [45][22], we can construct NWs and CNTs into arrays with nanometer pitches. Two NW arrays can be placed perpendicular to each other to build a 2-D crossbar structure. At the crosspoints FETs or diodes can be formed. Types of FETs (p or n type) are determined by the doping types of NW channels.

Given the above progress on manufacturing and assembly of devices, we believe that it is time to explore the possible strategies to build nanoscale circuits and systems based on NW devices. In this chapter, we first show how to build static ratio circuit designs for combinational logic. Then we will discuss the difficulties to implement sequential logic on nanogrids. To address this problem, we propose a novel dynamic circuit style which enables the pipeline structure on nanogrids while requiring no explicit latches. This dynamic circuit design significantly improves the density of NASIC designs. By changing the dynamic control scheme, we are able to implement arbitrary logic functions with only one type of device. This significantly simplifies the manufacturing. We also propose a complete manufacturing process for NASIC in this chapter.
2.2 Static NASIC Designs

2.2.1 Static Logic Gates on Semiconductor Nanowires

![Figure 2.1: Static AND logic on an n-type nanowire.](image)

Figure 2.1 shows a static m-input AND gate on an n-type NW and corresponding circuit model. $R_c$ is the resistance of contacts between the NW and power supply microwires ($V_{dd}$ and $V_{ss}$). The channel resistances of the input n-FETs ($R_{fet}(V_1)$, …, $R_{fet}(V_m)$) are determined by input voltages ($V_1$, $V_2$, …, $V_m$). The values of channel resistances would be $R_{on}$ and $R_{off}$ when the input voltage is $V_{dd}$ (logic “1”) and $V_{ss}$ (logic “0”) respectively. The output voltage can be expressed as

$$V_o = V_{dd} \left( \frac{R_{pull} + R_c}{R_c + \sum_{i=1}^{m} R_{fet}(V_i) + R_{pull} + R_c} \right)$$

Experiments show that $R_{on}$ is much smaller than the contact resistance $R_c (<1M\Omega)$ and $R_{off}$ can be up to 100G$\Omega$ [75]. To make the circuit work properly, the input of the pulldown FET is driven to a level such that its channel resistance ($R_{pull}$) is far larger than $R_{on}$ but much smaller than $R_{off}$. A simple analysis shows the circuit in Figure 2.1 is effectively an AND logic. Figure 2.2 shows the static AND and OR logics
available on NWs. Static AND logic is implemented on n-type NWs while static OR logic on p-type NWs.

![Figure 2.2: Static AND and OR logics on p-type and n-type nanowires.](image)

### 2.2.2 Designs of Static NASIC Tiles

Using self-assembly techniques, we can arrange a set of NWs parallel in a certain direction. Then another set of NWs are aligned perpendicular to the previous NWs. This way we can build a 2-D nanogrid. FETs can be selectively formed at the crosspoints on nanogrid to implement a certain logic function.
Figure 2.3 shows an example of a nanotile - the building block for NASIC fabric. All horizontal NWs are doped to n-type while the vertical NWs are doped to p-type. Together, they form a nanoarray surrounded by conventional microwires (MWs) for power supply and control signals (e.g. pull signals). On the left-hand side of nanoarray an AND logic plane is formed which selectively ANDs the input signals (i.e., $a_0$, $b_0$, $c_0$ and their complementary forms), generating minterms on horizontal NWs. On the right-hand side of nanoarray forms an OR logic plane, which selectively ORs the minterms from the AND plane, generating the output signals. The whole nanoarray functions as a 2-level AND-OR circuit. It is well known that with complementary input signals, we can design any logic function in the AND-OR 2-level logic style. For example, the design in Figure 2.3 implements a 1-bit full adder.
2.3 Proposed Manufacturing Process

Figure 2.4: Possible manufacturing steps for NASICs: a) Grow NWs and dope them. b) Align NWs to form a regular array. c) Coarse-grain metallization using a lithographic mask. d) Additional metallization using perpendicular NWs as a fine-grain mask. FETs are formed at the crosspoints indicated by the dashed circles.

NASIC designs can be manufactured with a combination of self-assembly and more conventional top-down manufacturing steps. Figure 2.4 illustrates the key steps of our proposed manufacturing process:

1. Prepare and align NWs

   Gold nanoparticles are typically used as seeds to grow NWs under the control of seed catalysts [72]. During growth, these NWs are lightly doped for semiconductivity [75] (see Figure 2.4.a). For single-type FET NASICs, only one type of doping is used for both horizontal and vertical NWs. For NASICs with both types of FETs, each NW set (horizontal vs. vertical) will need to be differently doped.

   NWs can be aligned into parallel horizontal and vertical sets as shown in Figure 2.4.b. The potential alignment techniques include fluidic alignment [23], chemical patterning [74], self-assembled diblock copolymers [64] or SNAP [22][45]. Fluidic
alignment is a low cost approach but the good control of pitches and orientation of individual nanowires is difficult to achieve [23]. With chemical patterning techniques, better order and regularity of nanowire arrays is possible. However, regular chemical patterns on substrate at nanoscale (e.g., 10nm pitch) are quite challenging [74]. Diblock copolymers can achieve very small feature sizes (less than 10nm). The key challenge is achieving long range order in lamellar phases. With SNAP, high resolution (13nm pitch) and good precision has been demonstrated for parallel arrays of both semiconductor and metallic nanowires. Crossbar structures of metallic nanowires have also been shown. We believe that similar approach can be used to build semiconductor nanowire crossbar. One challenge of SNAP is that the number of nanowires can be manufactured in parallel is limited by the size of superlattice. Depending on the NW pitch assumed, other approaches relying on soft lithographic techniques [8] to align NWs on a substrate might also be possible.

2. Create FETs, metallic interconnect between FET channels, gate regions, and form 2-D NW grid

Regions on both the horizontal and vertical NWs where there should be no FET channels are first metalized over with the help of a lithographic mask (see Figure 2.4.c). In general, deep sub-micron lithography may not provide sharp shapes [58]. While the lithography mask in NASICs require 2NW pitch resolution (e.g., 20nm x 20nm at a 10nm NW pitch), which may be fairly demanding, the shape and size of these regions do not have to be precise. A crosspoint area has a rectangular shape proportional with the NW width – as opposed to the typically larger NW pitch. A metalized crosspoint region can, therefore, be of any shape up to a 2NWx2NW square area - beyond that size
another crosspoint could be covered causing a stuck-on defect (i.e., when some FET channels, that should normally be part of the design, are metalized over). This process step is, therefore, likely less challenging than a lithographic process in conventional CMOS with a similar feature size requiring exact shapes, sizes, and straight edges. Lithographic techniques with a resolution required for this step have been reported in [6][7]. Nevertheless, we expect this process step to be a key factor in determining the actual NW pitch that can be manufactured. The misalignment of this lithographic mask could generate stuck-on defects. As will be shown in the evaluation chapter (Chapter 5), these defects can be masked fairly well with a combination of built-in fault tolerance techniques. In Chapter 5, we also explore the impact of larger NW pitches on the density of the WISP-0 design. A larger NW pitch could facilitate manufacturing designs, even before all process steps are worked out.

Metallization of the NW gate regions can be done for each set of NWs in conjunction with the previous metallization step. The required resolution for gate regions is fairly low as each logic plane will have either its entire horizontal or all its vertical NWs acting as gates. After being metalized, the gate regions will need to be covered with an oxide shell. Once this step is completed, a 2-D NW grid can be assembled by moving one NW set on top of the other.

A fine-grained metallization step (see Figure 2.4.d) is responsible for creating the FET channels, creating the metallic interconnects between the FETs, and extending the metallic segments created in the earlier metallization step. Before this step, the assembled 2-D NW grid contains some metallic regions corresponding to (i) crosspoints where no FET channels are needed and (ii) gate regions; other segments of the NWs
remain doped as required for the FET channels. FET channels can be distinguished at the crosspoints by using one layer of NWs as a fine-grained mask over the other layer during a final metallization step. This step needs to be completed for both dimensions of a nanogrid – flipping of the structure might be required. After this, channels are formed at grid crosspoints (see, for example, the process in [79] with NiSi), in both dimensions, because the top layer protects the bottom NW from being metalized over; at the same time, the FET channels become automatically connected with small metallic NW segments. Crosspoint regions that have already been metalized in the previous step would remain metallic and would not be affected by this step.

3. Microwires and contacts can be added with lithographic process steps.

   As discussed in this section, while key individual steps have been demonstrated in laboratory settings (e.g., FETs at NW crosspoints, NW growth and specialization, NW alignment, and fine-grained metallization with the help of NWs to create FET channels), combining the necessary manufacturing steps remains a challenging and unproven process. However, by working on nanoscale fabrics and architectures, the research community can expose these requirements and tradeoffs between manufacturability and system-level capabilities, fueling more focused research on manufacturing techniques required for assembling nanoscale systems.

2.4 Dynamic NASIC Circuits

   The designs shown in Section 2.2 are combinational circuits. There are several disadvantages with these designs. First, static ratio circuits require careful sizing of devices for correct logic, which sets additional constraints on manufacturing. Second, static power consumption due to direct-path currents would limit the scaling of NASIC
tiles to arbitrary sizes. In addition, registers and latches are required in microprocessor designs for datapath pipelining and control units. However, due to topological and doping constraints, designs of sequential circuits are very difficult on 2-D nanofabrics where arbitrary doping and routing are not feasible. In this section, we first address the challenges to implement sequential circuits on 2-D nanofabrics with uniform doping along each dimension. Then we show how to use dynamically cascaded circuits adapted to the nanogrid to solve these problems.

### 2.4.1 Challenges in Designing Sequential Circuits in NASIC Fabric

![Figure 2.5: 1-bit NAND flip-flop design on 2-D NASIC fabric. The layout on NASIC fabric is shown on the left. The right side shows the corresponding schematic.](image)

A key challenge when designing NASIC circuits is the low effective density of sequential circuits. Figure 2.5 presents an example of a flip-flop design that could be implemented in NASIC fabric. The relative low area utilization is due to a feedback path (highlighted with shadow box in the figure) that is required for sequential circuits. As we can see in the figure, the feedback path is inefficient in area on 2-D NASIC
fabric where the arbitrary routing of signals is difficult. One insight from this design is that implementing flip-flops is difficult on 2-D fabrics without arbitrary routing. As we will show, dynamic circuit styles are a more natural choice due to relatively simpler layout and sizing considerations; mapping into 2-D grids with limited customization is more likely possible.

2.4.2 Proposed Dynamic Circuit Designs, Nano-latches

We propose to adapt conventional dynamic circuits to our NASIC fabric. In conventional CMOS process, dynamic circuits are less prevalent than static CMOS designs since static CMOS is more resilient against noise/variations and has lower power consumption. However, static CMOS circuits require many customizations such as device sizing, arbitrary placement and routing, which may be difficult to achieve with nanofabrics. Therefore, in NASIC, a dynamic circuit style is a better choice as they have more regular interconnections and are easier to fit into 2-D grids. We can implement dynamic circuits at nanoscale with the help of control signals generated in CMOS. For example, the circuits in Figure 2.6 show how to implement basic logic functions in a dynamic style on semiconductor NWs. An important observation is that outputs can be easily inverted by interchanging the power supply and ground connections.
Figure 2.6: Dynamic circuits implementing AND, NAND, OR, and NOR logic.

Similar to static NASIC tiles, dynamic AND and OR logics can be used to construct dynamic tiles. The design of 1-bit full adder is shown in Figure 2.7. Compared with the static design in Figure 2.3, the pull transistors are replaced with pre(dis)charge and evaluate transistors (shown in the shadowed boxes around the nanoarray) for dynamic operations.
A well-known issue for traditional dynamic circuits is the cascading problem, which may lead to reduced noise margin and potentially malfunctions [33]. MOS based dynamic designs apply various techniques, such as adding a CMOS inverter between dynamic stages (also called Domino logic [33]), to guarantee the correct functions. Unfortunately, we have found that these techniques are not suitable for NASIC fabric due to the routing constraints on 2-D grids.

To address the cascading problem, we propose a new dynamic circuit design which enables register-like behavior on the NWs without explicit latching. A novel aspect of our dynamic NASIC designs compared with traditional dynamic circuits is the introduction of a \textit{hold} phase for cascading purpose. Figure 2.8 shows a waveform that illustrates the \textit{pre(dis)charge-evaluate-hold} phases for the circuit in Figure 2.7. The
CMOS control signals (*hdis* and *heva* for AND circuit, *vpre* and *veva* for OR circuit) are responsible for coordinating the operations.

![Waveform for cascaded dynamic AND-OR circuit](image)

*Figure 2.8: Waveform for cascaded dynamic AND-OR circuit. The hold phase is added for cascading purpose. The approach might be extended by making the phases asymmetrical to achieve better overall latency.*

*Predischarge* phase: *hdis* is set to “1” and switches the predischarge FET on. This gates the output of the AND circuit in Figure 2.7 to *Vss* (logic “0”).

*Evaluate* phase: *hdis* is set to “0” and switches the predischarge FET off and the output of AND circuit is evaluated by setting *heva* to “1”. For example, if the inputs (a₀, b₀ and c₀) are “111”, the output of the top horizontal NW would be pulled to *Vdd* (logic “1”). If one of a₀, b₀, and c₀ is “0”, the output would still be “0”.

*Hold* phase: *hdis* and *heva* signals are set to “0” and the corresponding FETs are both switched off. The values on the horizontal NWs evaluated in the previous phase are held and can be used as the inputs to the following OR circuits.

There are several advantages to our proposed dynamic style for NASIC designs. First, it reduces the need of using the explicit latching circuits presented in Section 2.4.1. This significantly improves the density of NASIC designs. Second, the operation of dynamic logics is ratio-less, i.e., insensitive to the sizes of FETs. This feature obviously
makes our design more resilient against process variations. The control signals can be strongly driven to maximize signal swing, making our designs more robust against crosstalk or soft errors. Third, and perhaps the most interesting aspect is that we can latch data on nano-wires without explicitly adding sequential circuits. We call the concept of the temporary storage on the wire *nano-latch* [10].

The value stored in a nano-latch could be lost due to leakage currents if a stage stays in a hold phase for too long. We will discuss this effect in detail in Section 2.7.

2.5 NASIC Designs with Single-type of FETs

In order to produce complementary FETs, two different types of doped NWs must be used. Complementary FETs have been demonstrated on NWs in zinc oxide [27], silicon [73], and germanium [2], but in all cases significant differences in transport properties were found between the two types, sometimes much greater than those seen in today’s traditional CMOS FETs. By suitably modifying the NASIC dynamic control scheme and circuit style, we can implement arbitrary logic functions with one type of FETs in NASICs. A design using only n-FETs will implement a NAND-NAND cascaded scheme whereas a design using only p-FETs will implement a NOR-NOR scheme. Logically, these are equivalent with the original AND-OR. These schemes may thus be used with manufacturing processes where complementary devices are difficult to achieve. The 1-bit adder example with only n-FETs is shown in Figure 2.9.
In addition to manufacturing benefits, single-type designs also improve performance since p-FETs are typically much slower than n-FETs because of lower carrier mobility [33]. This has also been observed for NW FETs [69]. Therefore a NAND-NAND based design can run significantly faster than its AND-OR equivalent by eliminating slower p-FETs. Section 5.3 will provide the evaluation on this in detail.

2.6 Combining Logic Families in 2-D NASIC Fabric

In the design shown in Figure 2.7, the outputs \((c_1, s_0)\) and their complementary versions \((\neg c_1, \neg s_0)\) are generated in the sum-of-product form of the inputs. The signals on horizontal NWs (excluding the control NWs such as \(veva\) and \(vpre\)) correspond to
different partial products. For example, the signal on the top horizontal NW corresponds to partial product $a_0b_0c_0$; the signal on the second NW corresponds to partial product $a_0b_0\overline{c_0}$. Each output signal is the sum of selected partial products.

From Figure 2.7, we can see that different output signals require different groups of partial products. The output $c_1$, for example, requires partial products $a_0\overline{b_0}\overline{c_0}$, $\overline{a_0}b_0c_0$, $\overline{a_0}b_0\overline{c_0}$ and $\overline{a_0}b_0c_0$ while the output $\overline{c_1}$ requires $a_0b_0c_0$, $a_0b_0\overline{c_0}$, $a_0\overline{b_0}c_0$ and $\overline{a_0}b_0c_0$. The observation here is that true output and its complementary version will require different partial products if both of them are implemented in the same AND-OR logic planes. With single-FET designs such as shown in [48] this would be NAND-NAND. We have mentioned in Section 2.4.2 that the complementary outputs can be relatively easily generated by interchanging the power supply and ground connections. This way we can generate complementary outputs in AND-NOR style; note that the complementary output would then need the same partial products as the true output. We may therefore reduce the number of required partial products (i.e., the number of horizontal NWs) if a different control scheme is used. This thinking leads to our new nanofabric.

We propose to combine AND-OR and AND-NOR logic families (or NAND-NAND and NAND-AND) into the same NASIC logic plane. This technique is called Heterogeneous 2-Level (H2L) logic [66]. H2L logic requires some modifications on the OR plane. For comparison, the new circuit for the same 1-bit full adder but with H2L logic technique is shown in Figure 2.10. Note that in the design of Figure 2.7, all output NWs ($c_1$, $s_0$, $\overline{c_1}$ and $\overline{s_0}$) in the OR plane connect to the $V_{ss}$ MW at the top and to the $V_{dd}$ MW at the bottom. In the design of Figure 2.10, however, all complementary output
NWs ($\neg c_1$ and $\neg s_0$) are connected to $V_{dd}$ and $V_{ss}$ MWs in the opposite way. All true outputs ($c_1$ and $s_0$) of the design in Figure 2.10 are generated by AND-OR logic while all complementary outputs ($\neg c_1$ and $\neg s_0$) by AND-NOR logic. The right logic plane in Figure 2.10 now combines OR and NOR functions in the same plane. Compared with the design in Figure 2.7, the partial product $a_0b_0c_0$ (corresponding to the top horizontal NW) is not necessary and therefore is removed from the new design in Figure 2.10. This way we can reduce the number of horizontal NWs and indirectly the overall number of transistors. The approach can be automated and applied on larger scale designs.

![Figure 2.10: a 1-bit adder using H2L logic.](image-url)
2.6.1 Manufacturing Implications

A key advantage of this new fabric is that it effectively improves the density but does not introduce any new manufacturing challenges. The only modifications that are made are at the connections from NWs to $V_{dd}$ and $V_{ss}$ MWs. This manufacturing step is accomplished at microscale in a fashion similar to the original fabric style. Compared with the design in Figure 2.7, we have changed the order of vertical NWs in Figure 2.10, effectively segregating the OR and NOR logics. This rearrangement of vertical NWs ensures that the nano-micro interfacing is still at the microscale. Hence no additional manufacturing constraints are imposed. As can be seen in Figure 2.10, the dynamic control scheme otherwise remains completely unchanged. True and complementary output NWs use the same control signals as before.

2.6.2 Fault Tolerance Implications

Another interesting benefit of the H2L logic is that it also improves the yield of NASIC designs. The reason is quite simple: The total number of horizontal NWs and associated FETs is reduced compared to the original design – we can get the job done with fewer transistors. For a given defect rate, the expected number of defects in a design is also reduced. A design can therefore achieve better yield with H2L logic as compared to the original AND-OR. In addition, we find that H2L logic enables voting technique at nanoscale to improve fault tolerance further. Details on this will be provided in Section 4.
2.6.3 Applicability to Other Types of 2-Level Fabrics

The H2L logic technique can be easily applied onto nanofabrics based on 2-level logic. For example, on nFET-only NASIC fabrics [48], we can design circuits based on NAND-NAND/AND logic families. In general, the approach can be applied in 2-level logic based designs.

2.7 Circuit Analysis

To validate the concept of single-type FET dynamic circuits and analyze the sensitivity of circuits to key device parameters, we verify the new dynamic circuits with circuit-level simulations.

Detailed physical models for crossed NW FETs are not available yet. However, various nanowire FETs based on different processes have shown electrical behaviors similar to MOSFETs. Furthermore, desired NW FET electrical characteristics can be controlled very well. For example, researchers have found that by changing the material of metallic gates, both depletion mode and enhancement mode FETs can be built [38]. Therefore we believe that simulations with MOSFETs are a good initial option for detailed functional exploration. In addition, we also use lumped RC wire load models based on actual physical geometries to predict performance and power implications.

Several important design considerations have to be taken into account to make dynamic circuits work properly [33]. These problems will likely become even more critical in single-type FET dynamic circuits because of the reduced noise margins with only a single-type of device: e.g., when n-FETs are used as pull-up devices signal deterioration can occur in cascaded circuits. We will address these issues in the following subsections.
2.7.1 Simulation Setup

Figure 2.11: (a) The layout of a 4-stage cascaded NAND circuit implemented on 2 NASIC nanotiles. (b) Corresponding circuit schematic of the 4-stage cascaded NAND showing all signals.

Table 2.1. Parameter Values for Devices and Circuits

<table>
<thead>
<tr>
<th>Parameters</th>
<th>LJG-6</th>
<th>PTM-32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Length</td>
<td>17nm</td>
<td>34.9nm</td>
</tr>
<tr>
<td>Effective Channel Length (L_{eff})</td>
<td>17nm</td>
<td>17nm</td>
</tr>
<tr>
<td>Channel Width</td>
<td>10nm</td>
<td>20nm</td>
</tr>
<tr>
<td>Effective Channel Width (L_{eff})</td>
<td>10nm</td>
<td>10nm</td>
</tr>
<tr>
<td>Model Level</td>
<td>6</td>
<td>54</td>
</tr>
<tr>
<td>V_{th0}</td>
<td>0.134V</td>
<td>0.356V</td>
</tr>
<tr>
<td>V_{dd}</td>
<td>1.2V</td>
<td>1.2V</td>
</tr>
<tr>
<td>Clock Frequency (f)</td>
<td>20GHz</td>
<td>5GHz</td>
</tr>
</tbody>
</table>
Table 2.1 summarizes the baseline parameters of the devices and circuits in our simulations. We use a 4-stage cascaded dynamic NAND circuit shown in Figure 2.11 to explain our results. We simulate two different MOS device models for the FETs in the circuit: one is the level 6 Lattin-Jenkins-Grove model (LJG-6) in HSPICE [28]; the other one is Predictive Technology Model metal gate 32-nm (PTM-32) process [50]. The performance and power estimates with nanowire FETs are shown in the following section. The operating frequency (shown in the table) for circuits with LJG-6 model is much higher than with PTM-32 because of the lower threshold voltage and smaller parasitic capacitances. Based on these models, in conjunction with the simplified NW FET model, we provide insights on the requirements for functional correctness and discuss issues related to cascading and noise margins.

We first run a simulation of the circuit in Figure 2.11 to verify correct signal propagation after several stages in a cascaded design. To achieve this, we set the inputs $di_2$, $di_3$ and $di_4$ to “1” for enabling signals $dix$ and $diy$ to propagate to final output $do_4$.

The waveform for the circuit with the PTM model is shown in Figure 2.12. Simulation with LJG-6 model produces similar results. The top 2 curves show input signals ($dix$ and $diy$) and the bottom 4 curves represent the output signals at every stage ($do_1$, $do_2$, $do_3$ and $do_4$). The diagram clearly indicates a correct signal propagation: the output of the first stage ($do_1$) is logic “0” only when $dix$ and $diy$ are both logic “1”. This signal is propagated (with signal inversion at each stage due to NAND logic) along the pipeline as indicated by the 4 arrows in Figure 2.12 while maintaining a 0.5V noise margin.
2.7.2 Signal Integrity Considerations

There is a key difference of our dynamic NAND circuits from traditional CMOS dynamic designs: we use n-FETs for precharge devices as opposed to p-FETs in CMOS. It is well-known that p-FETs produce strong “1” and weak “0” while n-FETs produce strong “0” and weak “1”. In traditional dynamic CMOS designs, pull-up networks are therefore implemented with p-FETs and pull-down networks with n-FETs. Such a design style ensures that both “1” and “0” are strong [33].

However, this is difficult to achieve in a 2-D nanoscale fabric: it would require different doping types at different positions on a single NW and more customization at
nanoscale. Although axial-variant doping profiles on a NW have been demonstrated [44], it would add significant manufacturing and alignment requirements when assembling larger circuit structures.

In contrast, in the NASIC fabric, we always assume consistent doping type on every NW and thus same type of associated FETs. We make this assumption so that NASICs could be built with a combination of self-assembly and conventional manufacturing. In NASICs, AND/NAND circuits only use n-FETs while OR/NOR circuits only use p-FETs.

In single-type FET design we would expect some signal degradation at the output node of each stage. It is important that this degradation is within an acceptable range for each logic stage and would not accumulate across stages.

The waveform in Figure 2.12 indicates that the signal degradation is in a tolerable range; this is despite the output of each stage not being precharged fully to $V_{dd}$ but to a lower value due to using n-FETs as pull-up devices. On the other hand, there is no degradation for an output “0” as n-FETs are strong pull-down devices. Therefore, a dynamic NAND circuit will generate either weak “1” or strong “0” at output nodes.

Keeping $V_{dd}$ at reasonable levels can guarantee enough margin between logic “0” and “1” for proper switching of devices.

We need to find if the signal deterioration due to weak “1”s would become a problem across multiple stages. An interesting insight of NAND-NAND circuits is that the $V_{th}$ degradation will not accumulate during precharge/evaluate operations. Note that as the signal propagates through the logic stages, the weak “1”s are consistently transformed into strong “0”s thus limiting accumulation of signal degradation to a
single stage. For example, a weak “1” at the node \(do_1\) in Figure 2.11 produces a strong “0” at node \(do_2\) because of the n-FET pull-down network.

As such, this inverting behavior prevents the accumulation of signal degradation effects in the pipeline. Basically, the signal as it propagates goes through stages that use n-FET pull-ups (introduces weakness) followed by n-FET pull-downs (fixes weakness). Another way to view this is that the dynamic approach works due to the NASIC logic style: NAND gates are followed by other NAND gates.

### 2.7.3 Charge Sharing

![Charge Sharing in Dynamic Circuits](image)

**Figure 2.13:** Left: Charge sharing in dynamic circuits. Right: Load transistor for alleviating charge sharing effect.

Charge sharing is another important challenge in dynamic circuits. Consider a single dynamic stage on the left side of Figure 2.13, the output node \(do\) is isolated from \(V_{dd}\) or \(V_{ss}\) during the **hold** phase. The charge is stored on the load capacitance \(C_L\). If the input \(di_1\) is switched from “0” to “1” - because of the precharge phase in previous stage-while the other input signal remains as “0”, the channel of the transistor gated by \(di_1\) is
The charge stored on $C_L$ is then redistributed between $C_L$ and $C_a$ and the voltage at the output node do will drop. A simple analysis provides the following expression for the relative voltage drop at the output node:

$$\frac{\Delta V_o}{V_o} = \frac{C_a}{C_a + C_L}$$

The circles in Figure 2.12 indicate some places where charge sharing happens, causing a 0.3V voltage drop.

One solution to charge sharing is to precharge the critical internal node (node $a$ in Figure 2.13) [33]. This solution, however, is not feasible in a 2-D fabric due to the routing constraints. Based on the above expression, increasing the load capacitance is another possible approach. This implies that more charge is stored at the output node prior to sharing. Since precise placement of capacitors on the nanogrid is difficult, we propose to add one or more nanowire FETs at the output node to function as load devices, as shown in the right side of Figure 2.13. The drain and source of the load device are grounded to avoid power dissipation. We can also tweak the effective load capacitance by using different number of load transistors - and thus improving noise margin. This strategy with load devices fits the NASIC fabric very well without adding any new manufacturing requirement.

Experiments show that the noise margin can increase from 0.5V to 0.8V with 5 load transistors. Similar results are obtained with the LJG-6 model. The cost of this is increased area and some performance lost due to larger load capacitance. By default, we assume 5 load transistors throughout the rest part of Section 2.7. We choose this based on empirical observation as it yields good signal integrity with minimal area overhead.
We expect the desired number of load devices to be reviewed for individual manufacturing processes and specific devices in the future.

### 2.7.4 Capacitive Coupling

The isolated output nodes during the hold phase are sensitive to crosstalk due to coupling effects of parasitic capacitors. Crosstalk is becoming more and more prominent in deep sub-micron processes as devices and wires are placed closer to each other. This is even more challenging in nanoscale systems based on NWs owing to smaller feature sizes and closeness.

A special case of capacitive coupling is called clock feedthrough. In this, the switching activity of the clock signal may cause voltage overshoot at output nodes due to the coupling capacitance between gate and drain/source [33]. Fortunately, clock feedthrough is not projected to be critical in NASIC designs as it does not appear to deteriorate noise margins. Possible explanations include cancellation of noise effects and the unique nature of the stage-wise dynamic control scheme. Coupling effects between data signals are quite random and behave in a fashion similar to transient faults. NASICs built-in fault tolerance techniques can mask faulty signals caused by crosstalk as was shown in [12].

### 2.7.5 Sensitivity to Parameter Variations

Variations in device parameters are becoming more prominent as technology scales down [18] and will impact the functionality of integrated circuits by causing delay variations. Examples of device parameters impacted include channel length, width, threshold voltage, etc.
The NASIC fabric is based on regular grid structure using bottom-up self-assembly manufacturing such as fluidic alignment [75] or SNAP [45][22]. The pitch between NWs is determined by the diameter and oxide thickness of NWs which can be controlled fairly well with the method described in [72]. Somewhat counter-intuitively, this type of manufacturing may not be impacted from variations in geometrical parameters (e.g. channel length and width) to the same extent as in photolithography-based CMOS. Threshold voltage variation, on the other hand, seems more critical in a NASIC fabric because of non-uniform doping on NWs.

![Noise Margin vs. Threshold Voltage](image)

**Figure 2.14: Noise Margins of a dynamic NAND-NAND circuit achieved with different threshold voltages.**

To provide an initial insight into the impact of threshold voltage variation, we change the $V_{th \theta}$ of all FETs simultaneously and simulate the circuit in Figure 2.11 with different threshold voltages. We measure the resulting noise margins at $do_4$.

The results are shown in Figure 2.14. As we can see from the figure, the dynamic NAND-NAND circuits work in a wide range of threshold voltages for both the LJG-6 model and the PTM-32 model.
Devices with negative threshold voltages (i.e. depletion mode FETs) are likely to be more difficult to make functional with this circuit style. The reason is that the channels of depletion mode FETs are switched on by default when no gate voltage is applied. The charge stored at the output nodes is more likely to dissipate during evaluate and hold phases (even if the some of the input signals are logic “0”). Enhancement mode FETs are preferable for dynamic NASIC circuits.

Nevertheless, the working ranges of threshold voltage are quite wide, which implies that these dynamic circuits are quite resilient against threshold voltage variation. We also assumed the scenario when all FETs are affected the same way. In practice, the variations would be distributed non-uniformly.

2.8 Chapter Summary

In this chapter, we presented the structure of NASIC tiles – the basic building blocks for NASIC designs. In a NASIC tile, logics are implemented in a nanoarray, which is surrounded by lithographic MWs. MWs provide power supply and some control signals. We showed the design of static nanotiles based on static ratio logic and illustrated the challenges of implementing sequential circuits with static logic on 2-D NASIC fabric. A novel dynamic circuit style was proposed enabling high-density digital logic, sequential circuits and efficient pipeline structure without explicit latches. This not only improved the area efficiency of sequential circuits in NASIC fabric, but provides temporary data storage on NWs as well. In fact, all NASIC tiles can be implemented in dynamic style. With some minor modifications on the control signals,

2 Enhancement mode FETs have been demonstrated on ZnO NWs [59], InAs NWs [51] and Si/Ge NW Heterostructures [38][70].
we can change the logic style from AND-OR to NAND-NAND, requiring n-FETs only. This substantially improves the manufacturability of NASIC fabric. We can also combine different logic families into the same NASIC fabric for higher density and better yield. The concept of nanoscale dynamic circuits was proved by circuit-level simulations. A complete manufacturing process for NASIC fabric was also presented.
CHAPTER 3

NASIC ARCHITECTURE AND WIRE-STREAMING PROCESSOR

3.1 Introduction

So far, we have shown the building blocks of NASIC designs – nanotiles. A new dynamic circuit design was also presented. In this chapter, we will show how to cascade dynamic nanotiles into a pipeline structure. We also mentioned that because of the unique control scheme of our dynamic circuits, the signal can be stored on NWs temporarily without requiring explicit latches. As will be shown in this chapter, this feature makes the cascading of nanotiles efficient in area. With the help of pipeline structure, a complex computing task can be split into several simpler pipeline stages.

Using the proposed cascading technique, we design a simple but complete stream processor called the WIre Streaming Processor (WISP-0) on 2-D NW fabrics. WISP-0 implements a canonic pipeline architecture of 5 stages in 5 dynamic nanotiles. We will present the detailed design of each block in WISP-0 in this chapter. In addition to providing a prototype of NASIC design, WISP-0 is also used as an example to characterize NASIC designs in terms of density, speed, power consumption, and compare it with a functionally equivalent CMOS version, which will be shown in Chapter 5.

3.2 Pipeline Structure and Multi-tile Design

As mentioned in Chapter 2, the addition of a hold phase in the dynamic control scheme enables the cascading of nanotiles for a pipeline structure. Figure 3.1 shows an example of 2-bit carry-ripple adder with 2-stage pipeline realized across two nanotiles.
Each stage implements a 1-bit full adder (shown in the dashed boxes). Some signals (e.g., \( a_1, \sim a_1, b_1, \sim b_1 \) at the first stage and \( s_0, \sim s_0 \) at the second stage) are simply propagated to the next stage without any processing (shown in the shadowed regions). By cascading, a 2-bit NASIC adder is effectively split into 2 dynamic nanotiles.

As shown in Figure 3.1, the communication between two adjacent nanotiles is realized through NWs that are across two nanotiles (e.g., NWs with signals \( a_1, b_1, c_1, s_0, \) and their complementary versions). In general, the local communication between adjacent nanotiles is provided by NWs while global communication is provided by MWs. This strategy ensures that most data propagation and evaluation happen only on NWs. This is in contrast with many other nanoscale fabrics requiring conventional devices to contribute to the logic implementation. For example, CMOL requires a
CMOS inverter between two stages of nanoscale OR logics. The bulky CMOS inverter definitely presents heavy capacitance load to the nanoscale circuits and thus has great impact on the achieved performance. NASIC designs, however, avoid such a problem by using MWs only for power supply and control. This strategy enhances the performance of NASIC designs as compared to other proposals.

3.3 Wire-Streaming Processor

![Figure 3.2: The floorplan of WISP-0.](image)

Using the concept of nano-latch and other NASIC design principles, we design a simple but complete stream processor, called WISP-0. WISP-0 implements a typical 5-stage pipeline architecture including *fetch, decode, register file, execute*, and *write back*. WISP-0 consists of five nanotiles, as shown in the floorplan in Figure 3.2. A nanotile is shown in the figure as a box surrounded by dashed lines. NWs are used to provide
communications between adjacent nanotiles. Each nanotile is driven by the surrounding
MWs, which are not shown in the figure. These nanotiles are designed in dynamic style
and are cascaded together with NWs to form a 5-stage pipeline. The proposed implicit
nano-latches are used to support pipelining in WISP-0.

The five nanotiles work as follows. The PC block implements the program
counter, which generates a 4-bit address each cycle. The ROM tile is a read-only
memory structure that can store up to 16 7-bit instructions. One instruction is fetched
from the ROM every cycle based on the address generated by the PC. The instruction
then is decoded into opcode and operands in the DEC (decoder) block. Next, the values
of the operands are read from the 4-entry register file (the RF block in the figure), and
are sent to the ALU for the execution of the instruction. The result will be written back
to the register file if necessary.

Currently, WISP-0 supports five instructions: nop, mov, movi, add, and mult.
Each instruction contains a 3-bit opcode and two 2-bit operands. The first operand
could also be used as the destination address if necessary.

The nop instruction is included to avoid RAW (read after write) data hazards,
because control logics and bypass networks are difficult to implement on 2-D fabrics
and would have very significant overall density impact due to the routing problem
shown earlier. In WISP-0, we assume that the compiler has the ability to create codes
without data hazards by inserting nop.

Next, we will present the detailed design of each block in WISP-0. For each
nanotile, the surrounding MWs are not shown in the following figures to improve the
readability. The transistors are simplified as dots as well.
Figure 3.3: The layout (left side) and schematic (right side) of the program counter in WISP-0.

The Program Counter (PC), as shown in Figure 3.3, implements a 4-bit accumulator. It is composed of a 4-bit incrementer (bottom half in the layout) and a 4-bit nano-latch (top half in the layout). The output of the incrementer is delayed for one cycle in the nano-latch and fed back to the incrementer in the next cycle. The instruction address, which is the output of the incrementer, is therefore increased by one in each cycle. This block shows an example of how to implement feedback networks in a single nanotile. We can implement an arbitrary finite state machine this way.
Figure 3.4: The layout (left side) of the instruction ROM in WISP-0. On the right side are the codes stored in the ROM.

Figure 3.4 shows an example of the instruction ROM; the corresponding codes stored are shown on the right side, which can not be modified during runtime. As we mentioned before, necessary \textit{nop}'s are inserted in the codes to avoid data hazards. The \textit{DEC} block takes instructions from \textit{ROM} and decodes them into opcodes and operands. Figure 3.5 shows the layout of the decoder.
Figure 3.5: The layout of the instruction decoder in WISP-0.

Figure 3.6: The layout (left side) and schematic (right side) of the register file in WISP-0.

Figure 3.6 shows the design of a 4-entry 2-bit register file. The data (and complementary formats) are stored on the bottom 16 horizontal NWs in the figure, which implements a 16-bit implicit nano-latch. The values of the registers are selected
by the 2-bit 4-to-1 multiplexer (2-bit MUX41 in the figure), and another multiplexer (2-bit MUX21) is used to choose between immediate values (for instruction movi) and register values. At the same time, opcode and dest (destination register address) are pipelined to ALU. If ALU needs to write results back to the register file, the data and register address will enter from the top right corner of the tile and the corresponding register value is updated by the logic in the bottom right.

Figure 3.7: The layout (left side) and schematic (right side) of the ALU in WISP-0.

Figure 3.7 shows the layout and schematic of ALU that implements both addition and multiplication functions. The arithmetic unit integrates an adder and
multiplier together to save area. It takes the inputs (at the bottom) from the register file and produces the write-back result. At the same time, the write-back address is decoded by the 2-to-4 decoder on the top and transmitted to the register file along with the result. The result will be written in the corresponding register in the next cycle.

The five blocks are combined together to form the stream processor WISP-0.

3.4 Comparison with Other Nanoscale Computing Architectures

Besides NASIC, several other nanoscale architectures have been proposed. 
Table 3.1 shows the comparison of four recent fabric designs including NASIC (our work), NanoPLA [4], CMOL [40], and a molecule-based fabric proposed by HP/UCLA [36][77]. NASICs use field-effect transistors (FETs) at nano-crossbar junctions to implement all logic, rather than diodes or molecular switches such as proposed by NanoPLA, CMOL and FPNI. NanoPLA uses FETs in decoders (this is required for addressing grid crosspoints and for reprogramming the fabric around defects) and for signal restoration. NASIC is also different from the other fabric schemes in the areas of applications targeted and fault tolerance: while most fabrics rely on reconfigurable devices, defect map extraction, and reconfiguration around defects, NASICs use built-in fault tolerance techniques at various levels to mask faults. Details of built-in fault tolerance techniques in NASIC fabric will be discussed in the next chapter.
Table 3.1. Comparison of Nanoscale Computing Architectures

<table>
<thead>
<tr>
<th></th>
<th>NASIC</th>
<th>NanoPLA</th>
<th>CMOL</th>
<th>FPNI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nanodevices</td>
<td>Single or complementary types of FETs</td>
<td>Diodes + FETs as restoration</td>
<td>Molecular switches</td>
<td>Molecular switches</td>
</tr>
<tr>
<td>Target Applications</td>
<td>ASIC-style logic, processors</td>
<td>FPGA logic</td>
<td>FPGA logic, Memory</td>
<td>FPGA logic</td>
</tr>
<tr>
<td>Defect Tolerance</td>
<td>Built-in defect tolerance at various levels of granularity</td>
<td>Reconfiguration</td>
<td>Reconfiguration</td>
<td>Reconfiguration</td>
</tr>
<tr>
<td>CMOS Roles</td>
<td>$V_{dd}/V_{ss}$ and dynamic control signals</td>
<td>$V_{dd}/V_{ss}$ defect map extraction, reconfiguration</td>
<td>Logic functions, signal restoration and reconfiguration</td>
<td>All logic functions</td>
</tr>
<tr>
<td>Manufacturing Challenges</td>
<td>Alignment during metallization of the crosspoints with no FETs for logic customization</td>
<td>Implementation of stochastic address decoder</td>
<td>Nano-micro interface: pins with different heights required; some alignment between nano grid and CMOS cells</td>
<td>Nano-micro interface: large pads; the upper nanowires bend down to touch pads</td>
</tr>
</tbody>
</table>

Most other fabrics are targeted and evaluated for logic applications targeting FPGAs and comparison is often done with CMOS FPGA logic. In contrast, the NASIC fabric focuses on microprocessor designs and datapath. All proposals face various manufacturing difficulties at this time. The CMOL fabric uses a somewhat challenging 2-level interconnect solution - with different height vertical pins that need to connect the CMOS cells to the nanogrids [40]. The NanoPLA approach requires complex defect map extraction and addressing decoder where all crosspoints need to be reached [4]. All fabrics with exception of NASICs assume the availability of reconfigurable devices. All designs use a variant of 2-level logic as underlying logic family.
3.5 Chapter Summary

In this chapter, we first showed how the new dynamic circuit design enables efficient cascading of NASIC tiles. Pipeline structures can be formed by cascading dynamic NASIC tiles without explicit latching. As a multi-tile application of NASIC fabric, we designed a stream microprocessor WISP-0 with a 5-stage pipeline, which is built in 5 dynamic NASIC tiles. Designs of 5 tiles of WISP-0 were shown illustrated in detail. A comparison of NASIC with three other nanoscale architectures was also presented.
CHAPTER 4

4 FAULT TOLERANCE

4.1 Introduction

Nanoscale computing systems face challenges not encountered in the world of traditional microelectronic devices. For example, although the manufacturing process is improving rapidly, the defect levels in nanofabrics are still at least a few percent range [75]. This fact makes fault tolerance a critical aspect in nanoscale systems. We have to build enough fault-tolerance to sustain functionality when a substantial fraction of devices are faulty.

The fault tolerance techniques are highly related to fault types and distributions (i.e., the fault model). In this chapter, we first introduce the fault model assumed in this dissertation. The model is based on the observations from other researchers and our proposed manufacturing. Next, we introduce and compare two main fault tolerance approaches that can be applied on nanoscale computing systems – reconfiguration and built-in redundancy. Then we describe in detail our hierarchical built-in fault tolerance techniques including n-way redundancy, weak pull-up/down NWs, code-based fault-masking and system-level voting. The evaluations of various fault tolerance techniques are provided in Chapter 5.
4.2 Fault Model in NASICs

4.2.1 Types and Sources of Errors

Sources of errors include permanent defects, process and environmental variation related errors, transient errors including internal and external noise related ones.

Permanent defects are mainly caused by the manufacturing process. The small nanowire dimensions combined with the self-assembly process, driven by the promise of cheaper manufacturing, is expected to contribute to high defect rates in nanoscale designs. Examples of permanent defects in NASIC fabrics would include malfunctioning FET devices, broken NWs, bridging faults between NWs, and contact problems between controlling MWs and NWs. For example, in a process that requires the metallization of segments connecting NASIC FETs, the channels of transistors could be metalized over and therefore stuck-on. The NWs used as gate control have a core-shell structure [75] and, therefore, if a shell is thicker than expected, the FETs controlled by these gates may have no bias applied. Prevalent defect types are also dependent on the types of transistors used. The FET channels will be conducting for depletion mode FETs [78] but will be cut-off for enhancement-mode FETs [57]. This means that when a FET has no bias applied it would be either always conducting (easier to tolerate) or would be cut off (much harder to tolerate) depending on its type.

Process variation related errors are caused by speed deviations due to device parameter variations. These errors occur typically for certain input combinations as a result of larger than expected circuit delays for those input combinations. While the actual parameter variation in NASIC depends on the manufacturing process ultimately
used (so this data is currently not available), research from deep sub-micron CMOS technology underlines the seriousness of this problem. We project that delay variations in NASICs would be caused by doping variations on the NWs used for channels and by channel length variations caused by the metallization process that separates FETs from each other (by creating small metallic interconnects between them) and they could be fairly significant.

Internal noise related faults caused by higher frequency and crosstalk between NWs are to be expected in fabrics like NASICs where NWs are placed close to each other. The NASIC control and the dynamic logic used could also affect noise margins. External noise factors such as radiation could be also present: with small dimensions, there might be an increasing likelihood that an α-particle, neutron or proton hitting the chip would cause transient faults. Other noise sources such as electromagnetic interference and electrostatic discharge could cause permanent faults [8].

Overall, we expect that these faults and process variation related ones will be less of a problem in NASICs compared to manufacturing defects, but factors to account for nevertheless. Our objective in the NASIC project is to address all these different sources of errors in a uniform manner with built-in fault tolerance techniques at fabric, circuit, and architecture levels.

4.2.2 Fault Model Assumed

In NASICs we consider a fairly generic model with both uniform and clustered defects and three main types of permanent defects: NWs may be broken, the transistors at the crosspoints may be stuck-on (no active transistor at crosspoint) or stuck-off (channel is switched off). A stuck-off transistor can also be treated as a broken NW.
The initial thinking is that the more common defect type is due to stuck-on FETs as a consequence of the metallization process used. NASIC fabrics require a mask at a 2NW pitch for one of their metallization steps (to avoid channels at crosspoints where no FETs are placed). Stuck-off FETs are also less likely especially in depletion mode fabrics. Recent thinking from [76] suggests that we will be able to control the reliability of NWs fairly well so broken NWs or stuck-off FETs will be likely less frequent than stuck-on FETs. In addition to permanent defects, other error sources such as due to process variation and transient faults are also discussed.

In this dissertation we consider defect rates of up to 15%. As suggested by other researchers, the defect levels in nanofabrics are in a few percent range [75]. During our initial work we found that defect rates greater than 15% would likely eliminate the density benefits of nanoscale fabrics compared to projected CMOS, in the context of microprocessor designs. Fabrics with higher defect rates might still be applicable as replacement technology for FPGAs and structured ASICs: e.g., if lookup-tables for programming of interconnect in FPGAs could be replaced with programmable devices; the lost density due to high-defect rates will likely be offset.

Defects could be uniformly distributed or clustered. In NASIC, we consider both cases. Accurate cluster defect models are not available because they are highly dependent on manufacturing. In this section, a cluster defect model is introduced. The model is somewhat preliminary but we believe it should be good for initial evaluation purposes.

We set a probability for defect clusters or cluster rate. FETs belonging to clusters would have greater probabilities to be defective than in defect models based on
uniformly distributed defects. Intuitively, the probability of a FET being defective decreases with increasing distances from the center of the cluster it belongs to.

Figure 4.1 shows how the probability of defects is modeled in a cluster. Parameters of this model include $a$, representing the probability of defects in nodes adjacent to cluster centers, and $n$ representing the maximum distance between the outmost defective transistors or NWs and the center; $n$ also determines the size of clusters. The impact of clustered defects and efficiency of proposed fault-tolerance techniques against clustered defects are provided in Section 5.5.3

Figure 4.1: A simple model for clustered defects; shows how defect probabilities are decreasing for FETs and NWs further away from a cluster center. With manufacturing approaches crystallizing, a more accurate defect cluster model would need to be developed.

Table 4.1 summarizes the fault model assumed in NASIC fabric:

Table 4.1. Fault Model for NASIC Fabric

<table>
<thead>
<tr>
<th>Fault Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stuck-on Transistors</td>
<td>Channels are always on. Prevalent in NASIC due to metallization process.</td>
</tr>
<tr>
<td>Stuck-off Transistors</td>
<td>Channels are always off. Very rare and only for enhancement-mode FETs.</td>
</tr>
<tr>
<td>Broken NWs</td>
<td>Nanowire segments between transistors are broken.</td>
</tr>
<tr>
<td>Transient Faults</td>
<td>Temporary faulty signals because of crosstalk/cosmic particles. Very rare.</td>
</tr>
<tr>
<td>Clustered Defects</td>
<td>Defective transistors and broken NWs are clustered together.</td>
</tr>
</tbody>
</table>
4.3 Reconfiguration vs. Built-in Redundancy

Two main fault tolerance approaches have been proposed to handle defects at nanoscale: reconfiguration and built-in redundancy.

If reconfigurable devices are available, defective devices might be replaceable after manufacturing. Reconfiguration based approaches, however, include significant technical challenges: (i) highly complex interfaces are required between micro and nano circuits for accessing defect maps and reprogramming around defects - this is considered by many researchers a serious manufacturing challenge due to the alignment requirement of a large number of NWs with programming MWs, (ii) special reconfigurable nanodevices are needed requiring unique materials with programmable and reversible characteristics, and (iii) an accurate defect map has to be extracted through a limited number of pins from a fabric with perhaps orders of magnitude more devices than in conventional designs.

In addition to the potentially intractable complexity, it might not always be possible to correctly extract such a map from a fabric with very high defect rates. Reconfiguration has been proposed at higher levels (e.g., node level in [34]) where it may not require a fully accurate defect map, assuming that self-checking at node level is supported. However, the number of devices and associated complexity of a single node might make the nanoscale implementation almost always defective.

Furthermore, reconfiguration-based approaches would primarily address permanent defects; it might be difficult, if not unfeasible, to work around faults caused by device parameter variations visible only for certain input combinations, or internal/external noise related faults that are transient.
Alternatively, as shown in this dissertation, we can introduce built-in redundancy at various granularities, such as fabric, circuit, and architecture levels, to make nanoscale designs functional even in the presence of errors, while carefully managing area tradeoffs. Such built-in redundancy could possibly address more than just permanent defects. Faults caused by speed irregularities due to device parameter variations, noise, and other transient errors could be potentially masked. Compared with reconfiguration based approaches, this strategy also simplifies the micro-nano interfacing: no access to every crosspoint in the nanoarray is necessary. Furthermore, a defect map is not needed and the devices used do not have to be reconfigurable. Built-in fault-tolerance techniques have been proposed in conventional PLA architectures [60]. Unfortunately, they all require arbitrary routing and placement, which seems difficult to achieve in nanofabrics due to manufacturing constraints. Next we will show our hierarchical fault tolerance techniques that can effectively tolerate various sources of faults within manufacturing/layout constraints in NASIC fabric.

4.4 Built-in Fault-Tolerance in NASICs

4.4.1 Circuit-Level and Structural Redundancy

Figure 4.2 shows a simple example of a NASIC circuit implementing an AND-OR logic function with built-in redundancy: redundant copies of NWs are added and redundant signals are created and logically merged in the logic planes with the regular signals. As shown in Figure 4.2, horizontal NWs are predischarged to “0” and then evaluated. Vertical NWs are instead precharged to “1” and then evaluated. The circuit
implements the logic function $o_1 = ab + c$; $a'$ is the redundant copy of $a$ and so on. Signal $a$ and $a'$ are called a NW pair.

A NASIC design is effectively a connected chain of AND-OR (or equivalent) logic planes. Our objective is to mask defects/faults either in the logic stage where they occur or following ones. For example, a break on a horizontal NW in the AND plane (see, for example, position “A” in the figure) causes the signal on the NW to be “0”. This is because the NW is disconnected from $V_{dd}$. The faulty “0” signal can, however, be masked by the following logic OR plane if the corresponding duplicated/redundant NW is not defective.

A NW break at position “B” can be masked by the AND plane in the next stage. Similar masking can be achieved for breaks on vertical NWs. Stuck-off FETs can be modeled as broken nanowires; the defect tolerance would work as described above. For
stuck-on FETs, the situation is relatively simpler as each FET has its redundant copy: if one of the two transistors is stuck-on, the circuit still works.

### 4.4.2 Improving Fault-Tolerance by Interleaving NWs

While the previous technique can mask many types of defects, faults at certain positions are difficult to mask. For example, if there is a break at position “C” in Figure 4.2, the bottom horizontal NW is disconnected from ground preventing predischarge. The signal on this NW may potentially retain a logic “1” from a previous evaluation. Because of OR logic on the vertical NWs, the two vertical NWs would then be set to logic “1”. Since both outputs on the vertical NWs are faulty, the error cannot be masked in the next stage. In Figure 4.2, the thicker segments along the horizontal NWs show the locations at which faults are difficult to mask. We call these segments *hard-to-mask* segments.
Figure 4.3: Interleaving NWs and adding weak pull-up/down NWs to reduce hard-to-mask regions. The bottom circuit has interleaved vertical NWs and weak pull-down NW between the AND and OR planes.

For nanotiles with multiple outputs, a particular arrangement of output NWs and their redundant copies could significantly reduce the size of hard-to-mask segments. This is shown in Figure 4.3: (a) presents a design in which each output NW and its redundant copy are adjacent to each other. In this arrangement, all segments to the right of the leftmost output NW pair ($o_1$ and $o_1'$ in Figure 4.3 (a)) are hard-to-mask. Alternatively, the interleaved version in Figure 4.3 (b), shows an arrangement in which the output NWs and their redundant copies are separated into two groups ($o_1$ and $o_2$...
form one group; \( o_1' \) and \( o_2' \) form another group). In this case, the size of the hard-to-mask segments is reduced. In general, the size of hard-to-mask segments can be reduced in larger scale designs to half, i.e., to half of the region covered by the vertical NWs plus the segment related to the control FET. This latter region is fixed and for most designs adds a negligible area. Interleaving is also helpful in masking clustered defects because duplicated NWs are set apart from one another.

### 4.4.3 Adding Weak Pull-up/down NWs

Even after built-in redundancy and careful interleaving, there are still some hard-to-mask segments remaining: for example, the thick lines in Figure 4.3(b). A possible solution to mitigate this problem is to insert weak pull-down vertical wires between the AND and OR planes. The idea is to pull down (or up depending on logic plane) floating inputs, due to broken NWs, that would cause logic faults: e.g., a floating “1” input to an OR plane that would make the OR logic always compute “1”. Modifying floating signals to a preferred logic level would allow masking in following logic planes.

A weak pull-down NW does not affect operation if there are no defects, but introduces a performance tradeoff when there are defects, by slowing the circuit down somewhat. They may also be difficult to manufacture and will contribute to leakage power.

At each crosspoint between a vertical pull-down wire and horizontal NWs there is a resistance created. This resistance has to be made larger than the switch-on resistance (estimated to be much smaller than 1M\( \Omega \) according to [74][75]) of a FET and smaller than the switch-off resistance (over 10G\( \Omega \)). To ease manufacturing one could also use MWs instead of the NWs to implement weak pull-up/down wires. Notice that
weak pull NWs are needed only if broken NWs are present in specific regions of designs.

### 4.4.4 Code-based Fault-masking Circuits at Nanoscale

There has been a considerable amount of work done in the field of coding for fault masking in logic in the past. Much of it is based on restoring logic that follows the logic in which faults may occur [37][16][41]. These approaches are problematic when working with nanofabrics because the fault rates are expected to be so high that the restoring logic would itself have faults in it. For example, with residue codes [30], components of the correcting circuitry are often assumed to be defect free, or operand lengths are increased prohibitively, such as in the case of arithmetic with stochastic computing and serialized data [61].

Error correction in general has been proposed by other researchers for nanoscale designs [17][47], however, error correction was used either in memory or at the interface between micro and nanoscale circuits. We are the first to apply a code-based fault-masking technique directly to a logic fabric, at the gate level, with significant layout constraints and the first group to evaluate a nanoscale processor design with a combination of code-based fault-masking, structural, and system-level techniques.

#### 4.4.4.1 Hamming Distance

The Hamming distance between two input codes is defined as the number of bits that are different. For example, the Hamming distance between “000” and “001” is 1. For the simple 1-bit adder design in Figure 2.7, the minimum Hamming distance
between the input codes is 1. Therefore, in that example, we cannot tolerate any defect on vertical NWs.

By adding redundant bits to the input signals, we are able to increase the minimum Hamming distance of input codes. In the 2-way redundancy example shown in Figure 4.2, the input codes are simply duplicated and the Hamming distance is increased to 2. With a minimum Hamming distance of 2, the design with 2-way redundancy can tolerate 1-bit error on the input signals. In the following subsection, we will show the required circuit-level modification to achieve error-masking with built-in code-base fault-masking circuits and redundant code signals, for a more efficient defect masking.

### 4.4.4.2 Error-Correcting Code Background

Achieving a certain Hamming distance between codes with minimum redundant bits is a well-known problem in the communication area. These codes are called error-correcting codes and are widely used to correct signal errors in noisy channels. Various kinds of error-correcting codes have been proposed and used; the Hamming code is one of the most popular codes due to its simplicity [1].

Considering a set of 3-bit codes \{“000”, “001”, “010”, “011”, “100”, “101”, “110”, “111”\}, the minimum Hamming distance between these codes is 1. By adding 3 redundant bits to the codes, we can achieve a Hamming distance of 3. The redundant bits (shown in parentheses below) are not unique according to the coding theory. An example of a new code set is \{“(000)000”, “(011)001”, “(101)010”, “(110)011”, “(110)100”, “(101)101”, “(011)110”, “(000)111”\}.
Obviously, this code set is more efficient than the one created by a simple signal duplication used in 2-way redundancy - which only achieves a Hamming distance of 2 with 3 added redundant bits.

In general, the number of required redundant bits is determined by the desired Hamming distance and the code width. For a given Hamming distance, the error-correcting code rate, defined as the ratio between the original data width and the width of all signals including redundant bits, approaches 1 as the original data width gets large [1] - which means the relative overhead goes down. For example, 11-bit wide data would only need 4 redundant bits to achieve a Hamming distance of 3.

This dissertation focuses on Hamming codes [24]; we also explored a variety of other techniques such as based on BCH codes [52] and found they are essentially of the same efficiency.

**4.4.4.3 Code-based Fault-masking in NASICs**

We first analyze how faulty signals impact circuit execution if there is no protection provided by fault-masking circuitry. We use the AND plane shown in Figure 4.4 (a) as an example (OR plane is omitted for clarity). It has 3 horizontal NWs corresponding to 3 different minterms of input signals. For example, the top horizontal NW corresponds to the minterm \(\neg a_0\neg b_0\neg c_0\) as three FETs on this NW are all on the complementary inputs. Therefore, the top horizontal NW is *selected* (i.e., the output of this NW is “1”) only when the input pattern is “000” during correct execution. Similarly, the middle and bottom horizontal NW correspond to minterms \(\neg a_0b_0c_0\) and \(a_0\neg b_0c_0\), and are selected by input patterns “001” and “101”, respectively.
However, in the presence of faults horizontal nanowires may be incorrectly selected. For example, if the input $\sim c_0$ is stuck at “1”, the FET in the white circle in Figure 4.4 (a) will be always switched on. This implies that the top horizontal NW can be selected by both “000” and “001”. Note that “000” and “001” differ only on the faulty bit ($c_0$). For a single bit input error, a horizontal NW can be incorrectly selected by some input pattern(s) that differ by only one bit from the correct input. Input patterns with 2 or more bit difference will not select the horizontal NW in the presence of a single bit error because at least one FET is switched off correctly (e.g., input pattern “101” will not select the “000” horizontal NW in this case because the FET in the dark circle in Figure 4.4 (a) is switched off).

We can define the Hamming distance between horizontal NWs as the number of bits where transistors are at complementary positions. For Hamming distance of $m$ between horizontal NWs, incorrect selection of NWs would not happen when there are no more than $m-1$ faults. Therefore, increasing the Hamming distance between horizontal NWs (e.g., by using additional code bits) can prevent incorrect selections, thereby providing fault resilience.
Error-correcting codes (e.g., Hamming code\(^3\)) have been used to efficiently increase Hamming distance between code words in communication theory. As far as we know, we are the first to incorporate Hamming code into logic circuits to tolerate fault rates as high as 10\%–15\%. We apply error-correcting code technique to achieve the desired Hamming distance between horizontal NWs.

In the example in Figure 4.4 (b), for the middle horizontal NW, FET positions are at \(~a_0, ~b_0\) and \(c_0\), which is translated to information bits “001”. Using the Hamming code relationship shown below, the corresponding code bits are “011”. Additional FETs are added to this horizontal NW using the same translation rule between FET positions and code bits. In this way, we can add code bits \((r_2, r_1\) and \(r_0\) and their complementary forms) to the original input signals adding error-masking FETs to every horizontal NW and achieving a Hamming distance of at least 3 between them. In a cascaded circuit, code bits are generated in the previous stage. At the very first pipeline stage, they can be generated in CMOS.

\[
\begin{align*}
    r_0 &= c_0 \oplus b_0 \\
    r_1 &= c_0 \oplus a_0 \\
    r_2 &= a_0 \oplus b_0
\end{align*}
\]

The code-based fault-masking approach does not detect/correct errors in the traditional sense of Hamming codes. Instead we use Hamming codes to determine FET positions in our logic circuits such that faulty signals can be masked implicitly.

\(^3\) Hamming codes are chosen for their simplicity and easy implementation with 2-level logic. They have small overhead in terms of area and number of transistors, which are desirable in high defect-rate fabrics. Other codes with similar characteristics such as BCH codes show very similar fault-masking capability.
Note that the fault-masking FETs do not change the correct outputs when the design is defect free. The key insight here is that *these FETs mask any of the original FETs in case they would generate a faulty 1*.

### 4.4.4.4 One-bit NASIC Adder with Code-based Fault-masking

We apply this code-based fault-masking technique to the 1-bit NASIC adder using the method described above. The new adder is shown in Figure 4.5.

![1-bit NASIC full adder with code-based fault-masking](image)

**Figure 4.5:** 1-bit NASIC full adder with code-based fault-masking. The circuits in the shadowed area are redundant circuits added for the purpose of error correction.

The fault-masking FETs for these 3 redundant bits are shown in the left-side shadowed area. Circuits in the left-side shadowed area are actually helping in providing the correct output on each horizontal NW (input to the OR plane); the right-side shadowed area is used to generate redundant output signals (for the fault-masking circuitry in the next stage). The redundant outputs ($o_2$, $o_1$, and $o_0$) satisfy:
\[ O_0 = S \oplus C_1 \]
\[ O_1 = S \]
\[ O_2 = C_1 \]

### 4.4.4.5 Code-based Fault-masking Combined with 2-way Redundancy

There is one issue with the code-based fault-masking technique: complementary signals are required for proper functionality. However the minterm signals on horizontal NWs are not complementary. Thus, it is not be feasible to apply the code-based fault-masking technique for defects on horizontal NWs. Creating a complementary version for each minterm is not feasible on a 2-D fabric with this type of 2-level logic. We therefore apply 2-way redundancy technique on horizontal NWs. As will be shown in the next chapter, the yield of WISP-0 can still be improved considerably with this hybrid approach.

### 4.4.5 Adding CMOS Voting

Voting based techniques such as TMR [53] have been used extensively in CMOS. The basic concept is illustrated in Figure 4.6. There are 3 identical modules (e.g. A_1, A_2, A_3) performing a given task. All three modules perform the task independently and their outputs (i.e., a_1, a_2 and a_3) are fed into a majority voting circuit (shown as a shadowed box with label “V” in Figure 4.6). The output \( a \) of the voting circuit is sent to the next stage.
Figure 4.6: Triple modular redundancy configuration in a pipelined system.

Voting techniques require that the probability of a defect in the voting circuit is much smaller than in the design they are applied to. This is clearly the case in conventional technology. TMR with nanoscale voting circuitry is difficult as is in NASIC designs because at 5-15% fabric defect rates the voting circuits themselves would be likely defective.

Nevertheless, in pipelined processor designs one could add TMR, e.g., with majority voting, at certain points in a design in CMOS, without affecting throughput significantly. We could vote either at each stage or on the final outputs. Voting helps where the other nanoscale techniques leave faulty outputs.

CMOS voting circuits are much more reliable than nanoscale circuits: the defect density at 65nm technology node of CMOS logic is 1395 defects/m\(^2\), which can be translated to a defect rate of 9X10\(^{-8}\) per device [31]. Compared with defect rates of nanodevices, this is negligible.

Therefore, \( P_{TMR}^0 \) (defined as the probability that voting result \( a \) is correct), is determined solely by the probability that each module (\( A_1, A_2 \) or \( A_3 \)) generates correct output, \( P_M \) [53]:

\[
P_{TMR}^0 (P_M) = P_M^3 + \binom{3}{1} P_M^2 (1 - P_M) = P_M^3 + 3P_M^2 (1 - P_M)
\]
Different voting schemes yield different probability of correct voting. For example, Six Modular Redundancy (6MR), which means voting on 6 redundant copies, yields the following probability of correct outputs:

\[ P_{6\text{MR}}^0(P_M, P_0) \]

\[ = P_M^6 + \binom{6}{1} P_M^5 (1 - P_M) + \binom{6}{2} P_M^4 (1 - P_M)^2 + \binom{6}{3} P_0 P_M^3 (1 - P_M)^3 \]

\[ = P_M^6 + 6P_M^5 (1 - P_M) + 15P_M^4 (1 - P_M)^2 + 20P_0 P_M^3 (1 - P_M)^3 \]

Note that the probability of correcting voting for 6MR is also determined by \( P_0 \), the probability that correct output of each module is 0. This assumes for the case that 3 signals to the 6MR voter is “1” and 3 signals are “0”, the voting results is always generated as “0”. Therefore there is a \( P_0 \) chance that voter provided the correct voting. If voter always generates “1” for this case, the \( P_0 \) in the above equation should be replaced with \( P_1 \) or \( 1-P_0 \). \( P_0 \) and \( P_1 \) are application specific but they can be approximated as 0.5.

CMOS voting can achieve extra benefits in NASIC designs. As mentioned before, complementary signals are needed in NASIC designs. If each nanotile has two extra identical replicas, 3 redundant original signals (e.g., \( a_1, a_2, a_3 \)) and their complementary forms (\( \sim a_1, \sim a_2, \sim a_3 \)) are generated. We can actually vote on 6 duplicated signals and achieve better fault tolerance than conventional TMR.

There are challenges for implementing voting circuits in CMOS in a nanoscale fabric including manufacturability: inserting CMOS voting circuits between nanoscale modules would need a complex interfacing between nano- and micro-circuits. In addition, performance would be severely impacted by CMOS voting circuits which will present heavy load capacitances to the nanoscale circuits. Density benefits of nanoscale
implementation may also be reduced. Nevertheless, *CMOS voting still provides us with an idea on the upper bound of the benefit from a voting technique.* In the next section, we will show our nanoscale voting technique enabled by the H2L logic presented in Section 2.6, which eliminates the manufacturability issues imposed by CMOS voting, while still providing significant yield benefits.

### 4.4.6 Voting at Nanoscale

#### 4.4.6.1 Nanoscale Voting with Unreliable Voting Circuits

We explored possible strategies to implement voting circuits at nanoscale. The challenge is to achieve reliable voting using unreliable nanoscale voting circuits: the final reliability $R_{TMR}$ depends not only on the reliability of each module but also on the reliability of voting circuits:

$$R_{TMR}(R_M, R_V) = R_V R_{TMR}^0(R_M)$$

where $R_V$ is the reliability of voting circuits. Similar expression is applicable for 6MR. The equations are based on similar assumption as in [53], i.e., voting circuits are independent from computing modules in reliability\(^4\).

\(^4\) This is relatively conservative since the voting result may still be correct even when 2 out of 3 modules and the voting circuit are faulty. However, this expression reveals the effective factors that determine the overall reliability $R$.  

75
Figure 4.7: The reliabilities of output \( R \) after the TMR/6MR voting circuits.

Figure 4.7 shows the overall reliabilities \( R \) with voting circuits given different \( R_M \) and \( R_V \). The black dash line represents the reliabilities of original signals. Three thin solid lines represent the reliabilities of TMR outputs and three thick lines for the reliabilities of 6MR outputs. From the figure, we can see that if the voting is perfect \( (R_V=1) \), it always improves the reliability. 6MR is more efficient than TMR (\( P_0 \) and \( P_I \) are assumed to be 0.5 here) but typically with the cost of more components. If the voting circuits could be faulty, voting helps only in a certain range of \( R_M \).

From (1), there are 3 possible ways to improve the overall reliability: a) improve the reliability of each module \( R_M \), b) improve the reliability of voting circuits \( R_V \), c) improve the voting logic itself (e.g., 6MR vs. TMR). Based on this discussion, we will show the implementations of nanoscale voting in AND-OR and new AND-OR/NOR fabrics and discuss the benefit of H2L logic based implementation.

4.4.6.2 Nanoscale TMR in AND-OR NASIC Fabric
As mentioned before, complementary signals are necessary in a fabric based on 2-level logic. An original signal and its complementary version can provide “dual-rail” redundancy. However, voting on “dual-rail” signals (e.g., \(a_1\) and \(\sim a_1\) in Figure 4.8) requires signal inversion, which is impossible with 2-level AND-OR logic on the 2-D grid. Therefore, as shown in Figure 4.8, the voters on original signals and complementary signals are separated from each other in a pure AND-OR fabric and the “dual-rail” redundancy is effectively unutilized. Simulations in Section 5.5 indicate that voting in pure AND-OR NASIC fabric helps only when the defect rates are very small. For higher defect rates, the nanoscale TMR actually deteriorates the overall yield.

4.4.6.3 Nanoscale 6MR in AND-OR/NOR NASIC Fabric

H2L logic proposed in Section 2.6 provides us with the flexibility to produce complementary outputs whenever necessary. Given this capability, we can vote only on
original outputs from logic tiles and generate the complementary signals in voting circuits using AND-NOR logic. As shown in Figure 4.9, there is no need to generate complementary output in each module (A₁, A₂ and A₃) for voting. Instead, we generate 3 more original copies (a₁', a₂' and a₃') for more redundancy and vote on all 6 signals (6MR). In the voting circuit, the original and complementary outputs are generated using H2L logic (see Figure 4.10). These act as inputs for the next logic tile.

![Figure 4.10: A 1-bit 6MR H2L voter.](image)
Note that the area of $A_1$, $A_2$ and $A_3$ in Figure 4.9 is actually smaller than in Figure 4.8 and they provide more redundancy (6 copies compared with 3 copies in Figure 4.8). By using H2L logic, we not only improve the voting scheme (use 6MR instead of TMR) but the yield of each computing module ($R_M$). Simulations (Section 5.5) indicate that the overall reliability is significantly improved.

In this dissertation, we introduce above fault-tolerance techniques into all parts of WISP-0 while simultaneously managing their area efficiency. The fault tolerance approach used is based on both structural/fabric redundancy, built-in code-based fault-masking circuitry at nanoscale, and CMOS/Nano-based voting at the architectural level. The combined techniques make redundant circuits more tuned for specific designs and better tradeoff between area overhead and fault tolerance can be achieved. The evaluation results of these aspects will be shown in the next chapter.

4.5 Chapter Summary

In this chapter, we discussed the fault tolerance issue for NASIC. We first introduced all possible sources of faults in NASIC fabric, including permanent defects and faults due to process variation and noise (internal and external). A fairly generic fault model was proposed for NASIC fabric. In NASIC, we consider stuck-on/off transistors and broken NWs. Both uniform distributed and clustered defects are included in the model.

Two main approaches have been proposed to deal with faults for nanoscale computing systems. Reconfiguration-based approaches are challenging as they require the complex nano-micro interfacing to access every nanodevices. By contrast, we proposed in this chapter to apply different built-in fault tolerance techniques at different
granularities to automatically maintain the functionality of NASIC designs even if a substantial fraction of devices are faulty.
CHAPTER 5

5 EVALUATIONS OF DENSITY, SPEED, POWER AND YIELD

5.1 Introduction

Nanoscale devices have shown great density advantage over conventional CMOS devices. This advantage, however, could be easily lost when they are integrated into computing systems due to topological, interconnect and doping constraints. Built-in redundancy required for fault tolerance would impact on the effective density further. In this chapter, we use our wire-streaming microprocessor WISP-0 as a design example to evaluate the achieved density after various fault tolerance techniques have been applied and compared with the equivalent CMOS design. The results indicate that WISP-0 with built-in redundancy is still 3–4X denser than the equivalent design with 18nm CMOS technology, the most advanced CMOS technology predicted by ITRS2006 [31]. Speed and power consumption of WISP-0 are also estimated and compared with the CMOS version (This work has been done in conjunction with other colleagues in my research group). To evaluate the effectiveness of our built-in fault tolerance techniques, a yield simulator is developed and is used to estimate the yields of WISP-0 with different fault tolerance techniques. The simulations show that our built-in fault tolerance techniques can effectively mask faulty signals due to permanent defects and transient faults due to internal or external noises. The yield-density products of WISP-0 are also discussed for analyzing area-yield tradeoffs.
5.2 Density

5.2.1 Comparison with Equivalent CMOS Processor

To evaluate the density benefits of NASIC designs, we compared NASIC WISP-0 with equivalent CMOS versions. We designed this processor in Verilog-HDL, synthesized it to 180nm CMOS. We derive the area with the help of the Synopsys Design Compiler tool. Next, we scaled it to various projected technology nodes based on the predicted parameters by ITRS, assuming area scales down quadratically with minimum feature size. For the purpose of this thesis, we assume that the CMOS version of WISP-0 is defect-free and no fault tolerance technique is applied. This means that our CMOS area numbers are fairly optimistic.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW-pitch</td>
<td>10nm</td>
</tr>
<tr>
<td>NW-width</td>
<td>4nm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technology Node (ITRS 2006)</th>
<th>MW pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>70nm</td>
<td>170nm</td>
</tr>
<tr>
<td>45nm</td>
<td>108nm</td>
</tr>
<tr>
<td>32nm</td>
<td>76nm</td>
</tr>
<tr>
<td>18nm</td>
<td>42nm</td>
</tr>
</tbody>
</table>
The normalized density of WISP-0 for the various scenarios is shown in Figure 5.1. Technology parameters used in the calculations are listed in Table 5.1. The notation used is: \textit{w/o Red} stands for WISP-0 without redundancy (or baseline); \textit{2-way} stands for AND-OR WISP-0 with 2-way redundancy; \textit{EC3/2way} denotes an AND-OR WISP-0 with code-based fault-masking using a Hamming distance of 3 on vertical NWs (nanowires) and 2-way redundancy on horizontal NWs; The prefix \textit{H2L} refers to the Heterogeneous 2-Level logic technique; \textit{2-way+Nano TMR} stands for 2-way redundancy plus nanoscale TMR (see Figure 4.8); \textit{H2L 2-way+6MR} stands for 2-way redundancy plus nanoscale 6MR using H2L technique(see Figure 4.9). The density values of WISP-0 with CMOS 6MR are not provided because the area overhead of CMOS voters are highly dependent on the manufacturing process. While other combinations are possible, we found these to be most insightful and representative. The 2-way redundancy techniques also incorporate the techniques discussed in Sections 4.4.1 and 4.4.2 (i.e., interleaving and weak pull-up/down wires).
We can see from the results that the area overhead of adding 2-way redundancy for the nanoscale designs is roughly 3X when MWs (microwires) in NASICs are assumed to have the same dimensions as MWs would have in 18nm CMOS technology. A WISP-0 design based on \textit{EC3/2-way} requires around 40\% more area than one based on 2-way redundancy for both horizontal and vertical NWs, but better fault tolerance is achieved (see Section 5.5). Nanoscale voting- brings an extra 3~4X area overhead because there are 3 copies of the nanoscale blocks plus the voters themselves.

The H2L logic technique improves the density of WISP-0 significantly for all possible scenarios. For NASIC WISP-0 without redundancy, at 45nm CMOS technology node assumed for its MWs, the density improvement of NASIC H2L over NASIC AND-OR WISP-0 is 26\%. After applying 2-way redundancy and nanoscale voting in both, the corresponding improvement would be 43\%. At 18nm CMOS technology node for MWs, without redundancy, this improvement is 41\%. After applying 2-way redundancy, the density improvement is 52\%. Overall, the density improvement increases for more advanced CMOS processes. This is because the area overhead of MWs in NASICs assuming MWs at 18nm technology is much smaller than at 45nm technology node and thus the corresponding area reduction from H2L technique is more prominent.

Overall, the density of a NASIC based WISP-0 remains at least 4X (\textit{H2L 2-way+Nano 6MR}) or 9X (\textit{H2L EC3/2way}) greater than the density of the corresponding CMOS processor at 18nm even after fault tolerance is added.
5.2.2 Impact of NW Pitch on Density

In the previous analyses, we assumed that the pitch between NWs is 10nm. While this has been demonstrated in the laboratory, it will take time until we can reliably manufacture larger designs at this scale (the same way as it took the semiconductor industry decades to refine lithography to today’s resolution). A larger NW pitch may come with lower defect rates and it will also be significantly easier to manufacture. For example, a 20nm pitch design would require the NASIC metallization masks at 40nm resolution: a much more doable undertaking than 20nm masks. On the other hand, as expected, a larger NW pitch will result in lower overall density so it is important to understand its impact at the system level.

The impact of a 20nm NW pitch on density is presented in Figure 5.2. Note that the density of H2L WISP-0 with EC3/2-way is still 3X better than 18nm CMOS technology. Even after nanoscale 6MR is applied, the density of WISP-0 is still comparable to 18nm CMOS technology. Note that the NASIC process provides higher density even with larger lithographic feature size (40nm instead of 18nm). This is a result of a high density interconnect structure combined with high-density logic (due to smaller area occupied by individual transistors) in a NW-based fabric. A plausible option might be to start manufacturing at a relatively lower density and gradually scale with improvements in nano-manufacturing.
Figure 5.2: Normalized WISP-0 density with different defect fault tolerance techniques assuming a 20-nm NW pitch at various technology nodes.

5.3 Speed

5.3.1 Parameter Assumptions

Table 5.2. Parameter Values Used for Speed Estimation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW-pitch</td>
<td>10nm</td>
</tr>
<tr>
<td>NW-shell thickness ($t_{si}$)</td>
<td>1nm</td>
</tr>
<tr>
<td>NW-width (w)</td>
<td>4nm</td>
</tr>
<tr>
<td>Dielectric Constant of SiO2($\varepsilon_r$)</td>
<td>2.2</td>
</tr>
<tr>
<td>Resistivity of Si ($\rho_{Si}$)</td>
<td>$10^{-5}$ Ωm</td>
</tr>
<tr>
<td>Resistivity of NiSi ($\rho_{NiSi}$)</td>
<td>$10^{-7}$ Ωm</td>
</tr>
<tr>
<td>NW-MW contact Resistance ($R_c$)</td>
<td>10 kΩ</td>
</tr>
<tr>
<td>Transistor ON Resistance ($R_{ON}$)</td>
<td>4 kΩ</td>
</tr>
</tbody>
</table>
To estimate the speed of NASIC designs, the following parameters are assumed: a NW-MW contact resistance of 10kΩ and resistivity values of $10^{-7}$ Ωm and $10^{-5}$ Ωm for NiSi and Si respectively were used in these calculations [79]. $R_{ON}$ for a transistor of length 5nm and width 4nm was calculated to be roughly 4kΩ. A nanowire pitch of 10nm, an oxide layer thickness of 1nm, and a dielectric constant of 2.2 were assumed. Table 5.2 summarizes all the parameter values used in our calculations.

5.3.2 Delay Calculations

A lumped RC model was used for the worst-case delay analysis. Expressions from [4] were used for capacitance estimation. These calculations take into account NW-NW junction capacitances and relatively realistic coupling scenarios. The coupling capacitance per unit length was found to be 39.04pf/m. The junction capacitance was found to be 0.652aF.

<table>
<thead>
<tr>
<th></th>
<th>Control NW(H)</th>
<th>Datapath NW(H)</th>
<th>Control NW(V)</th>
<th>Datapath NW(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pre/eva</td>
<td>pre/eva</td>
<td>pre/eva</td>
<td>pre/eva</td>
</tr>
<tr>
<td>PC</td>
<td>14.99</td>
<td>9.78</td>
<td>25.27</td>
<td>11.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25.27</td>
<td></td>
<td>11.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.08</td>
<td></td>
<td>4.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25.27</td>
<td></td>
<td>32.43</td>
</tr>
<tr>
<td>ROM</td>
<td>8.48</td>
<td>11.08</td>
<td>33.47</td>
<td>9.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>33.47</td>
<td></td>
<td>20.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.78</td>
<td></td>
<td>82.68</td>
</tr>
<tr>
<td>DEC</td>
<td>11.74</td>
<td>20.21</td>
<td>83.33</td>
<td>11.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.21</td>
<td></td>
<td>55.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>83.33</td>
<td></td>
<td>143.1</td>
</tr>
<tr>
<td>RF</td>
<td>27.38</td>
<td>26.73</td>
<td>98.21</td>
<td>9.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26.73</td>
<td></td>
<td>42.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>98.21</td>
<td></td>
<td>167.6</td>
</tr>
<tr>
<td>ALU</td>
<td>29.34</td>
<td>18.26</td>
<td>37.78</td>
<td>16.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18.26</td>
<td></td>
<td>30.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37.78</td>
<td></td>
<td>138.7</td>
</tr>
</tbody>
</table>

Table 5.3 indicates the capacitive loading on each tile of WISP-0 for different clock phases. During each phase, there is one control NW and one or more datapath
NWs switching. In the table “Control NW (H)” refers to a Horizontal precharge/evaluate signal. Since the precharge and evaluate control NWs in one plane are geometrically identical, the capacitive loading on these NWs is the same. “Datapath NW (V)” refers to datapath nanowires in the vertical plane. The capacitive loading during precharge and evaluate is dissimilar for datapaths owing to different lengths and coupling effects.

Table 5.4 shows the maximum delay for the tiles of WISP-0 with AND-OR scheme. $h_{dis}$ and $he_{va}$ stand for the discharge and evaluate phases on horizontal NWs respectively, $v_{pre}$ and $ve_{va}$ are the precharge and evaluate phases on vertical NWs. All delay numbers are in picoseconds. As we can see in the table, the delays are unbalanced among stages in WISP-0. Larger tiles tend to have longer delays for longer wires and more devices associated. The delays are unbalanced among operating phases in the same stage also. This is partially due to the asymmetric ON resistances between n- and p-FETs. The speed of a pipelined structure is determined by the slowest stage. Specifically, the speed of AND-OR based WISP-0 is therefore limited by the vertical NWs in ALU. The corresponding delay is shown in the shadowed cell in Table 5.4.

<table>
<thead>
<tr>
<th></th>
<th>$h_{pre}$</th>
<th>$he_{va}$</th>
<th>$v_{pre}$</th>
<th>$ve_{va}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>0.056</td>
<td>0.177</td>
<td>0.045</td>
<td>0.415</td>
</tr>
<tr>
<td>ROM</td>
<td>0.047</td>
<td>0.480</td>
<td>0.163</td>
<td>6.015</td>
</tr>
<tr>
<td>DEC</td>
<td>0.154</td>
<td>1.025</td>
<td>0.633</td>
<td>2.327</td>
</tr>
<tr>
<td>RF</td>
<td>0.289</td>
<td>1.492</td>
<td>0.501</td>
<td>5.699</td>
</tr>
<tr>
<td>ALU</td>
<td>0.153</td>
<td>0.775</td>
<td>0.392</td>
<td>11.138</td>
</tr>
</tbody>
</table>
Table 5.5 shows the maximum delay for the tiles of WISP-0 with NAND-NAND scheme. The delays of horizontal operating phases are identical to the earlier AND-OR scheme. However, the vertical operating phases could be significantly faster due to the lower ON resistance values for n-FETs. In fact, the delay of the *veva* phase for NAND-NAND ALU is almost half the value for AND-OR ALU.

Assuming a 1/3 duty cycle (required by the *precharge-evaluate-hold* control scheme), the maximum clock frequency we found to be 30GHz for the AND-OR WISP-0 and 57GHz for NAND-NAND WISP-0 respectively. This comparison indicates almost 2X speedup for n-FET NASIC compared to the original version with 2 types of FETs.

Table 5.5. Delay (ps) – WISP-0 with NAND-NAND Logic

<table>
<thead>
<tr>
<th></th>
<th>hpre</th>
<th>heva</th>
<th>vpre</th>
<th>veva</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>0.056</td>
<td>0.177</td>
<td>0.032</td>
<td>0.231</td>
</tr>
<tr>
<td>ROM</td>
<td>0.047</td>
<td>0.480</td>
<td>0.106</td>
<td>2.955</td>
</tr>
<tr>
<td>DEC</td>
<td>0.154</td>
<td>1.025</td>
<td>0.475</td>
<td>1.512</td>
</tr>
<tr>
<td>RF</td>
<td>0.289</td>
<td>1.492</td>
<td>0.380</td>
<td>3.315</td>
</tr>
<tr>
<td>ALU</td>
<td>0.153</td>
<td>0.775</td>
<td>0.304</td>
<td>5.857</td>
</tr>
</tbody>
</table>

### 5.4 Power Consumption

#### 5.4.1 Dynamic Power

In CMOS designs, the dynamic power consumption can be expressed as:

$$ P_{dyn} = \alpha C_L V_{dd}^2 f $$
Where $\alpha$ is the switching activity factor, $C_L$ is the total load capacitance, $V_{dd}$ is the power supply voltage and $f$ is the clock frequency. The switching activity for static CMOS is the probability that the signals of internal nodes are switched from logic “0” to “1” [33]. The switching activities are therefore application specific and vary from a few percent (e.g., control units) to 40% (e.g., register files) [43].

NASICs use dynamic circuits. For dynamic circuits, power is calculated in a different way. For example, in a dynamic NAND circuit, every occurrence of “0” at the output node would cause an energy-consuming transition i.e., precharge-discharge. Therefore the switching activity factor for a dynamic NAND circuit is the probability that logic “0” occurs at an output node. In NASICs, since the signals are in a complementary form, the switching activity would be around 50%.

To provide a basis for comparison, we implemented an equivalent WISP-0 microprocessor in Verilog-HDL and synthesized it with a commercial 180nm CMOS process. The Synopsys Design Compiler (DC) reports that at 20MHz and 1.6V power supply voltage, the total dynamic power consumption of the CMOS WISP-0 is 215µW. We scaled this number to 18nm CMOS technology node using the above equation, assuming to a first order that the load capacitance scales linearly and that $V_{dd}$ for 18nm CMOS would be 0.7V based on the ITRS roadmap [31]. A simple calculation shows that at 57GHz operating frequency, the power consumption of the CMOS WISP-0 at 18nm would be 1,173µW.

---

5 The switching activity values provided in [13] need to be multiplied by 0.5 because half of the switchings are from “1” to “0” without dynamic power consumption.
6 The switching would be a little more than 50% if we take the precharge/evaluate transistors into account.
An initial estimation of the dynamic power of NASIC WISP-0 was done at the 57GHz operating frequency for a range of operating voltages between 0.7V-1.4V – we expect this to be the typical operating voltage range based on NW FET devices previously shown [70].

The expression used for this purpose is:

\[ P_{\text{dyn}} = \sum_{\text{pre,eva}} \left( C_{L1} + N \cdot C_{L2} \right) \cdot V_{dd}^2 \cdot f \]

where \( f \) is the clock frequency, \( C_{L1} \) is the capacitance on the control nanowire and \( C_{L2} \) is the capacitance on a datapath nanowire. \( N \) is the number of datapath nanowires switching simultaneously. In cases where \( N \) is variable (e.g., application specific), an average value is chosen assuming a 50% switching probability.

Table 5.6 shows the estimated power consumption for each tile in NASIC WISP-0. As expected, large tiles such as the Register File (RF) consume more power than others. The total power consumption of NASIC WISP-0 with 0.7V power supply is 343\( \mu \)W.

<table>
<thead>
<tr>
<th>( V_{dd} ) (V)</th>
<th>0.7</th>
<th>1</th>
<th>1.2</th>
<th>1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>15.3</td>
<td>31.1</td>
<td>58.7</td>
<td>101.2</td>
</tr>
<tr>
<td>ROM</td>
<td>26.8</td>
<td>54.6</td>
<td>102.8</td>
<td>176.9</td>
</tr>
<tr>
<td>DEC</td>
<td>70.0</td>
<td>142.8</td>
<td>268.7</td>
<td>462.7</td>
</tr>
<tr>
<td>RF</td>
<td>199.0</td>
<td>406.2</td>
<td>763.9</td>
<td>1315.8</td>
</tr>
<tr>
<td>ALU</td>
<td>32.0</td>
<td>65.4</td>
<td>122.9</td>
<td>211.9</td>
</tr>
<tr>
<td>Total</td>
<td>343.1</td>
<td>700.1</td>
<td>1317</td>
<td>2268.5</td>
</tr>
</tbody>
</table>
This implies that a NASIC WISP-0 consumes 29% power of the CMOS version shown in the previous subsection. However, it must be noted that with high defect rates expected in nano-manufacturing, redundancy based fault tolerance techniques are expected to be incorporated into NASIC for yield purposes as well as in CMOS. This would require more devices and larger capacitive loading in the design. In conclusion, we thus expect the active power consumption of WISP-0 with built-in redundancy to be comparable or better to the end-of-roadmap CMOS version depending on the levels of redundancy added to both fabrics. One might also be able to incorporate various power optimizations in NASICs. We currently are exploring techniques in this direction at both circuit, e.g., body biasing the NW FETs, and architecture levels, e.g., by removing redundant switching activity (if the switching frequency is very low, we can put components into a sleep mode by turning precharge and evaluate transistors off, a technique similar to clock gating in CMOS, reducing the overall switching activity significantly).

5.4.2 Leakage Power

We evaluated the leakage power for NASIC WISP-0. The leakage power is chiefly dependent on the $R_{OFF}$ of the NW FETs. While NW FET ON/OFF ratios up to $10^5$ have been reported [75], for the purpose of providing a conservative estimate, a ratio of $10^3$ has been assumed here (i.e., $R_{OFF}=3.75\text{M}\Omega$). The circuit may lose functionality at lower ratios. Also maximum leakage occurs in scenarios where only one FET on the NW is switched off and all others are on. Therefore the maximum leakage power for each NW can be expressed as:
The calculated leakage power for each tile of WISP-0 is shown in Table 5.7.

The total leakage power for NASIC WISP-0 is 46.9µW at 0.7V power supply, which is 13.7% of the dynamic power for WISP-0. As in the case of dynamic power, incorporating fault tolerance is expected to increase leakage power owing to more devices and larger area.

\[ P_{\text{leak}} = \frac{V_{dd}^2}{R_{OFF}} \]

The leakage power of CMOS WISP-0 at 180nm technology node is 305nW as reported by Design Compiler. It is expected that CMOS WISP-0 at 18nm technology node would consume proportionally much more than this because leakage power is exponentially related to feature size. To this end, our conclusion is that even after including leakage power, we expect that NASIC designs based even on a heavy fault tolerant logic to be comparable or have a slight advantage compared with CMOS versions. We expect that improvements on nanomanufacturing would reduce the needed

<table>
<thead>
<tr>
<th>V_{dd} (V)</th>
<th>0.7</th>
<th>1</th>
<th>1.2</th>
<th>1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>6.01</td>
<td>12.27</td>
<td>17.66</td>
<td>24.04</td>
</tr>
<tr>
<td>ROM</td>
<td>5.49</td>
<td>11.2</td>
<td>16.18</td>
<td>21.95</td>
</tr>
<tr>
<td>DEC</td>
<td>7.58</td>
<td>15.47</td>
<td>22.27</td>
<td>30.31</td>
</tr>
<tr>
<td>RF</td>
<td>20.12</td>
<td>41.07</td>
<td>59.14</td>
<td>80.49</td>
</tr>
<tr>
<td>ALU</td>
<td>7.71</td>
<td>15.73</td>
<td>22.66</td>
<td>30.84</td>
</tr>
<tr>
<td>Total</td>
<td>46.91</td>
<td>95.74</td>
<td>137.91</td>
<td>187.63</td>
</tr>
</tbody>
</table>
redundancy in NASIC and potentially increase its power benefits compared to projected state-of-the-art CMOS. Also, even CMOS designs will need redundancy at 18nm: the verdict on power consumption thus depends on various assumptions on defects and built-in support for resilience as well as performance points utilized. For example, if the performance of NASIC designs is 10X higher than CMOS designs, we can slow down the clock to 0.1X and the power consumption of NASIC designs could be 1/10 of CMOS designs. This provides designers with the flexibility for trade-offs between performance and power efficiency.

5.5 Yield

Using the approaches described in Chapter 4, we can incorporate the built-in fault tolerance techniques into all circuits of WISP-0. We used our NASIC CAD tools (developed with collaboration from the University of Brest, France) to modify WISP-0. To verify the efficiency of our fault-tolerance approaches, we developed a simulator to estimate the yield of WISP-0 for different defect rates and also considered other error sources.

5.5.1 Yield Evaluation of WISP-0

The simulation results for permanent defects are provided in Figure 5.3 (assumes defective FETs) and Figure 5.4 (assumes broken NWs). First, we present results assuming that defects are uniformly distributed. Clustered defects are addressed in separate subsequent subsections.

We assumed two types of defects - stuck-on transistors and broken NWs - in our simulations. Stuck-off transistors are treated as broken NWs as they have the same
effect. In addition, as discussed in Chapter 4, we project stuck-off FET defects and broken NWs to be less prevalent than stuck-on FETs.

From the results, we can see that our fault tolerance improves the yield of WISP-0 significantly. The yield of WISP-0 without redundancy is 0 when defect rate is only 2%. With 2-way redundancy, however, the yield of WISP-0 is still more than 20% when 5% transistors are defective. As expected, code-based fault-masking technique achieves better yield than 2-way redundancy. Compared with a 2-way redundancy approach, the improvement of the hybrid approach (EC3/2-way) on the yield of WISP-0 is 14% when the defect rate of transistors is at 2%, 129% at 5% defect rate, and 28.6X at 10%. Note that the improvement is greater for higher defect rates. Similar improvements are achieved for broken NWs.

The H2L logic technique improves the yield considerably. Compared with the AND-OR approach in 2-way Red+TMR scenario, the improvement of H2L logic on the yield of WISP-0 with 2-way Red+6MR is 15% when the defect rate of transistors is 2% and 147% at 5% defect rate.

As shown in Figure 5.3 and Figure 5.4, CMOS 6MR technique improves the yield of WISP-0 dramatically. This, however, would not be practical until the manufacturing challenges mentioned in Section 4.4.5 are well addressed. In this thesis, the yield values of CMOS 6MR are provided as the upper bound on voting schemes.

Nanoscale voting eliminates the manufacturing issues imposed by CMOS voting. From Figure 5.3 and Figure 5.4, we can see that nanoscale TMR technique in AND-OR NASIC fabric improves the yield when the defect rate is low. However, the

---

7 More complex designs would be faulty at even lower defect rates.
improvement vanishes for higher defect rates. For example, if the defect rate for transistors is higher than 7% or the defect rate for broken NWs is higher than 3%, then nanoscale TMR actually deteriorates the overall yield. This is because with high defect rate, the voting circuits themselves become so unreliable that they negatively impact yield.

With H2L logic, the effectiveness of nanoscale voting is significantly better. In addition to the yield improvement for the logic itself, the nanoscale 6MR technique in new NASIC fabric consistently improves the yield of WISP-0. Compared with WISP-0 with H2L logic and 2-way redundancy, the improvement of nanoscale 6MR technique is 7% and 47% respectively when 2% and 5% transistors are defective respectively. For broken NWs, similar results are achieved.

Figure 5.3: The yield achieved for WISP-0 with different techniques when only considering stuck-on transistors.
Figure 5.4: The yield achieved for WISP-0 with different techniques when only considering broken NWs.

Voting-based techniques provide good yield improvement but require relatively large area overhead. For example, as shown in Figure 5.3 and Figure 5.4 nanoscale 6MR combined with 2-way redundancy achieves higher yield than 2-way alone. This comes, however, at a cost of more than 3X larger area (see Figure 5.1). Therefore, it is important to understand the area overhead (or impact on density) of the different fault-tolerance techniques in conjunction with their fault masking ability.

5.5.2 WISP-0 Yield-Density Product Evaluation

To evaluate the tradeoff between yield improvement and area, we also consider the yield and density together in a combined metric. The yield-density product is a comprehensive indicator for the efficiency of different defect-tolerance techniques; it represents the ratio between the benefit (yield of designs) and its cost (area overhead).

The yield-density product results for various defect rates are presented in Figure 5.5 and Figure 5.6 respectively. We can see that the proposed built-in fault tolerance techniques significantly improve the yield-density products of WISP-0. H2L WISP-0
with 2-way redundancy achieves the best yield-density product as H2L logic technique improves both yield and density. Nanoscale voting techniques (i.e. *Nano TMR* and *Nano 6MR*) improves the yield of WISP-0 with more than 3X area overhead, deteriorating their yield-density products. Future NASIC CAD tools can take yield and yield-density product as different design constraints: if designers care more about yield, nanoscale voting may be necessary. However, if both yield and density are critical, some less area expensive fault tolerant schemes such as 2-way redundancy or code-based fault-masking technique are desirable.

![Stuck-on Transistors](image)

**Figure 5.5:** The yield-density products achieved for WISP-0 when only considering stuck-on FETs.
5.5.3 NASICs with Clustered Defects

In our previous results we assumed that all defects are uniformly distributed. However, defects can also be clustered as a group of adjacent FETs or a group of adjacent NWs could be damaged during the manufacturing process. In a 2-way redundancy scheme, if clustered defects make two redundant signals faulty, these faults cannot be masked. However, if the same two redundant signals are placed far-enough apart, clustered defects will be unlikely to make them faulty simultaneously.

5.5.4 WISP-0 Yield with Clustered Defects

Figure 5.7 shows the yield of WISP-0 assuming clustered transistor defects; Figure 5.8 shows the yield with clustered broken NWs. We assume \( a = 0.2 \) and \( n =2 \) in the simulations. The results indicate that our fault tolerance techniques also work for clustered defects/faults: the yield of WISP-0 with \( EC3/2\text{-way} \) remains at around 20%
even when the cluster rate of transistors is 5% for the parameters simulated. Note that each defect cluster may have multiple defects.

As with uniform defects, H2L technique improves the yield of WISP-0 significantly and the effectiveness of nanoscale voting is much better. For example, if the cluster defect rate for transistors is higher than 4%, nanoscale TMR in pure AND-OR fabric is actually deteriorating the overall yield. For clustered broken NWs, nanoscale TMR does not appear to work at all. However, with H2L technique, nanoscale 6MR consistently improves the yield for clustered defective transistors. Even for clustered broken NWs, the 6MR technique still improves the yield of WISP-0 when the cluster defect rate is below 4%.

**Figure 5.7:** The yield achieved for WISP-0 for various defect rates assuming clustered defective transistors.
Figure 5.8: The yield achieved for WISP-0 for various defect rates when considering clustered broken NWs.

The yield-density product of WISP-0 for clustered defects is shown in Figure 5.9 and Figure 5.10. We assume the same parameters for clustered defect model (i.e., \( a = 0.2 \) and \( n = 2 \)) in the simulations. Similar to uniform defects, the best yield-density product is achieved with H2L WISP-0 with 2-way redundancy.

Figure 5.9: The yield-density product achieved for WISP-0 when only considering stuck-on FETs.
5.5.5 Impact of Transient Errors

We extended the yield simulator to provide an initial analysis on the benefits of the built-in fault tolerance techniques for transient errors. This is shown in Figure 5.11. The results indicate that we could tolerate transient faults fairly well although the masking is less effective than for permanent defects. On the other hand, we expect these errors to be much less frequent than those caused by permanent defects. One insight is that the CMOS 6MR appears to have the best overall benefit for these types of errors. The reason is that as these errors are random and transient, if an error does not occur at the same time and same position across at least 4 copies (out of 6), the system-level TMR voting could mask it – assuming that other errors are correctly masked.
5.6 Chapter Summary

In this chapter, we used WISP-0 as a representative design to evaluate the density, speed, power consumption and yield of NASIC designs and compared with the equivalent CMOS version. The results show that NASIC WISP-0 is significantly denser than the advanced CMOS version: WISP-0 without built-in redundancy but with H2L technique will be 39X denser than the equivalent design with 18nm CMOS technology. After applying various built-in fault tolerance techniques, WISP-0 is still 3~4X denser. NASIC designs are also promising in speed and power consumption. The estimations indicate that WISP-0 is at least comparable to the CMOS version in speed and power consumption, if not better. The yield simulations show that our hierarchical fault tolerance can effectively mask faults caused by permanent defects, transient faults from external noises or internal crosstalk.
CHAPTER 6

6 CONCLUSION

This PhD dissertation investigated new NW FET-based circuits and architectures for high density/low power computational systems as alternatives to end-of-the-roadmap CMOS. It proposed a new nanoscale computational fabric, NASIC, that is based on semiconductor nanowires and targeting microprocessor designs; it evaluated key metrics for NASIC designs.

NASIC designs use nanoscale field effect transistors (FETs) on 2-D semiconductor nanowires to implement logic functions, with some supporting CMOS circuitry for control. NASIC designs are optimized towards specific applications for higher density and functionalized during manufacturing. The fabric is well aligned with the expected manufacturing constraints; its projected benefits scale with improvements in nano-manufacturing techniques.

A novel dynamic circuit design enables high-density digital logic, sequential circuits and efficient pipeline structures without explicit latching. Circuit-level simulations validate the concept of the new dynamic circuits. We also analyze key aspects of these circuits such as signal degradation, charge sharing, capacitive coupling, and sensitivity to key parameters. For example, we found that the NASIC dynamic circuits work in a wide range of threshold voltages.

With small modifications on the way of controlling nanowires, we can combine two logic families into the same NASIC fabric. This Heterogeneous 2-Level (H2L) technique significantly improves area efficiency of NASIC designs compared to pure AND-OR NASIC designs without additional manufacturing requirements. The density
improvement of H2L WISP-0 without redundancy is 41% for 18nm CMOS process over the simple AND-OR NASIC WISP-0. After applying 2-way redundancy, the density improvement of H2L logic technique is 52%.

Various built-in fault-tolerance techniques can be applied to sustain the functionality of NASIC designs in the presence of defects/faults. These techniques include n-way redundancy, code-based fault-masking circuitry and system-level CMOS or nanoscale voting. Our hierarchical built-in fault-tolerance techniques can be applied to NASIC designs at various granularities, such as transistor, circuit and architectural levels, while carefully managing area-yield tradeoffs.

Our built-in fault-tolerance techniques can improve the yields of NASIC designs significantly. Without redundancy, the yield of WISP-0 is 0, even for a very low defect rate. With our combined fault-tolerance techniques, however, the yield is much improved. For example, if CMOS voting is possible, the yield of WISP-0 is up to 50% even if 15% transistors are defective. 2-way redundancy, code-based fault-masking technique can also significantly improve the yield of WISP-0. The yield of WISP-0 with a defect rate of 5% is as high as 78% for H2L with nanoscale voting and 2-way redundancy. This is the first work that shows nanoscale voting in this defect rate range.

Evaluations show great density advantage of the NASIC fabric over CMOS: a NASIC-based microprocessor design with no redundancy is 39X denser than its equivalent 18nm CMOS implementation. After applying 2-way redundancy and nanoscale voting, the density remains 4X higher than in 18nm CMOS.

Preliminary estimations on the speed and power consumption of NASIC designs are presented in the dissertation. The results show that WISP-0 might be able to run at
about 57GHz and its power consumption is at least comparable to conventional 18nm CMOS technologies. Exact comparison of CMOS speeds and power consumption are ongoing in the NASIC group. Similarly, 3D NW device models will be utilized for a more accurate estimation of NASIC performance and power consumption.

This dissertation shows that with appropriate fabric, circuit and architectural design, as well as built-in fault-tolerance techniques, it is possible to implement high-density and reliable computations within unconventional manufacturing/design constraints, even if the underlying devices are unreliable. We believe that NASIC is one of the most promising nanofabrics and hope this dissertation will encourage more interest in this direction and make nanoscale computing realizable in the near future.


