Circuits, Architectures, and CAD for Low-Power FPGAs

by

Safeen Huda

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright 2017 by Safeen Huda
Abstract

Circuits, Architectures, and CAD for Low-Power FPGAs

Safeen Huda
Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto
2017

Field Programmable Gate Arrays (FPGAs) are becoming an ever more prominent platform for the implementation of digital systems. In contrast to traditional processors that map computations onto a series of predefined instructions, an FPGA maps a computation onto a custom tailored digital circuit, thus yielding performance benefits. FPGAs can also provide cost and time-to-market benefits compared to Application Specific Integrated Circuits (ASICs). However, these advantages do not come for free as FPGAs typically have higher power consumption than the aforementioned platforms. FPGA power consumption is specifically of great concern presently given the growing demand for compute power in power constrained environments such as in mobile devices and data centres. To that end, in this thesis we explore circuit and architectural techniques to reduce power dissipated in FPGAs, and present the necessary CAD tools to implement our proposed power optimization schemes.

First, we present a technique to reduce the dynamic power dissipation that arises because of glitches. For this technique we designed a programmable glitch filter which we propose to augment a conventional FPGA; the proposed circuit can be configured to eliminate glitches for a range of different pulse-widths. Our second power reduction technique is motivated by observing that there exists an abundance of unused routing conductors in FPGAs. We show that by tri-stating these unused routing conductors we can reduce the effective wire capacitance seen by adjacent conductors, thereby reducing dynamic power. Moreover, we show how tri-stated wires also results in reduced routing leakage. The third power reduction technique also seeks to leverage unused routing conductors; in this case we employ the unused routing conductors as charge reservoirs to enable a charge recycling technique to reduce dynamic power. Finally, this thesis presents a technique to aggressively reduce leakage power dissipated in the routing network of an FPGA by tri-stating both used and unused routing buffers; the used routing buffers in this case are temporarily activated when they are needed i.e. when they are in the process of transmitting data. Ensuring that all circuits are effectively shut-off at all other times allows us to reduce leakage power.
To my family.
Acknowledgements

My time in grad school has been a truly rewarding experience and as it comes to a close, I wish to acknowledge the help and support that I’ve received along the way.

I would first like to thank my supervisor, Prof. Jason Anderson for his guidance and unyielding encouragement over the many years that I’ve known him. He’s a widely acknowledged master of his field (although he is too humble to admit it) and I’ve been fortunate to learn so much from him about a wide range of different topics, including how to effectively carve out a manageable research project from an initial (and at times obscure) idea. He has always been a fantastic role model and a great friend; I’m truly grateful for being able to pursue my PhD under his supervision.

I would also like to thank the other members of my thesis committee: Prof. Vaughn Betz, Prof. Paul Chow, Prof. Jonathan Rose, and my external examiner, Dr. Jonathan Greene for their detailed review of my thesis and for their very helpful comments and suggestions. I’m not exaggerating when I say that the members of my thesis committee are luminaries in the FPGA world and all of them have had significant pioneering contributions to this field; I’m humbled to have had the honor to present my work to such an esteemed group of researchers. The feedback I received from them has greatly improved the quality of this thesis.

Part of the research conducted as part of my thesis was in collaboration with Fujitsu Laboratories, and I would like to extend my gratitude to Fujitsu for supporting my research. I would like to specifically thank Dr. Hirotaka Tamura-san for his guidance and insightful suggestions. I would also like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the University of Toronto for financial support I’ve received over the course of my degree.

A highlight of grad school has been the many friendships that have formed over the course of my studies. I would like to acknowledge: Abdullah Al Rashid, Xander Chin, Jongsok (James) Choi, Yasser Hassan, Sadegh Jalali, Waqas Khan, Saad Mahmood, Mario Milicevic, Vincent Mirian, Nauman Mufti, Ravi Shivnaraine, Andrew Shorten, Ruediger Willenberg, and Rizwan Zaeem. It has been a pleasure getting to know them over the past few years, they’ve definitely made grad school a lot more enjoyable. I would also like to acknowledge the past and present members of Jason’s research group; I’ve learned a lot about FPGA CAD and architecture through our various discussions/debates and I’m grateful for what they’ve all taught me.

Finally, I would like to thank my family, whose incessant support was key to the successful completion of my degree. I am grateful to my wife, Shamayel Ahmed, for her love, advice, and patience; she has been one of my pillars of strength over these past few years. I would like to thank my in-laws, Mr. and Mrs. Masud Ahmed as well as Alavi and Nafe (my sister-in-law and brother-in-law, respectively) for welcoming me into their family. I’ve been greatly influenced by many members of my extended family as well, but I’d likely to specifically acknowledge two key role models: Dr. Sajeedul Huq and Prof. Khaliqur Rahman, both of whom have been sources of inspiration and guidance over the course of my educational journey from high school through to the completion of my PhD. Last but not least, I would like to thank my parents, Shahedul and Rafaat Huda, for all that they have given me throughout my life. It’s difficult for me to describe how grateful I am to them: for their love, for their hard work and many sacrifices for the sake of my education, for their unwavering belief in me. All that I am today is because of them and, while I owe them a debt I will never be able to repay, I hope that they may feel recompensed by whatever success I achieve in life. I thank them for being such wonderful parents.
1 Introduction
  1.1 Contributions .............................................. 2
  1.2 Thesis Organization ..................................... 3

2 Background
  2.1 Introduction .............................................. 4
  2.2 FPGA Architecture and Circuit Structures .............. 4
  2.3 Power Consumption in FPGAs ............................ 6
    2.3.1 Dynamic Power ....................................... 6
    2.3.2 Static (Leakage) Power .............................. 8
    2.3.3 FPGA Power Breakdown ............................... 8
  2.4 Power and Clock Gating for FPGAs ...................... 12
    2.4.1 Static Power Gating ................................ 13
    2.4.2 Dynamic Power Gating .............................. 14
    2.4.3 Clock Gating ...................................... 16
  2.5 Multi $V_{DD}$ Architectures ............................ 17
    2.5.1 Configurable Dual-$V_{DD}$ Architecture ........ 18
    2.5.2 Fixed Dual-$V_{DD}$ Architecture ................. 20
  2.6 Power Reduction in FPGA Routing Structures .......... 22
    2.6.1 Leakage Power Reduction ............................ 22
    2.6.2 Leakage Minimization using SRAM Redundancy ...... 22
    2.6.3 Dual $V_T$ Routing Structures .................... 23
    2.6.4 Body Biasing and Gate Biasing ..................... 24
    2.6.5 Routing Switch with a Virtual $V_{DD}$ .......... 25
  2.7 Ultra-Low-Power FPGAs .................................. 27
  2.8 Examples from Industry .................................. 29
    2.8.1 Xilinx Low-Power Prototype ....................... 29
    2.8.2 Altera Programmable Power .................... 31
  2.9 Summary ................................................. 32
# Glitch Reduction

## 3.1 Introduction

## 3.2 Background

### 3.2.1 Previous Work on Glitch Power Reduction

### 3.2.2 The Limitations of Path-Delay Balancing

## 3.3 Proposed Glitch Filtering Circuitry

### 3.3.1 Circuit Overview

### 3.3.2 Alternative Glitch Filter Topology

### 3.3.3 Delay Line Design

### 3.3.4 PVT Sensitivity

### 3.3.5 Proposed Architecture

## 3.4 Glitch Filter Setting Optimization

### 3.4.1 CAD Flow

### 3.4.2 Glitch Power Optimization

## 3.5 Experimental Study

### 3.5.1 Maximum Power Reduction Assuming Ideal Delay Lines

### 3.5.2 Power Reduction with Real Delay Lines

### 3.5.3 Tool Run-time

## 3.6 Summary

---

# Leveraging Unused Resources for Energy Optimization of FPGA Interconnect

## 4.1 Introduction

## 4.2 Background

### 4.2.1 Interconnect Dynamic Power and Capacitance Scaling

### 4.2.2 Leakage Power in FPGA Interconnect

### 4.2.3 Wire Planning in ASICs and FPGAs

### 4.2.4 Routing Resource Utilization in FPGAs

## 4.3 Power Optimization Using Tri-State Buffers

### 4.3.1 Dynamic Power Optimization

### 4.3.2 Static Power Optimization

## 4.4 Tri-state Buffers

### 4.4.1 Buffer Comparison

## 4.5 Interconnect Modelling

## 4.6 Tool Support

### 4.6.1 Power Optimization Overview

### 4.6.2 Dynamic Power Optimization

### 4.6.3 Static Power Optimization

## 4.7 Experimental Study

### 4.7.1 Router Parameter Optimization

### 4.7.2 Area Overhead

### 4.7.3 Router Run-Time

## 4.8 Summary
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Charge Recycling for Power Reduction in FPGA Interconnect</td>
</tr>
<tr>
<td>5.1 Introduction</td>
</tr>
<tr>
<td>5.2 Background</td>
</tr>
<tr>
<td>5.2.1 Charge Recycling</td>
</tr>
<tr>
<td>5.3 CR-Capable FPGA Interconnect</td>
</tr>
<tr>
<td>5.4 Tool Support</td>
</tr>
<tr>
<td>5.4.1 Post-Routing CR Mode Selection</td>
</tr>
<tr>
<td>5.5 Experimental Study</td>
</tr>
<tr>
<td>5.5.1 Area Overhead</td>
</tr>
<tr>
<td>5.5.2 Results</td>
</tr>
<tr>
<td>5.6 Discussion</td>
</tr>
<tr>
<td>5.6.1 Clock Power Reduction Via Charge Recycling</td>
</tr>
<tr>
<td>5.6.2 BRAM Power Reduction Via Charge Recycling</td>
</tr>
<tr>
<td>5.7 Summary</td>
</tr>
<tr>
<td>6 Pulsed-Signalling</td>
</tr>
<tr>
<td>6.1 Introduction</td>
</tr>
<tr>
<td>6.2 Pulsed-Signalling for Leakage Power Reduction</td>
</tr>
<tr>
<td>6.2.1 Pulsed-Signalling Circuit</td>
</tr>
<tr>
<td>6.2.2 Noise Considerations</td>
</tr>
<tr>
<td>6.3 CAD Support</td>
</tr>
<tr>
<td>6.3.1 Power-Aware Router</td>
</tr>
<tr>
<td>6.3.2 Routing Wire Mode Selection</td>
</tr>
<tr>
<td>6.4 Experimental Study</td>
</tr>
<tr>
<td>6.4.1 Power Reduction</td>
</tr>
<tr>
<td>6.4.2 Area Overhead</td>
</tr>
<tr>
<td>6.4.3 Discussion</td>
</tr>
<tr>
<td>6.5 Summary</td>
</tr>
<tr>
<td>7 Conclusion</td>
</tr>
<tr>
<td>7.1 Summary and Contributions</td>
</tr>
<tr>
<td>7.2 Future Work</td>
</tr>
<tr>
<td>7.2.1 Glitch Reduction</td>
</tr>
<tr>
<td>7.2.2 Energy Optimization of Unused Routing Resources</td>
</tr>
<tr>
<td>7.2.3 Charge Recycling</td>
</tr>
<tr>
<td>7.2.4 Pulsed Signalling</td>
</tr>
<tr>
<td>7.2.5 Combining Power Reduction Techniques</td>
</tr>
<tr>
<td>7.3 Conclusion</td>
</tr>
</tbody>
</table>

**Bibliography**
List of Tables

3.1 UMass RCG HDL Benchmark set details ........................................... 54
3.2 Benchmark resource utilization and dynamic power statistics .................. 55
3.3 Detailed glitch and dynamic power reductions for glitch filters with three delay line variants. 57
3.4 Circuit-by-circuit run-time data for MILP solver ...................................... 60
3.5 Solver run-time when terminated early .................................................. 62

4.1 Comparison of area and leakage of buffer topologies .................................. 79
4.2 Resource statistics for VTR and MCNC circuits ......................................... 86
4.3 Detailed power reductions assuming $\rho_C = 2$ ........................................ 90
4.4 Router run-times for baseline router and power-aware router ....................... 94

5.1 Power reductions and area overhead for each benchmark in an architecture fully populated with CR-capable switches. TF refers to the timing relaxation factor (a value of 1 indicates timing constraints are not relaxed) ........................................ 124

6.1 Power reduction and area overhead results .......................................... 135
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example of a 3-input logic circuit implemented using a LUT.</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Top-level logic and routing architecture of FPGAs.</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Circuit structures present in routing network.</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Example showing conditions which result in glitches at the outputs of combinational gates.</td>
<td>8</td>
</tr>
<tr>
<td>2.5</td>
<td>Example showing inherent mismatch in arrival times due to structure of circuit.</td>
<td>8</td>
</tr>
<tr>
<td>2.6</td>
<td>Dynamic power breakdown for 4-LUT architectures.</td>
<td>9</td>
</tr>
<tr>
<td>2.7</td>
<td>Stratix-II (a 6-LUT architecture) dynamic power breakdown [93].</td>
<td>10</td>
</tr>
<tr>
<td>2.8</td>
<td>Estimated logic and interconnect dynamic power breakdown vs. LUT size. Plot taken from [68].</td>
<td>10</td>
</tr>
<tr>
<td>2.9</td>
<td>Static power breakdowns.</td>
<td>11</td>
</tr>
<tr>
<td>2.10</td>
<td>VTR estimated energy breakdown for a 6-LUT 40 nm FPGA [74].</td>
<td>11</td>
</tr>
<tr>
<td>2.11</td>
<td>A system containing modules whose outputs may temporarily be inconsequential.</td>
<td>12</td>
</tr>
<tr>
<td>2.12</td>
<td>Power gating</td>
<td>13</td>
</tr>
<tr>
<td>2.13</td>
<td>Tile architecture of the FPGA with dynamic power gating proposed in [20].</td>
<td>14</td>
</tr>
<tr>
<td>2.14</td>
<td>Block-level details of power gating circuitry.</td>
<td>16</td>
</tr>
<tr>
<td>2.15</td>
<td>Clock regions of varying granularity.</td>
<td>16</td>
</tr>
<tr>
<td>2.16</td>
<td>Region-level clock network architectures.</td>
<td>18</td>
</tr>
<tr>
<td>2.17</td>
<td>Configurable dual-$V_{DD}$ blocks.</td>
<td>19</td>
</tr>
<tr>
<td>2.18</td>
<td>Signal crossing voltage domains from $V_{DD1}$ to $V_{DD2}$.</td>
<td>19</td>
</tr>
<tr>
<td>2.19</td>
<td>Level converter schematic</td>
<td>20</td>
</tr>
<tr>
<td>2.20</td>
<td>Effect of level converter placement on power savings.</td>
<td>21</td>
</tr>
<tr>
<td>2.21</td>
<td>Fixed Dual-$V_{DD}$ architectures.</td>
<td>22</td>
</tr>
<tr>
<td>2.22</td>
<td>Leakage paths in routing circuits.</td>
<td>23</td>
</tr>
<tr>
<td>2.23</td>
<td>Routing multiplexer leakage reduction through SRAM redundancy.</td>
<td>24</td>
</tr>
<tr>
<td>2.24</td>
<td>Routing buffer with virtual $V_{DD}$.</td>
<td>26</td>
</tr>
<tr>
<td>2.25</td>
<td>Level restorer which is already present in conventional FPGA routing circuitry. Needed to convert the output of the preceding NMOS pass-gate multiplexers from $V_{DD} - V_T$ to $V_{DD}$.</td>
<td>27</td>
</tr>
<tr>
<td>2.26</td>
<td>Routing switch with both buffer and configuration memory connected to virtual supplies.</td>
<td>27</td>
</tr>
<tr>
<td>2.27</td>
<td>Signal degradation through a pass transistor multiplexer operating in Sub-$V_T$. The current through the off transistors, $M_3 - M_4$, diminishes the amplitude of the output current.</td>
<td>28</td>
</tr>
<tr>
<td>2.28</td>
<td>Power gating scheme with input isolation buffers and output keepers in the Xilinx prototype.</td>
<td>31</td>
</tr>
<tr>
<td>3.1</td>
<td>Glitch reduction through input delay balancing</td>
<td>34</td>
</tr>
</tbody>
</table>
3.2 Proposed glitch-free BLE from [58]. ................................................. 35
3.3 NMOS pass-gate circuit. ................................................................. 35
3.4 Temperature consequences on FPGA logic and routing delays. .......... 37
3.5 Proposed glitch-filter circuit. .......................................................... 39
3.6 Pulse-train at input aligned with glitch filter output leads to transmission of glitches from input to output without any attenuation. .................................................. 40
3.7 State-dependent asymmetric delay cell. ........................................... 41
3.8 Proposed glitch-filter circuit augmented with state-dependent asymmetric delay line. .................................................. 41
3.9 Pulse-width transfer characteristic of proposed glitch filter. ............... 42
3.10 Proposed alternative glitch filter topology. ...................................... 43
3.11 Pulse-width transfer characteristic of alternative glitch filter topology. .................................................. 44
3.12 Delay-cell implementation options. .................................................. 44
3.13 Proposed BLE architecture. ............................................................ 47
3.14 CAD/experimental flow for proposed glitch reduction technique. ......... 48
3.15 Behavioural model of glitch filter circuit. ......................................... 50
3.16 Glitch power reductions for delay lines with varying resolution and maximum delay. .................................................. 56
3.17 Max glitch power reduction vs. area overhead. ................................ 59
3.18 Incumbent solution normalized cost and estimated lower bound of optimal cost (also normalized) versus time for \( \text{apex2} \). ................. 61
3.19 Digital circuits may be partitioned to yield multiple independent combinational circuits whose corresponding MILPs may be solved independently. ......................... 63

4.1 Interconnect capacitance components. ............................................ 66
4.2 Leakage paths in a routing switch. ...................................................... 67
4.3 Interconnect utilization distribution for the VTR benchmark suite targeting an architecture with 1.3× minimum channel width. .......................... 69
4.4 Effective impedance of coupled routing conductors. ......................... 70
4.5 \( n^{th} \)-level neighbours of conductor \( i \). ............................................ 71
4.6 Capacitance reduction vs. \( \rho_C \) for different levels of modelling accuracy. .................................................. 72
4.7 Leakage through routing multiplexer when unselected input tri-stated. .......... 72
4.8 Conventional tri-state Buffers. .......................................................... 74
4.9 Proposed tri-state Buffers. ............................................................... 75
4.10 Leakage arising from using an HVT NMOS transistor in output stage. .......... 77
4.11 Relative buffer leakage versus boost factor (\( V_{DDB}/V_{DD} \)). .............. 79
4.12 Assumed layout of routing conductors within a channel. .................... 80
4.13 Detailed view of routing conductor layout. ........................................ 81
4.14 Impact of \( DP \) and \( SP \) on interconnect dynamic power reduction and critical path. .................................................. 88
4.15 Impact of \( SP \) and \( DP \) on active routing leakage reduction. ............... 89
4.16 Interconnect dynamic power reduction vs. \( \rho_C \). ................................ 92

5.1 Charge recovery and recycling in a CMOS circuit with an auxiliary reservoir capacitor. .................................................. 99
5.2 Charge recovery/recycling on transitions: output waveform during a rising transition (shown on left); output waveform during a falling transition (shown on right). ......................... 99
5.3 Theoretical power savings for different charge recycling scenarios. .......... 102
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>CR buffer circuit.</td>
</tr>
<tr>
<td>5.5</td>
<td>Circuit implementation of Delay Line in CR buffer.</td>
</tr>
<tr>
<td>5.6</td>
<td>CR Buffer with <em>Gating Circuitry</em> exposed.</td>
</tr>
<tr>
<td>5.7</td>
<td>Charge recycling (CR) circuit.</td>
</tr>
<tr>
<td>5.8</td>
<td>Simulated waveforms of switch output and reservoir nodes.</td>
</tr>
<tr>
<td>5.9</td>
<td>Abstracted representation of rise and fall transitions of CR buffers.</td>
</tr>
<tr>
<td>5.10</td>
<td>Low-power Schmitt trigger.</td>
</tr>
<tr>
<td>5.11</td>
<td>Estimated dynamic power savings versus PF.</td>
</tr>
<tr>
<td>5.12</td>
<td>Estimated critical path impact versus PF.</td>
</tr>
<tr>
<td>5.13</td>
<td>Experimental flow.</td>
</tr>
<tr>
<td>5.14</td>
<td>Area overhead and interconnect dynamic power savings vs. percentage of CR-capable switches.</td>
</tr>
<tr>
<td>5.15</td>
<td>Interconnect dynamic power reduction vs. critical path relaxation factor. Shown for an architecture fully populated with CR capable switches.</td>
</tr>
<tr>
<td>5.16</td>
<td>Proposed enhanced CR buffer which has access to a global reservoir.</td>
</tr>
<tr>
<td>5.17</td>
<td>Charge recycling in the CLB-level clock network.</td>
</tr>
<tr>
<td>5.18</td>
<td>Charge recycling-enabled BL Drivers in BRAMs.</td>
</tr>
<tr>
<td>5.19</td>
<td>Timing diagram of signals needed to facilitate charge recycling.</td>
</tr>
<tr>
<td>6.1</td>
<td>Pulsed-signalling buffer.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The last three decades have witnessed considerable growth in the use of Field Programmable Gate Arrays (FPGAs) across a wide range of applications. At their onset, FPGAs were primarily used for hardware emulation and to build “glue logic” – i.e., simple logic structures that could be used as an interface between more complex digital systems such as microprocessors. As the capability of FPGAs progressed and the economics pertaining to semiconductor development transformed over time, FPGAs became an increasingly attractive platform for the implementation of complex digital systems – a true alternative to Application Specific Integrated Circuits (ASICs), at least in low to medium volume markets. At the present time, the relentless technological evolution of FPGAs over 30 years has allowed them to gain traction in the high-performance computing space, where they are used alongside traditional processors to implement accelerators that provide improved computational throughput and energy efficiency. The $16.7B acquisition of Altera (a leading FPGA vendor) by Intel in 2015 and the use of FPGAs by Microsoft [82], Baidu [2] and Amazon [1] are strong indications that FPGAs are poised to play a key role in future datacenters to enable acceleration of applications ranging from machine learning to web search.

At the other end of the computing spectrum, there are also indications of the advancement of FPGAs in the low-power/mobile/IoT space. In fact one FPGA vendor, Lattice Semiconductor, has a dedicated product line (their iCE family) that targets these specific markets; Lattice’s FPGAs have recently been employed in Samsung’s Galaxy smart phones [50] as well as Apple’s latest iPhone 7 [98]. Moreover, FPGAs have been proposed as an attractive alternative to ASICs in ultra low power (ULP) applications due to the extreme cost constraints in this particular domain [22]. Together, these developments herald a new era for FPGAs as they are able to participate in two of the most lucrative domains in modern computing.

These are encouraging signs for the FPGA industry; however these particular application domains place great emphasis on energy efficiency. While the programmability of FPGAs allows them to gain several significant advantages, such as shorter time to market, lower per-unit cost (versus ASICs) in moderate volumes, reconfigurability thus easing testing and debug, and enabling the use of FPGAs in custom-computing applications. The programmability does not come for free. It is associated with overhead such that an FPGA implementation of a given circuit consumes more power, area, and is slower than a custom ASIC implementation. In particular, a previous study has shown that the power consumption of a design targeted to an FPGA is 7-14× higher than an equivalent ASIC implementation [57]. We are
Chapter 1. Introduction

therefore compelled to improve the energy efficiency of FPGAs to facilitate the increased proliferation of FPGAs in these markets; indeed improving energy efficiency is one of the key ingredients to the growth of the FPGA industry.

1.1 Contributions

In this thesis we present circuits, architectures, and the associated CAD support to implement different power reduction techniques for FPGAs. We present four different techniques to reduce power:

1. Chapter 3 presents a technique to reduce glitch power in FPGAs. Glitches are unnecessary transitions on logic signals that needlessly consume dynamic power. Glitches arise from imbalances in the combinational path delays to a signal, which may cause the signal to toggle multiple times in a given clock cycle before settling to its final value. We propose a low-cost circuit structure that is able to eliminate a majority of glitches. The structure, which is incorporated into the output buffers of FPGA logic elements, suppresses pulses on buffer outputs whose duration is shorter than a configurable time window (set at the time of FPGA configuration). Glitches are thereby eliminated “at the source” ensuring they do not propagate into the high-capacitance FPGA interconnect, saving power. An experimental study, using a custom CAD flow to optimize glitch reduction subject to timing constraints coupled with commercial tools for power analysis (from Altera), demonstrates that the proposed technique reduces \(\sim 70\%\) of glitches, at a cost of \(\sim 1\%\) reduction in speed performance. This work was published in [46].

2. Chapter 4 presents a technique to leverage the excess routing conductors present in conventional FPGA architectures to reduce both dynamic and static power. To reduce dynamic power, we propose to ensure that used routing conductors are adjacent to unused routing conductors that are left floating. In doing so, the effective coupling capacitance between the used routing conductor and the unused routing conductor is reduced because the original coupling capacitance is now placed \(\textit{in series}\) with other capacitances in the circuit. To reduce static power, we observe that leakage in routing multiplexers is dominated by specific paths. In this chapter, we show that the static power of the multiplexer may be significantly reduced if certain conductors attached to the multiplexer inputs are left unused and tristated. To ensure unused conductors are allowed to float requires the use of tri-state routing buffers, and to that end, we also propose two low-cost tri-state buffer topologies with different power and area-overhead trade-offs. We also introduce CAD techniques to maximize the likelihood that the aforementioned power reduction techniques may be applied to the most power critical routing conductors in a given design. Results show that interconnect dynamic power reductions of up to \(\sim 25\%\), interconnect static power reductions of up to \(\sim 80.9\%\), and overall interconnect energy reductions ranging between \(\sim 23-33\%\) are expected, with a critical path degradation of \(<1.8\%\), and a total area overhead of between \(\sim 2.6-4.1\%\), depending on the tri-state buffer topology employed. This work has been presented in [43] and another publication [44] is currently under review.

3. Chapter 5 presents another technique that leverages the unused routing conductors in an FPGA to reduce interconnect dynamic power. In this case we propose employing unused routing conductors as \textit{charge reservoirs} to enable a charge recycling power saving technique. Charge recycling via the unused conductors reduces the amount of charge drawn from the supply, which lowers energy consumption. To implement the proposed technique, we augment a conventional routing switch
with circuits that facilitate the various phases necessary to recycle charge and describe the CAD tool changes needed to support charge recycling at the routing and post-routing stages of the flow. Results show that dynamic power in the FPGA interconnect can be reduced by 15-18.4% depending on performance constraints. This work has been presented in [42].

4. Chapter 6 presents another technique to reduce leakage in the routing network of FPGAs. This chapter aims to extend the static power reduction techniques presented in Chapter 4. In addition to tri-stating unused routing conductors, we propose to dynamic tri-state used routing conductors whenever they are in a quiescent state (i.e. not active in propagating a signal transition). We call this proposed power reduction technique pulsed-signalling. As with the technique presented in Chapter 5, we also present the additional circuitry that is required to facilitate the desired dynamic gating behaviour. However, routing buffers that are tri-stated are susceptible to noise-induced bit errors. Consequently, only active routing conductors that can be safely guarded from noise sources may be permitted to operate in a dynamically tri-stated mode. As such we present CAD techniques that maximize the likelihood that used conductors may operate in a dynamic tri-stated mode to reduce power consumption. Experiments show that interconnect static power may be reduced by \(\sim 40\%\), depending on channel width assumptions. This work has been presented in [45].

1.2 Thesis Organization

The remainder of this dissertation is organized as follows: Chapter 2 provides background material presented in subsequent chapters. The primary contributions of this thesis are presented in Chapters 3, 4, 5 and 6. Finally, Chapter 7 concludes the dissertation.
Chapter 2

Background

2.1 Introduction

This chapter presents background material for the research presented in this thesis. We begin with a brief overview of the circuits and architectures employed in commercial FPGAs. Next, we discuss the principal components of the power dissipated in an FPGA, and highlight critical areas in which power may be optimized. Finally, the remainder of this thesis provides an extensive survey of past research and industrial efforts on power reduction strategies for FPGAs.

2.2 FPGA Architecture and Circuit Structures

At its essence, an FPGA is an array of programmable logic blocks, commonly referred to as Basic Logic Elements (BLEs), which can be programmed to implement different logic functions (ideally any arbitrary logic function). These BLEs are interconnected by a network of wires and programmable switches – this is referred to as the FPGA’s programmable routing network. Together, the array of BLEs and the programmable routing network allow the FPGA to implement (ideally) any arbitrary digital system.

Fig. 2.1(a) shows a $k$-input look-up table (LUT) followed by a by-passable flip-flop, which is the most common example of a BLE. Fig. 2.1(b) shows how LUTs are typically implemented in FPGAs using multiplexer and SRAM cells. Fig. 2.1(c) shows a transistor-level implementation of a 3-input LUT.
using multiplexers; the $k$-select lines of a multiplexer are driven by the inputs to the LUT, while the $2^k$ multiplexer inputs are connected to SRAM cells. By configuring the state of the $2^k$ SRAM cells, any $k$-input logic function may be realized; an example of a 3-input logic function being implemented in a 3-input LUT is shown in Fig. 2.1. Finally, the LUT multiplexers are typically implemented with pass-transistors trees, which only employ NMOS transistors to save area, as shown in Fig. 2.1(c).

Figure 2.1: Example of a 3-input logic circuit implemented using a LUT.

In modern FPGA architectures, a small number of BLEs are first combined to form a configurable logic block (CLB). Each CLB has an internal routing network which interconnects the BLEs contained within the block, as shown in Fig. 2.2(a). CLBs are arranged in an array and interconnected by a top-level routing network forms an FPGA. Fig. 2.2(b) shows an abstract view of a typical island-style FPGA, where a mesh-like routing network interconnects a two-dimensional array of CLBs. While many FPGA architectures have previously been explored, the island-style architecture has been the dominant architecture used/studied by vendors and researchers for almost two decades [19], and as such this architecture will be the prime focus of this thesis.

The mesh-like routing network of an island-style FPGA consists of horizontal and vertical wires – also
called tracks – which reside within routing channels; these wires connect to CLBs through connection boxes (CBs), and to other wires through switch boxes (SBs), as shown in the figure. SBs and CBs are implemented as pass-transistor-based multiplexers employing only NMOS transistors to minimize area, with an output driver circuit to buffer and recondition the output signal. Circuit structures for such a routing multiplexer are shown in Fig. 2.3; note that a feedback PMOS transistor is needed because the pass-transistor multiplexer is poor in transmitting a logic-“1”.

2.3 Power Consumption in FPGAs

2.3.1 Dynamic Power

The dynamic power dissipated in digital CMOS circuits arises from signal transitions. Whenever a signal toggles from logic-0 to logic-1, electric charge, and thus energy, is drawn from the power supply. The average dynamic power is given by the following equation:

\[ P_{avg} = \frac{1}{2} \sum_{i \in \text{signals}} f_i \cdot C_i \cdot V_{DD}^2 \]  

(2.1)
where $f_i$ is the average toggle rate of signal $i$ (in toggles/second); $C_i$ is the total capacitance of the conductors used to route signal $i$ and $V_{DD}$ is the supply voltage. The equation describes the average power drawn as charge is transferred from the power supply to a CMOS circuit’s output node. However, whenever a signal transitions from logic-0 to logic-1, a short circuit path is temporarily created between $V_{DD}$ and GND, which temporarily allows current to flow directly. This temporary short circuit therefore is also a component of the dynamic power consumed in a digital CMOS circuit, however through careful design, such short circuits can be made to last for a very short period of time or can be eliminated all together. As such, we ignore this component of dynamic power for the remainder of this thesis.

Equation 2.1 shows that dynamic power dissipated by a signal is proportional to its toggle rate. The toggles of a signal consist of both “useful” and “useless” signal transitions; these useless transitions are called glitches. Glitches commonly occur in digital circuits, and are a consequence of situations similar to that depicted in Figure 2.4. In the figure, we have an XOR gate with inputs $A$ and $B$, and output $OUT$.

Assume that $A$ and $B$ have the same state at the beginning of the cycle, and both signals make a single state transition during the cycle. As such, $OUT$ is logic “0” at both the beginning and end of the cycle. However, due to differences in the delays for the circuits driving inputs $A$ and $B$, $B$ makes its state transition before $A$, thus $A$ and $B$ are momentarily at different logic states. Thus, the output $OUT$ momentarily transitions to logic “1”. After a delay of $t_D$ seconds, $A$ makes its state transition, and the output $OUT$ now transitions back from logic “1” to “0”. The spurious transition $OUT$ momentarily makes when $A$ and $B$ are different values is known as a glitch. In general, glitches arise when the inputs to a combinational logic circuit momentarily assume a combination of values which causes unnecessary transitions at the output. These transitions have no functional value, and are a waste of energy. Each pair of transition consumes $CV^2_{DD}$ joules of energy, where $C$ is the capacitance of the net being driven by $OUT$.

Glitches have proven to be especially troublesome in FPGAs; whereas in an ASIC there exists freedom to minimize the disparity between the delays of different paths (and thus ensure that the arrival times of the input signals to a combinational circuit are well matched), there is no such freedom with FPGAs. For example, consider the circuit shown in Figure 2.5 which shows an inherent potential mismatch between the delays of the two paths of combinational logic, named path 1 and path 2, which converge at the XOR-gate. The delay mismatch is inherent because of the structure of this circuit (path 1 has more logic gates than path 2). If this circuit were to be implemented in an ASIC, given sufficient freedom in the ability to trade-off power with area or speed, the delay of the single logic gate in path 1 could be increased (for example, by reducing the sizing of its transistors), or the delays of the gates along path 2 could be decreased (by increasing the sizing of their transistors), all in a bid to equalize and align the arrival times at the inputs to the XOR-gate. In contrast, if this circuit were to be implemented in an FPGA, each of the logic gates comprising this net would be mapped to prefabricated circuits, whose delays cannot be optimized as freely, thus making the problem of equalizing arrival times difficult. As such, combinational logic circuits in FPGAs are prone to having a large difference in the arrival times of different input signals, and thus are prone to glitches. Recent work has shown that glitches account for, on average, 26% of total core dynamic power for a set of benchmark circuits, while glitches account for nearly 50% of total dynamic power for certain specific circuits in the benchmark set [88].
2.3.2 Static (Leakage) Power

While dynamic power arises when a digital circuit performs computations and does “work”, there is a constant power consumption that arises due to a CMOS transistor’s various leakage paths. This power is an ever-present component of a circuit’s power consumption and is known as static power, and exists even when the circuit is in an inactive quiescent state.

While there are several different current paths in an off MOS transistor, a significant path is from the drain-to-source terminal, referred to as subthreshold leakage current. For a precise analysis of subthreshold leakage, the interested reader is referred to [52]. For the purposes of this thesis, it is enough to highlight several trends pertaining to how subthreshold leakage varies with other criteria:

- Exponentially proportional to temperature.
- Linearly proportional to transistor width/length \((W/L)\).
- Exponentially proportional to \(V_{GS} - V_t\).

The first trend implies that subthreshold leakage increases as a chip heats up, creating a positive feedback loop. The second trend implies that wider transistors, while useful for speed, are more leakage-prone. The third trend hints at ways to reduce subthreshold current, namely, by reducing \(V_{GS}\) or raising \(V_t\).

It should also be noted that, under a given timing constraint, energy is minimized when \(V_{DD}\) and \(V_t\) are set such that the leakage power is approximately 30% of the total power dissipation [80]. Therefore, leakage power is quite naturally a major component of the total power dissipated in an FPGA.

2.3.3 FPGA Power Breakdown

Having discussed power consumption in digital circuits in a general sense, we now consider the specific sources of power consumption in FPGAs. Several prior works have characterized dynamic power in
FPGAs and have shed light on the contributions of each component of an FPGA to total dynamic power consumption. In [87], the authors studied the power breakdown in Xilinx’s Virtex-II device (Fig. 2.9(a)) showing that the bulk of the dynamic power dissipated in an FPGA is in its interconnect network, and this is corroborated by the breakdown reported for a Xilinx Spartan-3 device in [95] (Fig. 2.9(b)). The trends shown by these studies may be attributed to the fact that wiring segments in FPGAs are highly capacitive, primarily because of the area overhead required for programmability and that, when optimizing for both area and delay, FPGA architectures require relatively long wires in their interconnect [19].

![Interconnect breakdown](image1.png)

(a) Virtex-II dynamic power breakdown [87].

![Interconnect breakdown](image2.png)

(b) Spartan-3 dynamic power breakdown [95].

Figure 2.6: Dynamic power breakdown for 4-LUT architectures.

While the authors of [87] and [95] focused specifically on the power dissipated by programmable routing, logic, and input/output circuits (i.e. the circuits comprising an FPGA’s fabric), the power dissipated by other circuits comprising the core of modern FPGAs such as Block RAM (BRAM) and hardened arithmetic blocks need to be considered as well. In [93], the authors provide a dynamic power breakdown for an Altera Stratix-II device which includes power dissipated by BRAM in addition to programmable logic and routing circuits (Fig. 2.7). While the impact of BRAM power to total power dissipation is heavily design dependent (some designs may not use any BRAMs at all while others may use them very liberally), the authors nonetheless show that, on average, BRAMs contribute 18% of the total dynamic power dissipated in an FPGA. Interestingly, these results also show that the interconnect contributes 27% of total dynamic power; ignoring BRAM power consumption, we may estimate from the data that ~33% of the total dynamic power dissipated in the core fabric circuits is dissipated in the interconnect, ~51% is dissipated in logic circuits, while the remaining ~16% is dissipated by miscellaneous sources, which is in stark contrast to the results shown by [87] and [95]. The difference in these results is likely because Virtex-II and Spartan-3 are 4-LUT architectures [106, 107] while Stratix-II is a 6-LUT architecture [64]. A 6-LUT – which is favoured for high performance applications – will naturally consume more dynamic power than a 4-LUT because of the larger number of switched nodes (a 6-LUT will require a deeper pass-transistor tree than a 4-LUT) while also reducing the number of routing wires required to route a given design [6]; the use of 6-LUTs therefore shifts power dissipation from the interconnect to logic resources, a trend which has been studied in [68] and is shown in Fig. 2.8.
Modern FPGAs offered by the two largest vendors, Xilinx and Intel/Altera, are based on 6-LUT architectures, as such we can expect the power dissipation profile of the latest generation devices from these vendors to be in line with Fig. 2.7. However, while Xilinx and Intel/Altera primarily target markets which demand the highest levels of performance, other vendors such as Microsemi and Lattice have targeted growing markets which require lower performance but demand very high levels of energy/area efficiency. These vendors have in turn optimized the architectures of their FPGAs for these markets and have employed 4-LUTs in their latest products [60, 78], likely because 4-LUT architectures offer superior energy [68] and area [6] efficiency. For these same reasons, architectures employing 4-LUTs also seem to be favoured in the nascent embedded FPGA space by vendors such as Achronix [5], Efinix [32] and Flexlogix [33]. Consequently, the dynamic power breakdown of a modern FPGA is dependent on the underlying logic architecture, and is likely represented either by breakdowns show in Fig. 2.6 (for architectures with relatively small LUTs) or Fig. 2.7 (for architectures with large LUTs).

Fig. 2.9 shows the static power breakdowns of two 90 nm Xilinx FPGAs. Observe that most of the
leakage is within the SRAM configuration cells and the interconnect. However, the authors of [94] point out that SRAM leakage was not a target in the optimization of the FPGA and that it can be reduced by using long-channel or high-$V_t$ transistors in the SRAM cells. Such an approach would make the SRAMs slower, though this is not a significant issue as the configuration cells in FPGAs are generally only programmed once during device configuration.

Finally, Fig. 2.10 shows the energy dissipation breakdown estimated by the VTR 7.0 CAD flow when targeting a 40 nm, 6-LUT FPGA [74]. While comparisons between this plot and the previous plots shown in this section is difficult since overall energy dissipation is shown (instead of individual static and dynamic power breakdowns), it is noteworthy that interconnect is again estimated to dominate overall energy consumption.
2.4 Power and Clock Gating for FPGAs

As mentioned in the previous section, static power dissipation is a significant component of the total power dissipation of a chip, and is a function of the number of active circuits that are connected to the power supply. Since the circuits in an FPGA are prefabricated, when a design is implemented on an FPGA there may be many blocks which are unused, as the number of logic blocks or routing resources used by the design may be less than those provisioned by the FPGA. For example, one study has shown that 30-50% of routing resources in FPGAs are unused across a set of benchmark circuits [83]. However, since these unused blocks are still connected to the power supply, they needlessly dissipate static power. Static power gating has been proposed as a means to reduce the power consumption in such cases by disconnecting unused circuits from the power supply. We provide an overview of a static power gating technique proposed for FPGAs in Section 2.4.1.

![Figure 2.11: A system containing modules whose outputs may temporarily be inconsequential.](image)

Another situation which may arise – in any general digital system – is the one depicted in Figure 2.11. Here, we have two functional blocks $func_1$ and $func_2$ whose outputs are connected to a multiplexer, and the output of this multiplexer feeds the digital system encompassing these functional blocks. Note that when $ctl$ is a logic-0, the outputs of $func_2$ are inconsequential to the overall system, while when $ctl$ is a logic-1, the outputs of $func_1$ are inconsequential. Thus, based on the state of the system, one of the two functional blocks may needlessly be performing computations, and thus dissipating power. In fact, there are three components of the power which is needlessly being dissipated: the static power of the circuits contained within the functional block, the dynamic power of the block as it performs computations, and the dynamic power dissipated in the portion of the clock network exclusively serving...
the functional block. Another possible situation which may arise is that, based on the state of the system, one of the functional blocks may be completely idle, i.e., it does not perform any computations until the system assumes a certain state. In such a situation, the functional block needlessly dissipates static power and clock power (we assume that when a block is idle its internal nodes do not toggle and as such, the block does not dissipate any dynamic power). To optimize power dissipation in these cases, clock gating [108] and dynamic power gating [73] have been proposed. Dynamic power gating allows circuit blocks to be temporarily disconnected from the power supply while they are idle and/or their outputs are inconsequential to the system, while clock gating provides us with a means to disable parts of the clock tree which feed these blocks. Subsequent sections present approaches to dynamic power gating and clock gating in FPGAs.

2.4.1 Static Power Gating

Static power gating aims to reduce the static power of unused circuit blocks. Figure 2.12 shows how this can be accomplished: instead of connecting the blocks directly to $V_{DD}$ (the power rail), a PMOS transistor ($M_1$ in the figure) is placed in between the block and the rail. When the gate of the transistor is at logic-1, the transistor is off, disconnecting the block from the supply and reducing static power (the static power will substantially be reduced but not completely eliminated because $M_1$ will still leak when it is shut off). Conversely, when the gate is at logic-0 the transistor is on, thus connecting the block to the supply. Transistor $M_1$ is typically referred to as a sleep transistor. In the context of an FPGA, its gate would be connected to an SRAM configuration cell, programmed accordingly to turn circuit blocks on or off, based on their utilization.

![Power gating diagram](image)

Figure 2.12: Power gating

Static power gating in FPGAs has been explored in [35], where an architecture with gateable logic blocks (and not routing resources) was presented. In the study, the authors uniformly divide their FPGA into rectangular-shaped regions, where the power supply for each region is controlled by a single sleep transistor. The authors experiment with various region sizes – ranging from containing a single CLB to containing 64 CLBs – and analyze the area, power, and delay implications of the different levels of granularity. The authors were able to show static power reductions of $\sim 15\%$ for fine-grained regions, decreasing to $\sim 11.5\%$ for the most coarse-grained regions.

Sleep transistors must be sized in a manner such that they are able to supply the current demand of
the circuit(s) connected to the sleep transistor. The authors made the observation that the maximum current of a region scales sub-linearly with region size because the peak current of a large region is less than the sum of the peak currents of its smaller constituent regions. With the most fine-grained gating, an area overhead of 15% is expected, while the area overhead drops to 10% for the most coarse-grained gating considered in their study. Finally, the authors noted a performance penalty of 5-8% (based on the gating granularity), attributed to the delay impact of sleep transistors.

### 2.4.2 Dynamic Power Gating

An FPGA with support for dynamic power gating was presented in [20], and later extended to support power gating of switch blocks [21]. The work assumes the existence of a *power state controller* which would be implemented alongside a digital system on an FPGA. The power state controller is responsible for broadcasting power-gating control signals, used to dynamically power gate modules/functional blocks distributed throughout the chip. Given such a model, the authors propose to add sleep transistors for power gating (which would be done dynamically) with varying levels of granularity. The authors experimented with varying gating region size (where gating region refers to a region which is controlled by a single sleep transistor).

![Figure 2.13: Tile architecture of the FPGA with dynamic power gating proposed in [20]](image)

Figure 2.13 shows the proposed dynamic power architecture. In contrast to static power gating where a single SRAM cell controls the state of a sleep transistor, the control of sleep transistors in the proposed FPGA architecture is slightly more complex. In the proposed architecture, the sleep transistor can be controlled by one of three possible signals: a constant logic-0 if the circuits connected to the sleep transistor are to always be on and ungated, a constant logic-1 if these circuits are unused and we wish to statically power gate them, or the signal $PG\_CNTL_i$, which is an output of a power-state controller used to dynamically gate the circuit. Instead of routing the $PG\_CNTL_i$ signals separately, the authors propose using the existing routing network to route these signals from the power-state controller to the
sleep transistor; this can likely be done at low cost since routing resources in an FPGA are typically under utilized.

As with the work described in the above section, the FPGA is divided into regions of CLBs – all of the CLBs and connection blocks contained within each region are controlled by a single power transistor, although connection blocks at the perimeter of a region (i.e. those which connect to CLBs belonging to different power gating regions) have independent sleep transistors, as will be shown below. A single $PG_{CNTL}$ signal is thus used to control the state of all of the CLBs and connection blocks lying in the interior of a region; this signal must route to one of the channels along the perimeter of the region, from where it can connect to the sleep transistor, as depicted in the Figure 2.13. One point worth noting is that the chip is also divided into switch block regions, and these differ from the CLB regions in that each SB has its own sleep transistor. The authors took a very conservative approach to dealing with power gating of the switch blocks since they may carry power control signals (which must always be active) or signals belonging to different modules whose idle periods never coincide (and thus must always be on). However, the $PG_{CNTL}$ signal, which connects to the sleep transistors of the switch blocks within a region, is shared; this reduces the overhead of selecting multiple $PG_{CNTL}$ signals from the periphery of the region. Details of the circuitry controlling the sleep transistors of the CLBs, switch blocks, and connection blocks are shown in Figure 2.14.

The authors have shown that leakage can be reduced by about 45% using dynamic power gating for an architecture without a mechanism to power gate the switch blocks (this was the original architecture presented in [20]). The authors showed that a large region size (i.e., coarse-grained dynamic power gating) allowed for both a reduction in area overhead and an improvement in the amount of leakage power that could be reduced. The reduction in area overhead is expected since, as described in the previous section, the sleep transistor area scales sub-linearly with region size, while a coarse-grained architecture also allows for a reduction in the total number of SRAM cells (to set the state of the multiplexers driving the gates of the sleep transistors), $PG_{CNTL}$ signal routing multiplexers, and other additional circuits required for their power gating technique. The improvement in leakage current reduction is attributed to the fact that the proposed dynamic power gating architecture requires additional circuitry which adds to the total leakage of the FPGA, and these circuits are always on and thus always dissipating power. A larger region size means that the leakage power overhead introduced to the system is a smaller percent of the total leakage power. This, however, does not take into effect the fact that a smaller, finer-granularity region will be more likely to be gateable than a larger region. The authors have shown that the overall area overhead ranges between 2.5-4.5% as the granularity is varied from the most coarse to the most fine; this is considerably smaller than what was reported in [35], but this may be attributed to the difference in the two area models used in the two works. With the addition of gateable switch blocks, the power reduction increases to $\sim$92% (over a wide range of routing channel widths), at an area cost ranging from 6.7-7.7% (depending on the channel width).

Comparing dynamic power gating as a power reduction strategy for FPGAs over static power gating, we note that both techniques allow optimization of static power since we now have the freedom to disconnect unused circuitry from the power supply. However, while the power reduction afforded by static power gating is limited to the potential gains from unused circuitry, dynamic power gating allows leakage power savings of used circuits whenever they are idle. Both techniques have the same approximate area and performance cost (although dynamic power gating also requires an additional module – a power state controller – whose area estimates have not been taken into account by the authors of [20]).
Chapter 2. Background

2.4.3 Clock Gating

In previous generations of commercial FPGAs, the clock network could only be gated at the CLB level (i.e., the clock network within a CLB could dynamically be shut off) or at the root of the clock tree (meaning the entire clock network could be dynamically shut off) [11, 104], although more recent architectures now provide the ability to shut off the clock network at intermediate levels of the clock network tree [105]. The power savings achievable through clock gating of various clock network architectures was studied in [47]. As with various architectures relating to power gating discussed previously, the architecture of the clock network determines the likelihood of being able to dynamically shut off portions of the clock network, and thus save clock power. This work however did not consider the differences in area of the different clock network architectures.

The work considers an FPGA divided either into 4 regions (coarse) or 16 regions (fine) and, in each case, a skew-balanced H-tree delivers the clock signal to each region, as is shown in Figures 2.15(a) and

(a) Clock network with 4 regions. (b) Clock network with 16 regions.

Figure 2.15: Clock regions of varying granularity.
With a finely grained clock network, there is greater likelihood that an entire region may be clock gated than in coarse-grained network. Thus, a finely grained clock network may provide larger power savings. On the other hand, as shown in the figures, since the H-tree of a 16 region clock network has more levels of hierarchy than one with 4 clock regions, the finely grained clock network uses more metal and thus has a larger total capacitance. Assuming that each clock region has a structure similar to that depicted in Fig. 2.16(b), the amount of metal required to route a clock to a column spanning one of the FPGA’s quadrants is proportional to \( h + \frac{3}{4}v \), where \( h \) is the width of the chip and \( v \) is its height. In contrast, in a fine-grained clock network, the amount of metal required to route the clock for the same scenario is proportional to \( \frac{5}{2}h + v \), meaning that \( \frac{1}{4}(h + v) \) additional units of metal are required. This means that if there is very little clock gating, a coarse-grained clock network is likely to consume less power. Herein lies a trade-off – a coarse grained clock network is naturally less capacitive than a fine-grained clock network, but potentially less likely to allow entire regions of the chip to be clock gated.

The authors also studied the effects of different clock network architectures at the region level. Figures 2.16(a) and 2.16(b) show region-level clock networks with region-level gating and column-level gating, respectively. While topologically identical, the difference between the two architectures is that the column gating architecture allows the clock network to be gated either at the region or column level, while the region gating architecture only allows the clock to be gated at the region level. The authors thus study the 4 different permutations of number of regions and region-level clock network architecture, and assess the amount of power which can be reduced through clock gating for each architecture. The authors found that a coarse-grained architecture, on average, allowed for greater power reduction than a fine-grained architecture. This was despite the fine-grained architecture offering better chances of gating large portions of the clock network; the extra “overhead” resulting from the H-tree which distributes the clock to each region could not be compensated for, even under cases where there is substantial clock gating (i.e., large periods of time when clock signals are gated). The authors also found that a region-level clock network architecture which allowed column level gating was always superior to one with only region gating, although this was expected since a column level gating architecture provides increased flexibility with negligible power or area cost (even though the power and area cost for this architecture was not explicitly considered). Overall, the authors showed that under cases of significant clock gating, a coarse-grained clock network supporting column level gating allowed overall dynamic power reductions of \( \sim 8\% \).

### 2.5 Multi \( V_{DD} \) Architectures

In conventional synchronous digital systems, the overall performance of the system is dictated by the speed of the slowest path. Consider a digital system comprising three register-to-register paths, having register-to-register delays of 2 ns, 3 ns, and 4 ns, respectively, making the minimum clock period for this system 4 ns (if the \( t_{SU} \), or setup time, for the registers can be ignored). If the system operates at the minimum clock period, we observe that it is required that the delay along all of the paths is at most 4 ns, which means that the delays for path 1 and path 2 can be increased by 2 ns and 1 ns, respectively, without any consequence to overall system performance. In the parlance of static timing analysis, path 1 and path 2 are said to have, respectively, 2 ns and 1 ns of slack, while path 3 is said to be the critical path, as its delay cannot be increased without compromising system performance.

The freedom to increase the delay of paths with positive slack – the non-critical paths of a system
Figure 2.16: Region-level clock network architectures.

– presents a useful power optimization strategy as the available slack may now be traded-off for overall reduced power consumption; this technique is known as slack-fill \[22\]. While there are several ways in which delay and power of logic gates may be traded off, one popular technique which has been used to optimize power in ASICs involves the use of multiple $V_{DD}$’s \[97\]. With a linear delay model, the delay of a gate is approximately inversely proportional to $V_{DD}$, while dynamic and static power are quadratic and exponential functions of $V_{DD}$, respectively. Thus, relatively modest reductions in the $V_{DD}$ of a logic gate will result in proportionally modest increases in delay, but will yield substantial reductions in power. As such, in a digital system with multiple $V_{DD}$’s, logic gates on paths with sufficient slack can be connected to a lower voltage $V_{DD}$ rail, while gates on paths with little slack are connected to a higher voltage $V_{DD}$.

FPGAs in particular can benefit considerably from this technique; indeed any technique which allows timing slack to be traded off for power reductions could lead to considerable improvements in overall system level power consumption. In fact, both academic \[14\] and industrial \[10\] studies have shown that over many different benchmark circuits, FPGAs typically have a large number of paths with considerable slack. Therefore, there is much scope in FPGAs to utilize slack-filling techniques to reduce power. In the following sections, we review two approaches to incorporating multiple $V_{DD}$s in FPGAs.

### 2.5.1 Configurable Dual-$V_{DD}$ Architecture

A configurable dual-$V_{DD}$ FPGA architecture was proposed in \[34\]. The proposed architecture extends a conventional cluster-based island-style FPGA architecture and is shown in Figures 2.17(a) and 2.17(b).

As shown in the figures, the FPGA consists of conventional CLBs and routing switches, which can be connected to either one of two different supply voltages through different supply transistors. This allows CLBs or routing switches to either operate in a high-speed, high-power mode (when connected to the higher supply voltage), when these blocks are used by timing critical nets, or a low-speed, low-power mode (when connected to a lower supply voltage), which is suitable for nets with sufficient timing slack.
Chapter 2. Background

An additional block needed by this architecture – and indeed in any multi-\(V_{DD}\) architecture – is a level converter. Level converters are needed whenever nets cross from a low voltage domain to a high voltage domain, to prevent the situation depicted in Figure 2.18 from arising. In the figure, we have an inverter (on left) supplied by a voltage rail \(V_{DDL}\) driving the gate of a second inverter, which is connected to a second voltage rail, \(V_{DDH}\). The output of the first inverter is potentially problematic as it crosses between the two voltage domains, specifically when the output is at logic-1. In such a situation, the gate of the second inverter’s PMOS transistor (\(M_3\)) is at \(V_{DDL}\), while the source is at \(V_{DDH}\). Thus, the source-to-gate voltage of the PMOS device, \(V_{SG}\), is equal to \(V_{DDH} - V_{DDL}\). If \(V_{DDH} > V_{DDL}\), then \(V_{SG} > 0\) which can yield substantial increases in static power consumption even if \(V_{SG} < |V_{TP}|\), the threshold voltage of the PMOS device. On the other hand, if \(V_{DDH} < V_{DDL}\), then \(V_{SG} < 0\), and we do not encounter any problems related to increased static power consumption. A level shifter (converter) circuit such as the one shown in Figure 2.19 is required to convert a signal originating from a low \(V_{DD}\) domain to a higher \(V_{DD}\) to remedy this problem.

The authors of [34] experimented with where to place level shifters in their proposed architecture.

Figure 2.17: Configurable dual-\(V_{DD}\) blocks.

Figure 2.18: Signal crossing voltage domains from \(V_{DD1}\) to \(V_{DD2}\).
Figure 2.19: Level converter schematic.

Rather than distributing level shifters freely through the FPGA (which would be a costly proposition both from a power and area perspective), the authors place level shifters at either CLB output pins or input pins, since either option has costs and benefits. On the one hand, since a CLB has more input pins than output pins, placing level shifters at the input pins of a CLB results in increased area and increased static power dissipation arising from the leakage of the level shifters. On the other hand, placing level converters at CLB output pins may result in power reduction opportunities not being realized for multi-fanout nets. Figure 2.20(a) illustrates the issue; here we have a CLB configured to use the lower supply rail, $V_{DDL}$ which fans-out to two CLBs, one also configured to use the lower supply rail, while the other is configured to use the higher supply rail, $V_{DDH}$. If level converters are only available on CLB output pins, the signal voltage must be converted from $V_{DDL}$ to $V_{DDH}$ at the output of the root CLB, and as such, all switch boxes forming the fanout tree must also be configured to use $V_{DDH}$. Placing level converters at the input pins of a CLB, however, avoids this problem, as shown in Figure 2.20(b), as now the voltage conversion occurs after the signal has propagated through the interconnect network, potentially saving routing power. The authors’ experiments found that by placing level converters at CLB inputs, they were able to increase overall power reduction by 4%, while incurring an additional overall area penalty of 2%.

While the authors were able to show significant power savings – up to 61% of total power – the proposed architecture also incurs an area penalty of 21%-23% (depending on whether the level converters were located at CLB inputs or outputs), in addition to a performance degradation, which was not discussed. The increased area arises from the large supply transistors, which must be sized to supply enough current under worst case switching situations, for example if all of the routing buffers in a switch box undergo a “0” to “1” transition. While the power reductions seem promising, it would be imperative to reduce the large area cost to make multi-$V_{DD}$ architectures a more attractive proposition. This was addressed to some degree by fixed dual $V_{DD}$ architectures, discussed in the following section.

### 2.5.2 Fixed Dual-$V_{DD}$ Architecture

An alternative dual-$V_{DD}$ FPGA architecture was proposed in [70]. This FPGA architecture has a mix of CLBs, predefined to operate either at the nominal supply voltage (H-blocks), or at a reduced supply voltage to reduce power (L-blocks). All of the circuitry in the routing is assumed to operate at the
nominal supply voltage, as the authors deem the complexity in a multi-$V_{DD}$ routing fabric to be too costly due to extensive use of level shifters. The proposed architecture is shown in Figure 2.21(a). By predefining the voltage levels of the CLBs in the architecture, they avoid the overhead (in area and speed) a programmable multi-$V_{DD}$ architecture would entail, however this is at the cost of a less flexible architecture, where potentially both speed and power reduction are compromised. Effectively, the authors propose to reduce area by placing an increased burden on CAD tools, as they present a CAD flow which attempts to optimize power of the non-timing critical blocks in a netlist (during placement, these blocks would be assigned to L-blocks). For the architectures explored in this study, the authors found that the multi-$V_{DD}$ approach offers an incremental decrease in power consumption - on average an additional 2%. The power reduction in this approach is heavily dependent on the layout distribution of H-blocks and L-blocks, and for a target clock frequency, the architecture may result in an increase in the length of routing between blocks that are on timing-critical paths, resulting in increased dynamic power and offsetting the power reduction of the multi-$V_{DD}$ CLB architecture. The authors also did not assess the impact to performance, which is likely to degrade.

To remedy these problems, the authors extended their work and proposed a limited amount of programmability of the CLB supply voltage in [69]. In this work, in addition to L-blocks and H-blocks, the authors include CLBs whose supply voltage can be connected to either a low or high $V_{DD}$ level through a pair of power transistors (these are called P-blocks). This architecture is shown in Figure 2.21(b). In contrast to an architecture where all of the CLBs have a programmable supply voltage, this approach can offer advantages in speed, as the H-blocks would not suffer the performance penalty caused by power transistors, and area, since only a fraction of all of the CLBs would have power transistors. The programmability of the P-blocks makes the power reduction and performance less dependent on the layout distribution of fixed voltage CLBs, as these blocks can be configured to either operate in high speed or low power modes. In addition, these blocks can be power gated through their power

Figure 2.20: Effect of level converter placement on power savings.
transistors to further reduce power. Using the same CAD flow as their previous work, the authors study the preferred composition of L-blocks, H-blocks, and P-blocks, where these CLBs are arranged in an “interleaved” pattern (i.e., an H-block would be adjacent to an L-block, which would be adjacent to a P-block, etc.). The authors argue that an architecture comprising H-blocks, L-blocks, and P-blocks in the ratio of 1:1:3 is optimal as it leads to a 28% power reduction (roughly double the reduction in the non-programmable approach), while incurring a 14% area cost. While the impact to performance of this architecture is again not reported, performance is expected to decrease since intuitively, access between available H-blocks (which would be needed on the critical path) is restricted due to the layout pattern the authors chose.

2.6 Power Reduction in FPGA Routing Structures

2.6.1 Leakage Power Reduction

In this section, we discuss techniques to mitigate the leakage power dissipated in an FPGA’s routing network.

We begin with an overview of the leakage paths which exist in a routing switch. The subthreshold leakage of the routing switch, which dominates its static power consumption [83], is comprised of the leakage of the switch’s configuration SRAM cells, leakage in the output buffer, and leakage paths between input pins of the switch. These different leakage paths are shown in Figures 2.22(a), 2.22(b), and 2.22(c). The authors of [83] propose several techniques to reduce the leakage between input pins, and these will be discussed in the following sections.

2.6.2 Leakage Minimization using SRAM Redundancy

FPGA routing multiplexers typically consist of a first stage decoded multiplexer, followed by either a decoded or encoded multiplexer, similar to the topology shown in Figure 2.23(a). The figure shows, as an example, a 16:1 multiplexer implemented as four 4:1 decoded multiplexers acting in parallel to serve as the first stage, with a fully encoded 4:1 multiplexer serving as the second stage. While a decoded
multiplexer typically requires more transistors (owing to a greater number of SRAM cells), it offers better overall area-delay benefits as it allows for a smaller number of series pass transistors between an input path and the switch’s output buffer. In order to save area, the example multiplexer in Figure 2.23(a) shares the same configuration SRAM cells, however this leads to unnecessary leakage: if input pin 1 of the switch is to be connected to the output buffer, the shared configuration memory results in the transistors connected to input pins 5, 9, and 13 also being turned on. Thus, while leakage through the transistor connected to the pin 1 is inevitable because it is along the active path, it would be beneficial to reduce the leakage through the transistors connected to pins 5, 9, and 13 as these transistors are on inactive paths. The authors of [83] propose to do this by employing redundant configuration memory cells – instead of sharing the same configuration bits between the four parallel multiplexers for example, if pairs of multiplexers were made to share configuration memory, then half the number of input transistors would be turned on when a path is selected, reducing leakage through input pins. A multiplexer with redundant SRAM cells is shown in Figure 2.23(b).

While this approach would reduce the leakage through input pins, the increased number of SRAM cells increases configuration memory leakage. However, by using high-$V_T$ devices in the SRAM cells, the increase in overall leakage power from the additional memory cells can be reduced significantly; the total leakage reduction in this case is approximately 50%. This approach entails a fairly significant area cost – estimated to be between 15% - 30% depending on the amount of redundancy.

### 2.6.3 Dual $V_T$ Routing Structures

Another technique proposed to reduce routing switch leakage in [83] was the use of high-$V_T$ devices for pass transistors in routing switches. Using high-$V_T$ devices decreases subthreshold leakage, but it comes at the consequence of increased delay, which means that an architecture where all pass gates are made with high-$V_T$ transistors will lead to detrimental performance. However, as mentioned in Section 2.5, circuits implemented in FPGAs have many paths with considerable timing slack. Moreover, the routing conductors of an FPGA are typically under utilized [83]. As such, the authors proposed a routing architecture, where high-$V_T$ devices would only be used on a fraction of the pass transistors in the routing network, in the hopes that a timing-driven router would be able to route timing critical nets using nominal $V_T$ pass transistors and use high-$V_T$ transistors on all other paths in a bid to save power. The authors have shown that approximately 40% of pass transistors can have their $V_T$’s increased

---

**Figure 2.22:** Leakage paths in routing circuits.
by 20-45% without a considerable increase to the critical path delay. The authors estimate that this approach would result in a 40% reduction in leakage power of switch blocks.

### 2.6.4 Body Biasing and Gate Biasing

Another technique reduces the leakage of the routing switch by applying bias voltages at either the body or gate terminals of pass transistors. Recall from Section 2.3.2 that the subthreshold leakage of a transistor is an exponential function of $V_{GS} - V_T$. In the previous section, we discussed how dual-$V_T$ devices can be employed in the FPGA routing network to save power without any substantial impact on performance. However, this approach is inflexible as the $V_T$’s are fixed at fabrication time. An alternative is to set the $V_T$’s on a design-by-design basis, according to where positive slack resides in the implementation. This would naturally allow a greater number of high-$V_T$ transistors, thus allowing greater power reduction than if the transistors are to have their $V_T$’s set strictly during device fabrication. One mechanism to allow for such tuning of $V_T$’s of the pass transistors in an FPGA was shown [83] and
makes use of the body effect. The threshold voltage of a transistor is given by:

\[ V_{TH} = V_{T0} + \gamma (\sqrt{V_{SB}} + 2\phi_B - \sqrt{2}\phi_B), \]  

(2.2)

where \( V_{SB} \) is the source-to-body terminal voltage, \( V_{T0} \) is the nominal threshold, when \( V_{SB} \) is 0 V, while \( \gamma \) and \( \phi_B \) are related to device parameters. The equation shows that the threshold voltage of a transistor can be increased by increasing \( V_{SB} \). Thus, the \( V_T \) of a transistor can be adjusted after fabrication by tuning the bias voltage applied at its body terminal. Thus, the authors propose to employ this technique to adjust the \( V_T \)'s of routing pass transistors in an FPGA at the moment it is being configured. For transistors which lie along performance critical paths, \( V_{SB} \) would be set to 0 V to ensure a small \( V_T \), while for transistors which lie on paths with sufficient slack, \( V_{SB} \) would be set to a bias voltage to reduce \( V_T \) and thus reduce leakage. This offers further power gains over a dual-\( V_T \) routing architecture due to the flexibility afforded by the body biasing to trade-off delay for power reduction at configuration time, the authors report a 50% increase in leakage reduction. However, this comes at a considerable area cost, as the authors report a 60-100% increase in the area of the routing multiplexer.

An alternative approach is to decrease \( V_{GS} \) to reduce the term \( V_{GS} - V_T \). One technique to reduce the \( V_{GS} \) of pass transistors in routing switches which are off is to apply a negative bias voltage at the gate of the transistor (instead of setting the gate to 0 V). Since \( V_S \), the voltage of the source terminal of a pass transistor is greater than or equal to 0 V, setting the gate voltage to a negative bias results in \( V_{GS} < 0 \), reducing static power consumption. The gate terminal of pass transistors can easily be set to a negative voltage when they are unused by connecting the negative rail of the configuration SRAM cells (which sets the voltage on the gates of the pass transistors) to a negative voltage instead of ground.

This approach is interesting since it is akin to increasing the \( V_T \) of transistors that are off, since leakage current is a function of \( V_{GS} - V_T \), thus decreasing \( V_S \) or increasing \( V_T \) both have the same effect in reducing leakage. However, the decreased power consumption does not come at the cost of performance, since pass transistors which are active/used will not have a reduced voltage applied to their gate terminals. This technique will increase the subthreshold leakage of the SRAM cells (since the \( V_{GS} \) of an “off” NMOS transistor will now be positive), but if high-\( V_T \) transistors are used in the SRAM cells, the increased leakage can be kept below an acceptable level. However, this does restrict the magnitude of the negative bias that can be applied, since a negative bias of a large enough magnitude can cause excessive leakage in the SRAM cells. The negative bias voltage requires the use of an additional power rail, although due to the small current requirements and low transient activity of circuits connected to this additional rail, the overhead of including an additional power rail may be small. The authors expect a leakage reduction ranging from 2.5×-4× using this technique.

### 2.6.5 Routing Switch with a Virtual \( V_{DD} \)

The authors of [14] proposed reducing the power consumption of routing switches by incorporating a virtual \( V_{DD} \). Figure 2.24 shows the switch, which consists of a conventional routing switch with the exception of two transistors, \( M_{PX} \) which is a PMOS transistor, and \( M_{NX} \) which is an NMOS transistor, which connect the conventional routing buffer to the supply rail. The gates of these two transistors are connected to individual SRAM cells, and the proposed switch operates in one of three modes: in the first mode \( M_{PX} \) and \( M_{NX} \) are both on, in the second \( M_{PX} \) is off, while \( M_{NX} \) is on, and finally in the third both \( M_{PX} \) and \( M_{NX} \) are off.

In the first mode, it is clear that the node voltage of the virtual supply rail \( (V_X) \) is equal to \( V_{DD} \)
under static conditions, and as such in this first mode, the routing switch acts as a conventional switch. In the second mode, we note that since an NMOS transistor is acting as a pull-up device, the voltage of \( V_X \), under static conditions is \( V_{DD} - V_{TN} \). The virtual rail, which supplies the switch is at a reduced voltage in this mode, and this serves to reduce power in two ways. First of all, since the output of the switch can rise no higher than the voltage at \( V_X \), we see that the energy dissipated for each rising transition is \( C_{wire}V_{DD}(V_{DD} - V_T) \), where \( C_{wire} \) is the capacitance of the wire being driven by the switch. In contrast, when the switch operates in the first mode and \( V_X = V_{DD} \), the energy dissipated per transition is \( C_{wire}V_{DD}^2 \). As such when operating in low-power mode, there is a dynamic power reduction of \( V_T/V_{DD} \). Second, when \( V_X < V_{DD} \), the total subthreshold leakage of the switch is greatly reduced, since subthreshold leakage is an exponential function of \( V_{DD} \). Note that while in this mode the output high value of the routing buffer is \( V_{DD} - V_T \), additional level converters are not needed when this routing conductor connects to routing switches which are operating in the first mode (i.e., without a reduced virtual supply voltage). This is because a \( V_T \) drop is already present in conventional routing circuitry as a signal propagates from the input of a routing multiplexer to its output (the NMOS pass transistors cut-off once the source terminals reach a voltage one \( V_T \) lower than their gate voltage, which for an active pass transistor would be \( V_{DD} - V_T \)). As such, the level-restoring circuitry – as shown in Figure 2.25 – found in conventional routing buffers is already capable of converting the lower voltage level \( (V_{DD} - V_T) \) to \( V_{DD} \). The third mode of operation is when both \( M_1 \) and \( M_2 \) are both off; in this state the routing buffer is disconnected from the supply rail and is thus power gated. This mode is useful to reduce the leakage power of unused routing conductors.

An extension of this routing buffer design is shown in Figure 2.26, where the SRAM configuration memory for the switch connects to a virtual supply instead of directly to \( V_{DD} \); again a PMOS transistor \( M_{PX} \) and an NMOS transistor \( M_{NX} \) are used to connect the buffer to the rail, and the gates of these supply access transistors are connected to the same SRAM cells driving the gates of \( M_1 \) and \( M_2 \), as shown in the figure. By extending the use of virtual supplies, further power reductions are possible, as now the SRAM cells will dissipate less static power. While the two virtual supplies for the switch (one serving the output buffer, the other serving the configuration memory) could have been merged to minimize the
area overhead, the two virtual supplies were intentionally separated to isolate the SRAM’s supply from voltage transients (which occur when the output of the switch undergoes a “0” to “1” transition), as even momentary voltage droops can lead to data disturbance in the SRAM cells. Nonetheless, because the operation of the SRAM cells is static, transistors $M_{PX}$ and $M_{NX}$ can be sized to minimum widths.

The authors demonstrated leakage power reduction for their proposed routing switch of between 33-40% when operating in low-power mode and 60-65% when operating in sleep mode, depending on the amount of downstream loading and temperature. When the switch’s configuration memory is also connected to a virtual supply, leakage power reduction of 41-42% is observed in low-power mode and between 67-69% in sleep mode. The authors also report a $\sim$28% reduction in dynamic power when operating in low-power mode.

2.7 Ultra-Low-Power FPGAs

The past decade has seen increasing interest in ultra-low-power (ULP) devices, as a wide range of mobile/embedded applications, such as in health care, home automation, and environmental control,
require low cost and low power devices [22,62]. Designers have often resorted to operating circuits in the subthreshold regime to minimize the energy required per operation—since many of these applications do not require high performance but do require maximum battery life, operating circuits in the subthreshold regime is an acceptable, and in fact, a necessary strategy. Sub-$V_T$ processors have been proposed as suitable platforms for ULP applications, as they offer very low energy per instruction and flexibility as they can be used for a wide range of applications [51,111], however their simple ISAs result in large latencies in performing computations. On the other hand, sub-$V_T$ ASICs allow for drastic improvement to performance, while also offering high energy efficiency; however their limited flexibility to target multiple applications (since by definition an ASIC is application-specific) means that device cost is high for low-volume applications. Sub-$V_T$ FPGAs have therefore been proposed as a low cost, relatively high-performance, low-power platform for ULP applications.

The design of a sub-$V_T$ FPGA has recently been explored, where the different design tradeoffs and the design challenges are discussed [22,86]. In [22], the authors consider a conventional FPGA architecture and discuss potential problems faced when the $V_{DD}$ is lowered below $V_T$. The main conclusion drawn is that while FPGA logic elements can operate in subthreshold, the routing circuitry needs special consideration. With conventional FPGA routing architecture, research has shown that direct-drive buffers offer a superior area-delay tradeoff to conventional tristate buffers [63], while pass gates offer superior area efficiency but suffer from significant delay degradation [19]. However, both direct drive and tristate buffers result in increased static power dissipation, and thus from the point of view of improved energy efficiency (at the cost of decreased performance), the authors propose a routing architecture consisting exclusively of pass gates in an ULP FPGA. However, the authors note that such an approach leads to very poor tolerance to process, voltage, and temperature variation.

![Figure 2.27: Signal degradation through a pass transistor multiplexer operating in Sub-$V_T$. The current through the off transistors, $M_2$ – $M_4$, diminishes the amplitude of the output current.](image)

Consider the pass-transistor multiplexer depicted in Figure 2.27. The figure shows an input current to the active path of the multiplexer ($i_{IN}$), leakage currents through the disabled paths ($i_{leak}$), and the
output current of the multiplexer \(i_{out}\). Since this multiplexer is operating in subthreshold, the ratio \(i_{in}/i_{leak}\) is very small, and as such the signal amplitude of \(i_{out}\) is degraded by the leakage through the inactive paths of the multiplexer (since, by Kirchoff’s current law, \(i_{out} = i_{in} - 3i_{leak}\)). Furthermore, since in the subthreshold regime, a transistor’s current is an exponential function of \(V_G\), \(V_S\), and \(V_T\) (which is also related to temperature), variations in process, voltage, and temperature can result in a drastic difference in the expected output current of the multiplexer. Moreover, this effect is compounded if a signal has to propagate through multiple routing switches before reaching its destination. Thus, while pass gates are attractive for ULP applications, the fact that signal propagation and attenuation through pass gate routing switches is sensitive to process, voltage and temperature variation led the authors to propose using repeaters (buffers) in routing to allow for improved tolerance to variation at the expense of increased power consumption.

The authors subsequently aimed to remedy this problem by proposing an ULP-optimized FPGA architecture [86], wherein a pass-gate routing architecture is employed to minimize power consumption, and architectural and circuit optimizations are incorporated to mitigate the problems they observed in their initial study of ULP FPGAs. The first architectural optimization is to increase the number of BLEs in each CLB, thereby minimizing the length of routing used outside each CLB. This reduces the amount of attenuation a signal would experience as it travels through the routing network. The authors also propose to slightly increase the applied gate voltage on active pass transistors to increase their transmission properties, and finally, the authors propose adding an asynchronous, single-ended sense amplifier at the input pins of CLBs to sense a degraded signal. A sense amplifier offers superior sensitivity to weak signals compared to a conventional buffer, and as such, this approach allows for improvements to variation tolerance. The authors assessed the benefits of their proposed architecture by fabricating an FPGA featuring their proposed architectural and circuit-level enhancements, as well as a conventional FPGA architecture also operating in sub-\(V_T\). The authors demonstrated an area improvement of \(2.7 \times\), performance improvement of \(14 \times\), and energy efficiency improvement of \(4.7 \times\) with their proposed optimizations over a “conventional” sub-\(V_T\) FPGA.

2.8 Examples from Industry

2.8.1 Xilinx Low-Power Prototype

A prototype FPGA targeting battery-powered applications was presented by Xilinx, and showcased a range of different power optimization techniques [95]; specifically, the authors investigated voltage scaling, process optimization, and power gating. The authors based their architecture on a Xilinx Spartan-3, modified to incorporate various power optimization techniques.

The authors describe use of voltage scaling to reduce power, since as discussed in Section 2.5, reduction in \(V_{DD}\) results in quadratic and exponential reductions in dynamic and static power, respectively. However, reduction in \(V_{DD}\) results in increased delay, and as such this technique is a trade-off between power savings and performance penalty. Since the authors were targeting battery-powered applications, where peak silicon performance is not typically required, this power optimization technique appears to be well justified. One point worth noting is that reducing \(V_{DD}\) also reduces the noise immunity of circuits, and care must be taken in ensuring power is reduced while robustness is not compromised. In fact, while the authors propose to reduce \(V_{DD}\) to reduce power, they opt to keep configuration SRAM cells at the nominal \(V_{DD}\). This design decision was most likely influenced by a desire to ensure the configuration memory remained robust (to noise, single-event upsets (SEUs), etc.), since a failure in the
configuration memory can lead to catastrophic failure of the entire system. The authors propose to scale $V_{DD}$ from the nominal 1.2V to 1.0V, and claim a 35% reduction in active power (i.e., both dynamic and static power), and a 23% reduction in standby power (which includes only static power). This is at a performance cost of 15%.

The second optimization technique was the use of mid-oxide, high-$V_T$ devices in the configuration SRAM. As discussed in Section 2.6.2, high-$V_T$ devices can be used in the configuration memory without any negative effects, since these circuits are static and do not have any impact on an FPGA’s runtime performance. However, the use of high-$V_T$ transistors only reduces the subthreshold leakage, and has no effect on gate leakage, which can contribute up to 50% of the total leakage. The gate leakage can be reduced by observing that it is an exponential function of oxide thickness \[109\]; as such the use of thick oxide devices allows for reduced leakage. However, since the thickest oxide transistors in a standard CMOS process are geared towards I/O devices (and are thus large), they are not well-suited for an FPGA’s configuration memory cells. Thus, mid-oxide devices are used, which are suitable for large scale integration and at the same time, serve to reduce gate leakage. The use of high-$V_T$, mid-oxide transistors leads to a reduction in active power of 13%, a reduction in standby power of 43%, and an area cost of 8% (since mid-oxide transistors cannot be minimum width in a standard process).

Finally, the authors describe circuitry to enable power gating in their proposed power-optimized FPGA. Battery applications often have standby modes to reduce power, and so power gating can be employed during such states to minimize power dissipation. While this approach isn’t entirely identical to dynamic power gating (as was discussed in Section 2.4.2, dynamic power gating allows subcircuits of the FPGA to be gated while the overarching system continues to run), it does require that the configuration memory cells are never power gated. As for the granularity of the power gating scheme, the authors chose a gating region size of one tile; that is, a sleep transistor would be used to gate the power of a single tile in the FPGA. Figure 2.28 shows the power gating scheme. Note that the authors chose to use an NMOS footer transistor to act as the sleep transistor in their scheme, as they argue it provides similar power savings but better performance compared to a PMOS header transistor (shown in Figure 2.12). As shown in the figure, the authors also use isolation buffers (whose ground pins are connected to the virtual ground of the tile) at the inputs to the tile and PMOS keeper transistors (which are enabled when the tile is power gated) at the outputs of the tile; this additional circuitry is required to prevent short circuits and/or increased leakage when the tile is power gated but is connected to other tiles which are active. The power gating mechanism implemented in this prototype was able to reduce active power by 7%, as the power dissipation of unused tiles is reduced, and was able to reduce standby power by 51%. This comes at a cost of 7% impact to performance and 8% increase in area.

Overall, combining all techniques, the authors claim 46% reduction in active power and 99% reduction in standby power, with an overall 27% reduction in performance and 40% increase in area. While the area and performance costs can be attributed to the individual area and performance costs of each optimization technique, the authors claim that unoptimized layout is also a contributor to the observed area and performance costs. It is estimated that optimized layout could reduce the area overhead by almost a factor of 2, and this would also lead to reduced performance penalty as well (since larger tile area indicates routing wires are longer, and thus slower).
2.8.2 Altera Programmable Power

A power reduction technique which appears in commercial FPGAs is Altera’s *Programmable Power Technology*, which first appeared in their Stratix-III device. The architectural details and decisions made in implementing this technology were detailed in [65], and makes use of selective body biasing to minimize the power consumption of paths that do not need the highest levels of performance, similar to the body biasing technique described in Section 2.6.4. In contrast to the technique described in Section 2.6.4, Programmable Power Technology attempts to reduce power on a broader scale, since the bias voltages on the body terminals for both logic and routing resources can be adjusted to trade-off excess slack for reduced power consumption. In this power reduction technique, two different voltages are distributed throughout the chip – one voltage, $V_{\text{bnom}}$, being the nominal body bias for transistors to allow them to operate at full speed, while another, $V_{\text{blp}}$, is a bias voltage to increase the threshold voltage of transistors and thus reduce power. The body terminals of the transistors comprising logic and routing resources can be set to one of the two different voltages (through access transistors) to select between operation in a nominal mode (i.e., nominal speed and nominal leakage), or a low-speed/low-power mode, based on the availability of slack on the paths on which these resources lie. Since it has been found that a large fraction of the paths of a design operate with significant slack, and thus can be slowed down without impact to overall system performance in both a previous academic study [14] and in Altera’s own internal studies [10], Programmable Power Technology aims to offer reduced power consumption without impacting system performance through body biasing.

In architecting an FPGA to support Programmable Power Technology, two parameters must be optimized: the voltage $V_{\text{blp}}$ and the granularity of the body bias selection scheme. For NMOS devices, an increase in $V_{\text{blp}}$ leads to a corresponding exponential reduction in subthreshold leakage (as long as the P-N junction between the base and the source of the transistor continues to stay reverse biased), but this comes at the cost of an approximately linearly proportional increase in delay. Thus in an FPGA, a large voltage for $V_{\text{blp}}$ will mean each block which is set to operate in low-power mode will dissipate less...
power, but at the same time, overall there will be a reduction in the number of blocks that can be put into low power mode (since it will be more difficult to find paths with enough slack to accommodate low power mode). Thus, the choice of \( V_{blp} \) must be optimized to find a compromise between the expected number of blocks that will operate in low-power mode (have sufficient slack), and the power savings that can be achieved per block when operating in low-power mode. This trade-off, and the optimization of \( V_{blp} \) is described in a previous academic study [54].

Selective body-biasing incurs an area penalty, since it requires additional configuration memory, access transistors to allow the body to connect to the different bias voltages, and layout overhead due to design rules restricting the spacing of wells whose bias voltages are independent, not to mention the overhead due to the circuitry which is responsible for generating different body bias voltages. As such, while a finely grained body biasing scheme will offer the greatest flexibility as it allows the power/speed mode of individual circuits to be optimized, and thus offers the greatest power savings, such a scheme also results in the largest area overhead (this is similar to the granularity and area overhead trade-off we discussed concerning power gating schemes in Section 2.4). A coarser granularity on the other hand may not offer the same power reduction as it has less flexibility, but may be more favorable from an area overhead point of view. As such in [65], the authors consider a wide range of granularities, for their proposed body-biasing scheme; they experimented with architectures where the body bias for each LE could be selected independently, architectures where all the LEs within a CLB would have shared body biases, and architectures where all the LEs in two paired CLBs would have shared body biases, in order of decreasing granularity (and thus decreasing overhead). They also experimented with schemes where the routing circuitry would always operate in high-speed mode (leads to less power savings, but also less cost), schemes where the routing resources could also operate in low-power mode but would share their body biases with the logic in the associated CLBs, and schemes where the routing resources could have their body biases independently set; these different variants all offer different power-savings/overhead-cost trade-offs. The experiments revealed that since signals in the routing associated with a CLB (i.e. traversing through the CLB and/or through adjacent switch blocks and channels) are highly correlated with the signals inside the CLB, the body biases for the logic structures within a CLB and all of its associated routing circuitry can all be shared. The authors also found that the coarsest granularity which they experimented with (where the body biases of pairs of LABs would be shared) offered appreciable savings in static power, but at the same time resulted in reduced area overhead. This was the granularity that was pursued and implemented in Stratix-III and subsequent chips. The authors claimed that Programmable Power provides lower leakage at double the logic density compared to Altera’s previous 90nm Stratix-II device.

2.9 Summary

In this chapter, we briefly discussed the basics of FPGA architecture, power dissipation in FPGAs, and then provided an extensive survey of previous approaches to power reduction in FPGAs. This survey touched upon methods to apply power and clock gating in FPGAs, multi-\( V_{DD} \) FPGA architectures, and power reduction strategies for the routing networks of FPGAs. We also briefly discussed the challenges presented in ultra-low power FPGAs, which operate at subthreshold supply voltages, and finally discussed power reduction strategies that have been considered by FPGA vendors. The following chapters will present power reduction strategies which compliment or improve upon the works surveyed in this chapter.
Chapter 3

Glitch Reduction

3.1 Introduction

In this chapter, we present a glitch power reduction technique which specifically targets FPGAs. As discussed in Section 2.3.1, glitches can account for a significant portion of the dynamic power dissipated in FPGAs, with previous studies indicating that glitches account for over 20% of the dynamic power [30, 88]. We propose novel glitch filtering circuitry which serves to completely eliminate glitches of a range of pulse-widths. The circuitry is incorporated into the buffers present at logic element outputs. Glitches are eliminated immediately after they are generated, and most importantly, before they can propagate into the high-capacitance programmable interconnection network, where they would otherwise result in significant energy waste. We also present an optimization algorithm to maximize the glitch power reduction (by applying appropriate settings on each glitch filter), subject to timing constraints. We present a full CAD flow which we use to assess the merits of our power reduction technique. Experiments show that glitch power can be reduced by up to $\sim 70\%$ at an area cost of $< 3\%$, with an average critical-path degradation of $\sim 1\%$.

3.2 Background

3.2.1 Previous Work on Glitch Power Reduction

Several prior works have addressed glitch power in FPGAs [28, 58, 88, 102]. One recent approach [88] proposed to make use of the prevalent don’t-care states in logic circuits to minimize glitches. The authors proposed to set the don’t-care states to specific values such that when the input vector to a logic circuit momentarily assumes an intermediate (don’t-care) state (while it transitions between two states), the output does not make a spurious transition. This technique is a light-weight approach to glitch reduction, as it has zero performance or area overheads, and offers reasonable glitch power reduction: an average of 13.7% over a set of benchmark circuits.

More direct approaches to glitch reduction were proposed in [58] and [30]. While these two works offered different approaches to glitch reduction, the overall strategy was similar to that depicted in Figure 3.1. The figure shows two signals, $A$ and $B$, that are inputs to an XOR gate, where $B$ arrives $t_D$ seconds earlier than $A$. In theory, glitches can be eliminated if we equalize the delay along the input paths to ensure all signal transitions arrive at the same time at the different inputs to the logic circuit. This is shown in the figure, where a delay, $\Delta t$, is added to $B$. The difference in the arrival times of the two
signals to the XOR gate is therefore $t_D - \Delta t$, which is also the width of the resulting glitch at OUT. Clearly, if the added delay can be calibrated such that it is equal to $t_D$, then the two signal transitions arrive at exactly the same time to the inputs of the XOR gate, and the glitch is eliminated.

In [30], path delay equalization was proposed by inserting additional routing conductors along paths with early arrival times – the additional delay of each routing conductor slows the path down. While this approach requires no additional changes to the FPGA architecture or circuitry, and does not result in an area penalty (since routing conductors are often underutilized to begin with), the power reductions arising from the elimination of glitches are offset by the increased dynamic power due to the use of additional routing resources. In contrast, [58] proposes a modified logic element (LE) with programmable delay lines at the input pins, shown Figure 3.2. The programmable delay lines are used to adjust the arrival delays at each input pin so that they may be equalized and glitch power may be eliminated. While this approach does incur an area overhead associated with programmable delay lines on each input of a LUT (which may not be insignificant for large LUT sizes), the authors claim the ability to completely eliminate glitch power with this approach.

### 3.2.2 The Limitations of Path-Delay Balancing

One fundamental problem with the above “path delay equalization” approaches is that in general, circuit delays are a function of voltage, temperature and in some process corners, are also a function of logic state, because of unequal rise and fall delays. The reason for these effects is that modern commercial FPGAs are comprised of both CMOS gates and NMOS pass transistors. Multiplexers, which form the core of both programmable logic elements and routing switches, are effectively large trees of NMOS pass-transistors, along with CMOS level-shifters and buffers. These two different styles of circuitry respond differently to changes in operating parameters, such as temperature. We may quantify the different sensitivities to temperature by considering the analytical expressions relating the delays of these elements to transistor parameters.

The rise/fall propagation delay\(^1\) of a CMOS inverter when excited by a step-input may be derived by considering the current flowing through the inverter’s NMOS/PMOS drive transistor and into or out

---

\(^1\)This is typically defined as the difference between the point in time when an input signal of a logic gate undergoing a transition between $V_{DD}$ and GND crosses $V_{DD}/2$ and when the corresponding transition at the output of that logic gate crosses $V_{DD}/2$. 

---

**Figure 3.1:** Glitch reduction through input delay balancing.
of a load capacitor, \( C_L \), as detailed in [76]. The inverter’s delay is given by:

\[
\tau_{inv} = \frac{C_L (2V_t + \ln\left(\frac{3V_{DD} - 4V_t}{V_{DD}}\right)(V_{DD} - V_t))}{\mu C_{ox} (W/L)(V_{DD} - V_t)^2}
\]  

(3.1)

where \( \mu \), \( C_{ox} \), \( W \), \( L \), and \( V_t \) are the mobility, normalized oxide capacitance, gate width, gate length, and threshold voltage of the inverter’s NMOS or PMOS transistor, respectively.

Similarly, we may derive expressions for the delay of an NMOS pass-gate when driving a load capacitor \( C_L \). Fig. 3.3 shows an NMOS pass-gate where its drain terminal is connected to an input voltage source \( V_{in} \), its source is connected to the load, and its gate is driven to \( V_{DD} \) by a DC voltage source. If \( V_{out} \) is initially \( \sim V_{DD} \) volts, observe that the fall delay of the pass-gate (i.e. when \( V_{in} \) suddenly transitions from \( V_{DD} \) to \( GND \)) is approximately identical to Equation (3.1) since the voltage conditions at all of the terminals of the pass-gate are identical to those of the NMOS transistor of an inverter. On the other hand, there are significant differences between the rise delays of NMOS pass-gates and inverters. We may investigate this difference by first considering the differential equation which governs the time evolution

\[2\] In contrast to CMOS gates where the PMOS transistors in the pull-up network operate initially in saturation and then in triode as the output transitions from \( GND \) to \( V_{DD}/2 \), an NMOS pass-gate operates exclusively in saturation as the output transitions from \( GND \) to \( V_{DD} - V_{in} \), whereafter it operates in the sub-threshold regime. If \( V_{DD} - V_{in} > V_{DD}/2 \)
of $V_{\text{out}}$ when $V_{\text{in}}$ undergoes a sudden transition from GND to $V_{\text{DD}}$ (i.e. a positive step excitation):

$$C_L \frac{dV_{\text{out}}}{dt} = \frac{1}{2} \mu C_{\text{ox}} (W/L) (V_{\text{DD}} - V_{\text{f}} - V_{\text{out}})^2$$

From this differential equation, we may derive the rise delay of an NMOS pass-gate as:

$$\tau_{r,\text{pgate}} = \int_0^{V_{\text{DD}}/2} \frac{2C_L dV_{\text{out}}}{\mu C_{\text{ox}} (W/L) (V_{\text{DD}} - V_{\text{f}} - V_{\text{out}})^2}$$

$$= \mu NC_{\text{OX}} (W/L) (V_{\text{DD}}/2 - V_{\text{f}}) (V_{\text{DD}} - V_{\text{f}}).$$

Equations 3.1 and 3.3 shed light on the transistor parameter sensitivities of the transient responses of CMOS inverters (and by extension, any CMOS gate) and NMOS pass-gates. Equation 3.1 show that the rise/fall delay of CMOS gates will increase (in an asymptotic sense) linearly with increases in $V_{\text{f}}$, and decrease linearly with $\mu$. While both parameters decrease in magnitude with increasing temperature, the combined effect of these two parameters typically results in an overall increased delay with increasing temperature for CMOS gates [61]. The fall delay of an NMOS pass-gate will therefore also typically increase with temperature, since it possesses the same parameter sensitivities as the rise/fall delay of a CMOS gate. On the other hand Equation 3.3 indicates that while the rise delay of an NMOS pass-gate also decreases linearly with increases in $\mu$, it will increase (in an asymptotic sense) quadratically with increases in $V_{\text{f}}$. The increased sensitivity to $V_{\text{f}}$ in this case typically results in an inverse temperature-dependence characteristic – i.e. delay decreases with temperature. This characteristic of NMOS pass-gates leads to two observable characteristics:

- Paths which are dominated by pass-transistors have inverse temperature dependence on delay, while paths which are dominated by CMOS gates will more likely experience delay degradation with increased temperature.

- The inherent asymmetry in rise and fall behaviour of pass-transistors means that the rise and fall delays can only be balanced at a single PVT corner. Under other conditions, the rise and fall delays of pass-transistors are unbalanced.

LUT and routing delay data from a design placed and routed in a Stratix III FPGA illustrate these trends, as shown in Figure 3.4. Figure 3.4(a) shows the rise and fall delays for each of the six inputs to a Stratix III LUT at 0°C and 85°C. Note that inputs A, B and C, which connect to pass-transistors at deep levels of the LUT’s pass-transistor tree, exhibit an inverse temperature dependency (likely owing to the fact that these paths are pass-transistor dominated), while inputs D, E, and F, which connect to pass-transistors closer to the output of the LUT exhibit slight delay degradation with increased temperature. Note also the large difference between rise and fall delay (~45% of the rise delay) for inputs A and B. Figure 3.4(b) highlights the unpredictable and temperature-dependent rise/fall delay imbalance of routing circuitry, where we define rise/fall delay imbalance as $2|\tau_r - \tau_f|/(\tau_r + \tau_f)$; $\tau_r$ and $\tau_f$ are the rise and fall delays, respectively, of a given path through the routing network. The plot shows the relationship between the rise/fall delay imbalance of each routing path in a sample circuit at 0°C and 85°C. In the figure, it is apparent that most routing paths exhibit a delay imbalance greater which is true given that typically $V_{\text{in}} < V_{\text{DD}}/2$, a single differential equation is sufficient to model the rise transition of an NMOS pass-gate.
Figure 3.4: Temperature consequences on FPGA logic and routing delays.
than 5%, with some approaching 40% at 0°C. Qualitatively, the plot also shows the unpredictability of the delay-imbalance over temperature, since a poor correlation between delay-imbalance at the two temperature points is apparent.

These characteristics of circuit structures in FPGAs means that the relative arrival times of signals at the inputs to a gate is a strong function of both temperature and logic-state (due to rise/fall delay imbalance). This implies that if delay equalization is used to remove glitches at a particular temperature and for a particular state, the glitches may reappear or new glitches may form at a different temperature/logic state. Circuitry to compensate for these variations would be prohibitive from an area point of view, since aside from the required additional circuitry needed to sense different temperature/logic states, these variations are a function of the specific mapping, placement, and routing of a design, thus making the variations highly unpredictable.

As opposed to prior path-delay-balancing techniques, we propose a glitch reduction technique with low area overhead which has the ability to eliminate all glitches whose pulse-widths are bounded across different process, voltage, and temperature conditions. The following sections detail the proposed circuitry and associated CAD support.

### 3.3 Proposed Glitch Filtering Circuitry

#### 3.3.1 Circuit Overview

At the core of our proposed glitch power reduction technique is the circuit shown in Figure 3.5, which we call a glitch filter, as it has the ability to suppress glitches. This circuit bears some resemblance to the circuit shown in [23], although there are several subtle differences.

The proposed circuitry is effectively a buffer with a first stage inverter, formed by transistors \( M_1 \) through \( M_6 \), followed by a second inverter formed by transistors \( M_7 \) and \( M_8 \). Transistors \( M_2 \) and \( M_3 \) form gating circuitry which can disconnect the input stage of the buffer from the output stage, and this enables the glitch filtering functionality of the proposed circuit, as will be described. The circuit also comprises a programmable non-inverting delay line, labelled \( D_1 \) in the figure, whose delay determines the glitch pulse-widths which are filtered out by the circuit. \( D_1 \) in turn is comprised of \( 2^n - 1 \) delay cells which effectively form a tapped delay line (to be described in greater detail later); the \( n \) SRAM cells shown in the figure are used to configure the delay of \( D_1 \). When the glitch-filtering mechanism of this circuit is not needed (i.e. for timing critical paths), \( M_5 \) and \( M_6 \) may be turned on by an SRAM cell, allowing transistors \( M_2 \) and \( M_3 \) to be bypassed. Note that the topology shows \( M_2 \) and \( M_3 \) in the middle of the stack and thus close to \( \text{OUT} \); to improve the delay of the circuit (when not operating in glitch filtering mode), \( M_1 \) and \( M_4 \) may be placed in the middle of the stack instead (i.e. the positions of \( M_1 \) and \( M_4 \) would be swapped with \( M_2/M_6 \) and \( M_3/M_5 \), respectively). However the particular topology shown in Fig. 3.5 was chosen because it allows for increased parasitic capacitance of node \( \text{OUT} \), which is a dynamic node (as described below); this serves to reduce the worst case voltage fluctuation at this node.

To understand the operation of this circuit, we first begin with a description of the dynamic behaviour of the circuit in response to a single transition at the input, and then describe the response to a glitch (i.e. multiple consecutive transitions). Immediately following a transition at the input to this circuit – say from logic-“0” to logic-“1” – \( M_1 \) turns on and \( M_4 \) turns off. Because of the delay \( t_D \) of \( D_1 \), the output of the delay line remains at logic-“0”, and as such, \( M_3 \) remains on while \( M_2 \) remains off. Assuming the glitch filtering function of this circuit is enabled, \( M_5 \) and \( M_6 \) are both off. Consequently, this combination
of transistor states leads to both the pull-up and pull-down networks of the compound gate formed by transistors \( M_1 \) through \( M_6 \) being disconnected from node \( \text{OUT} \); at this point \( \text{OUT} \) is a dynamic node and retains its previous state from before the transition at \( \text{IN} \) (in our example, the voltage of \( \overline{\text{OUT}} \) is \( \sim V_{DD} \) volts). Therefore, as the transition from logic-“0” to logic-“1” at \( \text{IN} \) propagates through \( D_1 \), the output of the circuit \( \text{OUT} \), remains at logic-“0”. After a delay of \( t_D \) seconds, the transition at the input of the circuit is seen at the output of \( D_1 \), and this turns \( M_2 \) on, leading to \( \overline{\text{OUT}} \) being discharged, and \( \text{OUT} \) transitioning from logic-“0” to logic-“1”.

This analysis of the response of this circuit to a transition at the input highlights an important property: immediately following a transition at the input, the output of the buffer is prevented from following suit and propagating the transition at the input. Instead, the buffer is forced to wait a delay of \( t_D \) before the output stage of the circuit is connected to the input. If however, during this delay \( t_D \), the input transitions back to the previous value (i.e. a glitch has occurred), then the data value during the course of the spurious transition is not “seen” by the output stage when it is reconnected to \( B1 \) (the input buffer). As such, the spurious transition does not propagate to the \( \text{output} \), and the glitch input to the circuit is prevented from propagating and dissipating power in other areas of the chip.

While this discussion highlights how lone pulses may be filtered out, consider the scenario depicted in Fig. 3.6 where a train of pulses, \( x(t) \) appears at the input to the glitch filter (node \( \text{IN} \) in the figure). Assuming the glitch filter contains an ideal delay line, it follows that the delay-line output (node \( D_{\text{OUT}} \) in the figure) is equal to \( x(t - t_D) \), where \( t_D \) is the delay of the delay line. If \( x(t) \) is a periodic function with a period equal to \( t_D \), then by definition, the output of the delay line will be equal to \( x(t) \). Given the description of the operation of the circuit provided above, it follows that under such conditions the train of pulses will be allowed to pass through from the input to the output, even though individually each pulse has a pulse-width \( t_W < t_D \). In a more general sense, if the temporal separation, \( t_S \), between glitch \( i \) and \( i+1 \) is less than \( t_D \), glitch \( i+1 \) will only be partially filtered. If \( t_S + t_W = t_D \), where \( t_W \) is the pulse-width of a particular glitch, then the glitch will pass from the input to the output of the circuit without any attenuation or pulse-width reduction. Thus, in an analogy to passive electronic filters, while the proposed glitch filter has a low-pass characteristic, in that all lone glitches with pulse-widths less
than \( t_D \) may be filtered, it also has a resonance characteristic, where a sequence of glitches can pass through the filter without any attenuation for certain values of \( t_S \).

One simple scheme to minimize the effects of this “resonance” problem is to realize that this issue arises because the delay line of the glitch filter has a memory property, in that previous glitches persist in the delay line and may appear at the output of the delay line at the same time a new glitch has appeared at the input. This alignment of a previous glitch, delayed by the delay line with a new glitch appearing at the input is what allows the new glitch to appear at the output of the circuit unfiltered. A mechanism to effectively flush the delay line of its contents following the arrival of a restoring transition can ensure that subsequent glitches are not allowed to pass through to the output using the mechanism described above. Such a mechanism may be facilitated by constructing the delay line of the glitch filter with state-dependent asymmetric delay cells (which are inverting) as shown in Fig. 3.7. The delay of these cells is dependent on the polarity of the data transition as well as the state of the asymmetry control input which is labelled \( A \) in the figure; if \( A \) is logic-“1”, the cell will have a “slow” output-rise transition and a “fast” output-fall transition. Conversely, if \( A \) is logic-“0”, the cell will have a “slow” output-fall transition and a “fast” output-rise transition. The specific implementation of these delay cells will be discussed in the Section 3.3.3.

By constructing the glitch filter’s delay line with these delay-cells, we get the modified glitch filter shown in Fig. 3.8. Note that the delay through the delay line is now unbalanced and dependent on the state of the output of the circuit. Assume each state-dependent asymmetric delay cell has a “fast” delay of \( t_{DF} \) and a “slow” delay of \( t_D \), where \( t_{DF} \ll t_D \) by design. Let \( m \) be the number of delay cells within the delay line which are selected (and part of the signal path through the delay line). If the output is a logic-“1”, falling transitions at the input will propagate through the delay line with a delay of \( m \cdot t_D \) but a restoring transition (i.e. a rising transition at the input while the output is still at logic-“1”) will propagate through the delay line in \( m \cdot t_{DF} \) seconds. This allows a restoring transition to quickly ”flush” the delay line, to ensure previous transitions at the input do not continue to stay in flight in the delay line – this helps to significantly mitigate the resonant effects. With these enhancements, the proposed glitch filtering technique will be able to filter out glitches of pulse-width \( t_W \) seconds if \( t_W < m \cdot (t_D - t_{DF}) \) and \( t_S > t_{DF} \). Given that we will be able to make \( t_{DF} \) sufficiently small, the “resonant” effects previously

![Figure 3.6: Pulse-train at input aligned with glitch filter output leads to transmission of glitches from input to output without any attenuation.](image-url)
Chapter 3. Glitch Reduction

Figure 3.7: State-dependent asymmetric delay cell.

Described will be mitigated for a broad set of cases with this delay-line topology.

Figure 3.8: Proposed glitch-filter circuit augmented with state-dependent asymmetric delay line.

Finally, Fig. 3.9 shows the pulse-width transfer characteristic of the proposed glitch filter, when tuned to suppress glitches of pulse-width less than $\sim 300$ ps. The transfer characteristic was obtained through SPICE simulations of the response of the glitch filter to a lone pulse whose width was varied; by measuring the peak voltage of the resulting waveform at the output of the circuit for each of these pulse-widths, we may understand the glitch suppression characteristics of the circuit. The transfer characteristic shows that pulses of width less than $\sim 300$ ps are fully suppressed, while pulses of width greater than 300 ps
are allowed to pass from input to output without any attenuation.

![Figure 3.9: Pulse-width transfer characteristic of proposed glitch filter.](image)

### 3.3.2 Alternative Glitch Filter Topology

The glitch filter topology presented in the preceding section has several desirable characteristics, namely its relatively low area-overhead, highly selective glitch suppression (see Fig. 3.9), and the fact that it may easily be integrated into the logic architecture of an FPGA (to be discussed in Section 3.3.5). One potential concern with the circuit however is the fact that node $\overline{OUT}$ in Fig. 3.8 is a dynamic node, in that this node is allowed to float for $t_D$ seconds after any input transition, where $t_D$ is the (programmed) delay of the delay line. In some situations, this may be undesirable since this node may be vulnerable to coupling noise as well as leakage-induced voltage fluctuations which may lead to increased static power consumption in the output stage of the circuit. The former concern may be mitigated by ensuring that $\overline{OUT}$ is adequately shielded from other nodes to minimize the effects of capacitive coupling.

Similarly, leakage induced voltage fluctuations may be mitigated through careful circuit design. Observe that the worst case voltage fluctuation of $\overline{OUT}$, $\Delta V_{\overline{OUT}}$, due to leakage is given by:

$$\Delta V_{\overline{OUT}} = \frac{i_{\text{leak}}^{\overline{OUT}} \Delta t_{\text{dyn}}}{C_{\overline{OUT}}}, \quad (3.4)$$

where $\Delta t_{\text{dyn}}$ is the duration of time $\overline{OUT}$ is dynamic, $i_{\text{leak}}^{\overline{OUT}}$ is the average leakage current charging or discharging $\overline{OUT}$ while it is dynamic, and $C_{\overline{OUT}}$ is capacitance of node $\overline{OUT}$. Therefore, we may take measures to reduce the worst case leakage current and/or increasing the capacitance of the dynamic node to minimize the worst case voltage fluctuation for a worst case glitch pulse-width (which we wish to filter out). As an example, assuming that we wish to restrict $\Delta V_{\overline{OUT}}$ such that $\Delta V_{\overline{OUT}} \leq 100 \text{ mV}$ for $t_{\text{dyn}} = 5 \text{ ns}$, Equation 3.4 indicates that $i_{\text{leak}}^{\overline{OUT}} / C_{\overline{OUT}} \leq 2 \times 10^7 \text{ A/F}$. In 65 nm CMOS, in the worst case $i_{\text{leak}}^{\overline{OUT}}$ may be restricted to less than $\sim 10 \text{ nA}$ if high threshold voltage (HVT) transistors are employed. This therefore requires that $C_{\overline{OUT}} \geq 0.5 \text{ fF}$, which was achieved with the sizing of transistors $M_2$, $M_3$, and $M_5$ through $M_8$ which we chose for our study. In fact, the use of HVT transistors makes
it necessary to increase the sizes of $M_1$ through $M_6$ to ensure delay is not negatively impacted$^3$ which further increases the capacitance (and reduces the voltage fluctuation) of node $OUT$.

In cases where the ratio $\frac{i_{\text{leak}}}{C_{OUT}}$ may not be reduced to an acceptable level (for example in a very leaky process where even after using HVT transistors, $i_{\text{leak}}$ is still too high), node $OUT$ may be shunted with additional capacitance (using MOS capacitors for example). In either case, using HVT transistors or shunting node $OUT$ with additional capacitance leads to a small increase in switching power (since both techniques inevitably increase the capacitance that must be switched). This increase is very likely dwarfed by the overall power reduction provided by the glitch reduction technique, since the capacitance of $OUT$ is several orders of magnitude smaller than than the capacitances of inter and intra-CLB routing$^4$.

Figure 3.10: Proposed alternative glitch filter topology.

Nonetheless, we present an alternative glitch filter which functions in a similar manner to the one presented in the previous section, but does not contain any dynamic nodes, thereby making it immune to the potential complications described in this section. We observe that the asymmetric delay cell shown in Fig. 3.7 has a natural glitch filtering property: for example. if $A$ is held at logic-“0”, the cell has a fall delay of $t_D$ seconds, and a rise delay of $t_{DF}$ seconds. Assume the input to this cell is excited by a glitch which transitions from logic-“0” to logic-“1” at time $t_0$ and transitions back to logic-“0” at time $t_0 + t_P$.

Consequently, a falling transition will appear at the output at time $t_0 + t_D$, while a rising transition will appear at time $t_0 + t_P + t_{DF}$. Since $t_{DF} < t_D$, the rising transition will appear at the output “before” the falling transition if $t_P \leq t_D - t_{DF}$; in other words, the falling transition will never appear at the output under these conditions (the glitch at the input has thus been filtered). Therefore, the delay line shown in Fig. 3.8 may be used as a programmable glitch filter in its own right, as shown in Fig. 3.10.

This circuit will filter glitches of pulse-width $t_P$ if $t_P \leq m \cdot (t_D - t_{DF})$, where $m$ is an integer such that $0 \leq m \leq 2^n - 1$ (m is the number of delay cells within the delay line which are selected), while $t_{DF}$ and $t_D$ are the fast and slow delays, respectively, of each state-dependent asymmetric delay cell. Fig. 3.11 shows the simulated pulse-width transfer characteristic for this alternative glitch filter topology, where the delay line has been tuned to suppress glitches of pulse width less than $\sim 250$ ps. As with the glitch filter

$^3$The leakage of a transistor is linearly proportional to its width, $W$, but decays exponentially with $V_t$. Moreover, the performance of a transistor, as per Equation 3.1, increases linearly with $W$ and decreases linearly with $V_t$. As such, it is possible to simultaneously increase $V_t$ and $W$ such that overall leakage is reduced while performance degradation is kept to a minimum, in which case area is increased.

$^4$We measured $C_{OUT}$ to be $\sim 2.5-4.5$ fF (depending on the size of the output stage), while as we will show in Section 4.7 the capacitance of a length-4 wire in a 65-nm FPGA is estimated to be $\sim 200$ fF.
shown in the previous section, this transfer characteristic shows a relatively sharp transition between pulse-widths which are filtered and pulse-widths which are allowed to pass to the output without any attenuation, although Fig. 3.9 depicts a steeper transition between glitch suppression and transmission. For the remainder of this chapter however, we will concern ourselves solely with the glitch filter presented in the previous section. Future work may consider a comparison of the two glitch filter topologies we have presented.

\[ \text{Figure 3.11: Pulse-width transfer characteristic of alternative glitch filter topology.} \]

### 3.3.3 Delay Line Design

A number of options exist for the implementation of the programmable delay line \( D1 \). As will be discussed in Section 6.4, from our experiments we determined that each stage of our delay line had to provide, in the worst case, up to 600 \( \text{ps} \) of delay (some benchmarks required significantly less delay per stage as will be discussed in Section 6.4). Given that the FO4 delay in 65\text{nm} CMOS (for standard-\( V_t \)) is just over 20 \( \text{ps} \), achieving such a large delay per stage in an area and power-efficient manner can be
Chapter 3. Glitch Reduction

challenging. Since the delay of a CMOS gate is approximately a linear function of the product of the gate’s drive resistance and its output load capacitance, increasing either of these will result in an increase in delay. However, increasing delay solely through increasing a gate’s output capacitance would result in an unacceptably large power overhead, as such we considered two different approaches to effectively increase the drive resistance of the delay cells comprising the delay line, so that our delay targets could be achieved with minimal overhead.

The two alternative delay-cells are depicted in Figure 3.12. The first delay-cell shown in Figure 3.12(a) which we call a conventional delay cell is effectively a conventional inverter comprising of transistors with increased channel-lengths; a combination of a stack of series transistors (needed when the maximum modeled length by the foundry is less than the length necessary to meet a target delay) and transistors with increased length leads to higher drive resistance, allowing us to meet target delay. Note that while this technique also increases the input capacitance of the cell (which is the dominant load on the preceding cell in the delay-line), and thus will increase power, the power-overhead is still smaller than a delay-cell which achieves the same delay strictly through increased load capacitance. The second delay-cell shown in Figure 3.12(b), which is a current-starved delay cell, achieves increased delay by restricting the maximum pull-up/down current, by applying a bias voltage which is less (greater) than $V_{DD}$ ($GND$) on the gates of transistor $M6$ ($M5$). This technique allows us to degrade drive-resistance to meet our delay targets by finding suitable voltages $V_{PB}$ and $V_{NB}$.

These two techniques have different costs and trade-offs. For the conventional delay cell, an increase to its constituent transistors’ channel lengths results in increased area, and as mentioned previously, increased input-capacitance and thus power. In contrast, for the current-starved delay cell, as long as suitable bias voltages $V_{PB}$ and $V_{NB}$ can be generated and distributed throughout the chip in a reliable and cost efficient-manner, the transistors comprising the delay cell can be set to minimum length and width, thus minimizing area and power overheads. However, given than $V_{PB} > GND$ and $V_{NB} < V_{DD}$, transistors $M5$ and $M6$ have degraded overdrive voltage, thereby making them more sensitive to $V_t$ variation, although in this case sensitivity to variation and area overhead can be traded-off since increasing the size of $M5$ and $M6$, will reduce their $\sigma_{VT}$. In addition, there may be considerable area/power costs of distributing the voltages $V_{PB}$ and $V_{NB}$. In this work, however, we assume that both the costs of the bias generation and the current-starved delay-cell’s sensitivity to variation are insignificant, as this serves as a lower-bound estimate on the area and power overheads of the proposed glitch filtering circuitry. In contrast, the conventional delay cell allows us to form an upper-bound estimate on the area and power overheads of the glitch filtering circuit. Rigorous PVT analysis of the current-starved delay cell, as well as design and cost/benefit analysis of the bias generation circuitry is left for future work. Finally, observe that both of the delay cells shown in Figure 3.12 contain fast pull-up/down paths, which are activated by input $A$ for both cells; these paths enable the cells to have asymmetric, state-dependent delay, which serves to eliminate resonant effects as discussed previously.

3.3.4 PVT Sensitivity

Given the nature of the way in which glitches are suppressed using this technique, we may draw some conclusions about the ability of this circuit to suppress glitches in the face of variation in PVT. Section 3.2.2 highlighted the principal shortcoming in preventing glitches using delay balancing: it requires precise knowledge and control of path delays. As discussed previously, varying PVT conditions makes this difficult to guarantee in modern processes, particularly in FPGAs because of the unique combination
of various circuit structures which they employ. However, the exact bounds of path delays, and thus the bounds on the pulse-widths of resulting glitches, can be determined, and to some degree guaranteed.

The proposed glitch reduction technique in this work relies solely on the expected bounds of the glitches to be suppressed. If it is known in a circuit at a particular node which dissipates a large amount of power, that the vast majority of glitches at that node have pulse widths which are less than some bound $W_{\text{max}}$ over all PVT corners, then simply by setting the delay of the delay line of the glitch filter at that node to a value $\geq W_{\text{max}}$ ensures that those glitches will be eliminated for all possible operating conditions of the circuit. Moreover, we can always ensure that the delay line delay is greater than $W_{\text{max}}$ over all process corners; given that the delay line delay can be expressed as $t_D + \delta_D$, where $t_D$ is the nominal delay of the delay line and $\delta_D$ is some random deviation from the nominal delay due to PVT variation, we can guarantee that the delay line delay is greater than $W_{\text{max}}$ by setting $t_D \geq W_{\text{max}} + |\delta_D^{(\text{max})}|$, where $\delta_D^{(\text{max})}$ is the worst case variation of the delay line’s delay over all PVT corners (which can be modelled and bounded). There may be a resulting timing penalty in this case, but for energy critical applications, this technique provides an ability to guarantee glitch suppression. While the inherent robustness to PVT makes this approach even more appealing, in this work we perform glitch suppression using statistics gathered from a single corner. Multi-corner glitch reduction and optimization using this technique is therefore left for future work. In addition to improved robustness compared to delay-balancing glitch reduction approaches, the proposed circuit also offers reduced area overhead, since only a single glitch filter is required at the output of the BLE, whereas a BLE with input delay balancing will need a delay-line and associated circuitry at each input to the BLE, as shown previously in Figure 3.2.

3.3.5 Proposed Architecture

Figure 3.13 shows a proposed Basic Logic Element (BLE) incorporating the proposed glitch filtering circuitry. We assume a conventional BLE will have an output buffer consisting of the inverter and transistors shown in the figure; this buffer exists to restore the voltage at the output of the bypass multiplexer to full CMOS levels, and is necessary to drive the BLE’s output load. We propose to augment this buffer with glitch filtering circuitry consisting of transistors $M_1$-$M_6$, delay-line $D_1$, SRAM configuration cells, and additional auxiliary circuits as shown in the figure. The SRAM cells are used to configure the delay of the delay line. Recall that if the delay line is set to a delay of $t_D$ seconds, then all glitches input to the glitch filter of a pulse width less than $t_D$ will be filtered out. Thus, given information about the glitch statistics at the input to the glitch filter (which may be profiled through timing simulations of a given design mapped to the FPGA), the delay line can be configured accordingly to maximize glitch suppression and save power. However, the ability to filter glitches comes at a cost, namely an added delay of $t_D$ seconds. The configuration of the glitch filter delays is thus non-trivial. While it is desirable to filter out all possible glitches in the circuit, it may not be desirable because of a potential degradation in performance. Moreover, the configuration of each glitch filter must take into account effects of an added delay on downstream nodes, since the additional delay introduced by a glitch filter may in fact result in the inception of new glitches downstream. If these downstream glitches propagate through very capacitive interconnect network, there may be a resulting net increase in glitch power. The additional delay introduced by a particular glitch filter setting may also leave less room for downstream nodes to filter out glitches due to reduced slack. As such, the optimization of glitch filter settings is a combinatorial optimization problem, wherein global glitch power must be reduced under the
Figure 3.13: Proposed BLE architecture.

presence of timing constraints. The CAD flow and glitch optimization approach are described in next section.

The circuits shown above were designed and simulated using commercial 65nm STMicroelectronics models to verify functionality, and extract power and delay overheads. All simulations were conducted using Cadence’s Spectre simulator, with typical transistor models, 1V $V_{DD}$, and at a temperature of 28°C. We assume a transition time of 150 ps at the input to the buffer, and an output load of 20 fF (which is to represent the load of multiplexers and wire capacitance at the output of the BLE, similar to the loading considered in [26]). The delay penalty of the cell when the glitch-filtering mechanism is not used (and thus $M_5$ and $M_6$ are both on, thereby bypassing $M_3$ and $M_2$, respectively) is approximately 30 ps; this delay results from the drive strength degradation of the pull-up and pull-down paths because $M_1$ and $M_4$ are stacked with $M_6$ and $M_5$, respectively (i.e. a conventional buffer would have $M_1$ and $M_4$ connected to $OUT$ directly). For the baseline output buffer (without glitch filtering circuitry), under the aforementioned PVT conditions, we observed a $\sim$90 ps delay in our simulations, which grows to $\sim$120 ps with the addition of the glitch filtering circuitry.

For a glitch filter implemented with current-starved delay cells, the dynamic power overhead is $<<$ 1%, while for a conventional delay cell based delay-line, the glitch filter’s dynamic power overhead would increase to $\sim$1.5% (compared to estimated power dissipated in routing, which is typically the dominating component of total dynamic power consumption). The static power overhead is also negligible, since aside from transistors $M_1 - M_6$ and the transistors comprising input buffer $B1$, all transistors are HVT (high-threshold voltage transistors).
Finally, while the technique presented in this chapter bears some resemblance to the glitch reduction technique presented in [16], there are significant differences. In [16], the output of a combinational logic gate $g_i$ is frozen (i.e. it holds its previous state) every clock cycle for a period of $t_{\text{arr}}$ seconds after the rising edge, where $t_{\text{arr}}$ is the predetermined worst-case arrival time for gate $g_i$. This ensures glitches are suppressed since, by definition, spurious transitions cannot appear at the input to $g_i$ after a period of $t_{\text{arr}}$ seconds after the rising edge of the clock. Consequently, this approach requires the use of delay lines which have a delay proportional to the clock period. While this approach may be acceptable in an ASIC context, it would be very difficult to use in FPGAs because the clock periods of designs which target a given FPGA are not known a priori and may vary widely; for example designs with clock periods ranging from 10-100 ns may all target the same FPGA. A programmable delay line which has such a wide delay range would be very costly from an area point-of-view, and moreover, given that it must toggle every clock period it would have significant power overhead as well. In contrast, the solution presented in this chapter proposes to freeze gates after potentially spurious transitions – the technique therefore requires a delay-line range which is proportional to the expected pulse-width of glitches. From the point of view of FPGAs, this is naturally a less costly proposition than the previous technique, since in a general sense, glitch pulse-widths are much shorter than clock pulse-widths. Our proposed technique therefore promises reduced area and power overheads and overall greater practicality in an FPGA context over the previous approach.

### 3.4 Glitch Filter Setting Optimization

#### 3.4.1 CAD Flow

A proposed CAD flow supporting glitch filter-based power optimization is presented in Figure 3.14. While packing, placement, and routing remain the same as in a conventional FPGA CAD flow, we propose to add the following steps post-routing: glitch power analysis, glitch filter setting optimization, and final power analysis. We chose Altera’s Quartus II CAD software as the base CAD flow to extend.

![Figure 3.14: CAD/experimental flow for proposed glitch reduction technique.](image-url)
Details for each of the steps in our CAD flow are as follows: a delay-annotated netlist is first created after a design is synthesized, placed, and routed using Quartus’ EDA Netlist Writer tool. The tool is run with the glitch filtering setting enabled – this ensures that inertial delay specifications are contained in the output delay-annotated netlist, thus modelling the natural glitch filtering capabilities of the FPGA’s circuits. The delay-annotated netlist is then used as an input to a simulation tool (for this work we used Mentor Graphics’ Modelsim software) allowing for timing and functional simulations to be performed; functional simulations in this case are performed by removing all delay-annotations from the netlist. Note that the simulator is run with the interconnect and path transport delay options, which models all gate and interconnect delays as being purely transport if an inertial delay specification is not present in the delay annotated netlist. The results of the two simulations are then analyzed to yield cumulative distribution functions relating glitch pulse-width and glitch power are generated. Total glitch power is calculated on a node-by-node basis by comparing power reports generated from the functional and timing simulations. Since a functional simulation does not contain any glitches, differences in power between two different simulations represent glitch power. The exact distribution of pulse-widths are extracted from the timing simulation, and the relative frequency of glitches of a certain pulse width is used to estimate the dissipated power arising from glitches of that pulse width. These statistics, in addition to the netlist and its timing information (i.e. the timing graph), are used by the glitch filter setting optimization step to find the best settings for each glitch filter used in the circuit. The details of this step will be discussed in the following section.

To simulate the glitch statistics after applying our glitch filter settings, we created a simple behavioural model of our programmable glitch filter circuit as shown in Fig. 3.15; the exact glitch filter settings to be used for this circuit (as obtained from the glitch filter setting optimization step) are supplied as parameters. We augment the outputs of logic cells in the original Quartus II-generated netlist with instances of our glitch filter circuit, where each instance would have its glitch filtering parameters set by the previous stage of our CAD flow. A timing simulation is then performed on this modified netlist with the same set of random vectors used previously, and the output of this simulation is used to gauge power using Altera’s PowerPlay power estimation tool (via a ModelSim-generated .vcd file for switching activity data). Finally, the power consumption results obtained through simulations of the original netlist and the glitch filter augmented netlist may be compared to determine power savings.

### 3.4.2 Glitch Power Optimization

It is worthwhile to consider some of the challenges faced in optimization of glitch filter settings in a circuit. As mentioned above, reduction of all glitches at a node with pulse-width less than \( t_W \) results in an increase in delay at that node of \( t_W \). This trade-off must be considered carefully. To begin with, in the optimization of overall power reduction, we ought to allocate more of the timing slack available on a particular path to nodes on that path with the greatest power reduction opportunity (that is, nodes with the highest glitch power). As such, timing data must be used in conjunction with the power reduction opportunities for nodes along a particular path. In addition, we must be careful in considering the consequences a particular glitch filter setting may have on downstream nodes. When a particular node’s glitch filter is set to a setting of \( t_D \) seconds, all downstream nodes will experience a delay push-out of \( t_D \) seconds at one (or more, if there are re-convergent paths) of their inputs. This results in the profile of relative arrival times (that is, the difference in time between the signal arrival times at each input) of the inputs to each downstream node being altered, which may result in new glitches (and increased power...
module glitch_filter(in, out);

parameter slow_delay = 500; /* Delay-line delay */
parameter fast_delay = 30; /* Restoring edge delay */
input in;
output reg out;

reg dl_out;
wire fast_rise_slow_fall;
wire slow_rise_fast_fall;

/* Asymmetric delay paths */
assign #(fast_delay, slow_delay) fast_rise_slow_fall = in;
assign #(slow_delay, fast_delay) slow_rise_fast_fall = in;

/* Behavioural model of asymmetric delay line */
always @(*) begin
  if(dl_out == 0) begin
    dl_out <= #1 slow_rise_fast_fall;
  end
  else begin
    dl_out <= #1 fast_rise_slow_fall;
  end
end

/* Glitch filter output latch logic */
always @(*) begin
  if(in == dl_out) begin
    out <= in;
  end
end
endmodule

Figure 3.15: Behavioural model of glitch filter circuit.

consumption) arising downstream. Even if the power dissipation of downstream nodes does not change, the glitch statistics of downstream nodes will become “stale”, and therefore less useful for subsequent decision making.

In the absence of a model that relates changes in glitch-filter settings to altered glitch power characteristics on downstream nodes of the circuit, the glitch power analysis step (outlined above) would have to be executed frequently to ensure: (1) overall glitch power has not increased following changes to the glitch filter settings for a set of nodes and (2) the glitch power and pulse-width statistics for all affected nodes are updated so that they are always relevant and usable for glitch power optimization. This particular approach seemed to be intractable from a run-time point of view, and as such an alternative approach was pursued. Specifically, we opted to ensure that regardless the glitch filter settings applied to the various nodes in the circuit, the relative arrival times at all consequential nodes stayed unchanged. By consequential, we mean a node whose worst-case increased glitch power due to an altering of its relative fanin arrival times is significantly large. This means that nodes of relatively low output capacitance, whose worst-case glitch power is small in comparison to the potential glitch power savings elsewhere
in the circuit, would be allowed to have altered input arrival times. This approach therefore ensures that the two main concerns previously discussed are avoided. By ensuring the relative arrival times stay unchanged (for consequential nodes), we are assured that no new (consequential) glitches are created at a given node whenever we attempt to filter glitches at other points (specifically, upstream nodes) of the circuit, while this also ensures that the glitch data at each consequential node is never stale. Whenever a consequential node’s glitch filter settings are changed to a delay of $t_D$ seconds, we ensure that for all nodes downstream, the arrival times at each fanin path is similarly increased by $t_D$ seconds (by applying the necessary delay settings on upstream glitch filters). Admittedly, this is a somewhat conservative approach, and leads to compromised power reduction and development of a strategy to more optimally manage the effects of unequal delay push-out among the fanin paths of consequential nodes is left for future work.

**Problem Statement**

The optimization of glitch power as described in the preceding sections can in general be formulated as a constrained non-linear optimization problem. We are given as input a timing graph for a circuit, $G(V, E)$ where each vertex, $v \in V$, represents an input or output port of a block in the design (i.e. LUT, DFF, RAM, I/O), while each edge $e = (u, v)$, represents a signal path between ports (i.e. routing or a block’s internal timing arc). For each node $v$ in the circuit, we are given a glitch power density function, $GP_v(t)$ (this is obtained empirically during the glitch power analysis step in the CAD flow described in Section 3.4.1), which describes the amount of power dissipated at node $v$ by glitches of width $t$. The total glitch power at node $v$ is therefore:

$$P_v^{(\text{max})} = \int_0^\infty GP_v(\tau)d\tau.$$  \hspace{1cm} (3.5)

At each node $v$, we have a decision variable $d_v$, which is the specific glitch filter setting at node $v$. As described previously, the proposed glitch filtering circuitry is able to eliminate all glitches of pulse width less than $d_v$ from appearing at its output when the glitch filter’s delay line is set to a delay of $d_v$. This means that given a glitch filter setting of $d_v$, the total glitch power at node $v$ would be reduced to:

$$P_v = \int_{d_v}^\infty GP_v(\tau)d\tau.$$ \hspace{1cm} (3.6)

Therefore, total glitch power in the entire design (which we aim to minimize) is:

$$P_{\text{total}} = \sum_{v \in V} \int_{d_v}^\infty GP_v(\tau)d\tau.$$ \hspace{1cm} (3.7)

At each node $v$, we wish to keep track of the worst-case arrival time, $arr_v$, and this is expressed as:

$$arr_v = \max_{(u, v) \in E} arr_u + d_u + t_{uv},$$ \hspace{1cm} (3.8)

where $t_{uv}$ is the delay from node $u$ to $v$ in the graph, and $d_u$ is the delay push-out caused by the glitch filter setting at $u$ (i.e. it is $u$’s glitch filter delay setting). Let $CO$ be the set of circuit outputs – i.e. primary outputs, flip flop inputs, etc. Given a constrained critical path of $T$, we have the following constraint:

$$\forall v \in CO : arr_v \leq T$$ \hspace{1cm} (3.9)
As described in the previous section, we also wish to ensure that the delay push-out on each fanin edge of each node $v$ are within some tolerance level of one another. However, this should be a soft constraint, as we only wish to avoid the case of unequal input delay push-out on nodes which are consequential. Similar to arrival time, we can keep track of the minimum and maximum input delay push-out on each node $v$, $dp^{(min)}_v$ and $dp^{(max)}_v$ respectively, with the following equations:

$$dp^{(min)}_v = \min_{(u,v) \in E} dp^{(min)}_u + d_u$$ (3.10)

$$dp^{(max)}_v = \max_{(u,v) \in E} dp^{(max)}_u + d_u$$ (3.11)

We wish to ensure that $dp^{(max)}_v$ and $dp^{(min)}_v$ are within some acceptable threshold of one another, i.e.:

$$dp^{(max)}_v - dp^{(min)}_v \leq K_{tol}$$ (3.12)

Where $K_{tol}$ is the maximum allowed mismatch between input push-out (which can for example be obtained empirically and set to a constant value). Equation 3.12 represents a hard constraint, and so we may allow this constraint to be relaxed with the following modification:

$$dp^{(max)}_v - dp^{(min)}_v - K_{eq} \cdot e_v \leq K_{tol}$$ (3.13)

In contrast to Equation 3.12, we introduce a large constant $K_{eq}$ (which can be set to some value which provably will always be greater than $dp^{(max)}_v - dp^{(min)}_v$, such as the critical path delay), and a slack variable $e_v$, which is binary, and allows the constraint in Equation 3.12 to be violated for certain nodes. In order to objectively trade-off glitch power reduction opportunities available in the circuit with the requirement that nodes have equal delay push-out on input edges, we introduce a penalty term to Equation 3.7 to form our objective function:

$$P_{obj} = \sum_{v \in V} \int_{d_v}^{\infty} GP_v(\tau) d\tau + \sum_{v \in V} P_v \cdot K_p \cdot e_v,$$ (3.14)

where $K_p$ is an empirically determined penalty factor. The second term in Equation 3.14 effectively indicates that whenever a node’s input fanin delay push-outs are allowed to be unequal to one another, we must pay a penalty in power consumption pessimistically set to $K_p \cdot P_v$ (i.e. all glitch power reduction at node $v$ is now lost, and some additional power penalty may be incurred if $K_p > 1$). This ensures that we are careful in balancing the input delay push-out on consequential nodes, while allowing us to violate this condition on inconsequential nodes if this would allow for greater glitch power reduction in other parts of the circuit. Equations 3.8 - 3.11 and Equations 3.13 - 3.14 together define a constrained optimization problem.

**MILP Formulation**

While the optimization of Equation 3.14 may at first appear to be intractable given that $GP_v(t)$ are arbitrary non-linear functions, we can simplify the equation by observing that each glitch filter’s delay line has finite resolution and a limit on its maximum delay. In other words, $d_v = di_v \cdot res$, where $res$ is the finite resolution of the delay line, and $di_v$ is an integer decision variable (it is effectively the delay line setting for node $v$’s glitch filter) in the range from $0 \leq di_v \leq 2^B - 1$, where $B$ is the
number of configuration bits of the delay line. This means that given the finite number of values for \( d_v \), we also have a finite number of possible values for the glitch power dissipated at node \( v \). This observation allows us to recast Equation 3.14 as a linear equation coupled with linear constraints. First, let \( g_p_v[k] = \int_{(k-1) \text{res}}^{k \text{res}} GP_v(\tau) \, d\tau \) where \( 1 \leq k \leq 2^B - 1 \). We can then rewrite Equation 3.14 as:

\[
P_{\text{obj}} = \sum_{v \in V} \sum_{k=1}^{2^B-1} x_v[k] \cdot g_p_v[k] + \sum_{v \in V} \int_{0}^{\infty} GP_v(\tau) \, d\tau + \sum_{v \in V} P_v \cdot K_p \cdot e_v \tag{3.15}
\]

Where \( x_v[k] \) are binary decision variables for node \( v \), and indicate whether or not the glitch power corresponding to \( g_p_v[k] \) can be eliminated (i.e. if \( x_v[k] = 0 \), \( g_p_v[k] \) can be eliminated), and \( t_{DM} = (2^B - 1) \cdot \text{res} \) corresponds to the maximum delay of the delay line. The second term in this Equation describes glitch power that cannot be reduced due to the finite maximum delay of the glitch filter delay line. As such, this term is a constant, and therefore Equation 3.15 is a linear function of \( x_v[k] \) and \( e_v \).

We may relate this objective function with the underlying decision variables \( d_i \) with the following linear constraints at each node \( v \):

\[
\forall_{k \in \{1, \ldots, 2^B-1\}} : \text{res} \cdot d_i + k \cdot \text{res} \cdot x_v[k] \geq k \cdot \text{res} \tag{3.16}
\]

This constraint indicates that if \( d_i = j, x_v[k] = 1 \) for \( k > j \). Conversely, the optimizer has the freedom to set \( x_v[k] = 0 \) for \( k \leq j \) since the constraint will still be satisfied regardless of the (legal) value of \( x_v[k] \) within this range. The relationship between glitch power savings and delay line setting is therefore conveyed (albeit implicitly) through this constraint: if the delay line of the glitch filter corresponding to node \( i \) has a delay equal to \( j \cdot \text{res} \) seconds, all glitches of pulse-width less than \( j \cdot \text{res} \) seconds will be suppressed, allowing for power reduction. This is reflected in the objective function since the weights \( x_v[k] \) of the terms which correspond to the power consumption at node \( i \) for this range of pulse-widths \( g_p_v[k] \) for \( k \leq j \) can be set to 0. From the point of view of the objective function, the power dissipation corresponding to these glitches has been eliminated. However, since \( x_v[k] = 1 \) for \( k > j \), the terms corresponding to the glitch power at \( i \) for pulse widths greater than \( j \cdot \text{res} \) seconds will continue to count towards the total power dissipated in the circuit from the objective function’s perspective. In other words, the delay line setting is unable to filter glitches of pulse-width greater than \( \text{res} \cdot j \), and so the glitch power corresponding to these glitches cannot be eliminated.

The \( \min \) and \( \max \) constraints in Equations 3.8, 3.10 and 3.11 also represent non-linearities in our problem formulation, however these too may be recast into a linear form through the following constraints:

\[
\text{arr}_v \geq \forall_{(u,v) \in E} \text{arr}_u + d_u + t_{uv} \tag{3.17}
\]

\[
d_{p_v}^{\text{min}} \leq \forall_{(u,v) \in E} d_{p_u}^{\text{min}} + d_u \tag{3.18}
\]

\[
d_{p_v}^{\text{max}} \geq \forall_{(u,v) \in E} d_{p_u}^{\text{max}} + d_u \tag{3.19}
\]

Since Equation 3.15 is minimized whenever \( \text{arr}_v \) and \( d_{p_v}^{\text{max}} \) are minimized, and whenever \( d_{p_v}^{\text{min}} \) is maximized, any optimal solution to the objective function will guarantee that \( \text{arr}_v \) will be suitably minimal, while ensuring that the constraints in Equation 3.17 are met. As such, for all practical purposes, \( \text{arr}_v = \max_{(u,v) \in E}(\text{arr}_u + d_u + t_{uv}) \). Similar claims can also be made for \( d_{p_v}^{\text{max}} \) and \( d_{p_v}^{\text{min}} \) to ensure
that they are effectively the \( \max \) and \( \min \) of their respective arguments.

Together, the objective function in Equation 3.15 and the constraints in Equations 3.9, 3.13 and Equations 3.16-3.19 define a mixed-integer linear program (MILP), with binary variables \( x_v[k] \) and \( e_v \), integer variables \( d_v \), and continuous variables \( arr_v, dp_v^{(\min)} \) and \( dp_v^{(\max)} \). This MILP can be solved using standard mathematical optimization software. We used the commercial Gurobi Optimizer tool, version 6.0.5.

### 3.5 Experimental Study

We conducted a set of experiments to assess the power reductions attainable using our proposed technique. Our methodology is summarized in the CAD flow shown in Fig. 3.14. A circuit is first compiled using Altera’s Quartus II software, targeting 65 nm Stratix-III devices, to generate a delay-annotated netlist. This delay annotated netlist is then input to ModelSim for timing simulation, and the appropriate input vectors are used for simulation (10000 random vectors are generated for the MCNC circuits, while the UMass RCG HDL benchmark circuits are provided with test benches containing circuit-specific input vectors). A functional simulation is also performed using the same set of vectors to allow us to characterize the glitch statistics of the circuit. These glitch statistics along with timing information (also output by Quartus II in Standard Delay Format (SDF)) are then input to the glitch setting optimization framework described in Section 6.3. As indicated in Fig. 3.14, all power estimates were obtained from Altera’s PowerPlay power analyzer tool, which was supplied with signal activity data from timing/functional simulations. All experiments were run on the SciNet high performance computing system.

We conducted experiments on the 20 largest MCNC benchmark circuits, as well as the 6 circuits from the UMass RCG HDL Benchmark Collection. Since the glitch analysis step in our CAD flow requires functional and timing simulations, we chose the MCNC benchmark circuits as it is straightforward to generate testbenches for these designs that result in sufficient toggling on their internal nodes without requiring an intimate knowledge and understanding of how each of these benchmarks work. The UMass RCG HDL benchmarks were chosen because ready-made test benches are provided with the benchmark set; Table 3.1 provides a description of the circuits in this benchmark. Table 3.2 summarizes the resource utilization and baseline dynamic power breakdown characteristics of the benchmark circuits used in our study; note that the last column of the table indicates the portion of core dynamic power resulting from glitches for each circuit.

#### Table 3.1: UMass RCG HDL Benchmark set details.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ava</td>
<td>Viterbi algorithm-based error correction decoder</td>
</tr>
<tr>
<td>fdlc</td>
<td>Computes the discrete cosine transform</td>
</tr>
<tr>
<td>fir_filter</td>
<td>Simple digital finite impulse response filter</td>
</tr>
<tr>
<td>jpeg_encoder</td>
<td>JPEG image compression encoder</td>
</tr>
<tr>
<td>top_rs_decode</td>
<td>Reed-Solomon decoder system</td>
</tr>
<tr>
<td>turboSRAM</td>
<td>Error correction decoder based on turbo codes, implements the adaptive soft-output Viterbi algorithm</td>
</tr>
</tbody>
</table>

One important aspect of our experimental study was to investigate different trade-offs in area overhead and power reduction corresponding to the parameters of the glitch filter circuit. We considered the

---

5Experiments were run on SciNet’s General Purpose Cluster (GPC) whose nodes contain an Intel Xeon E5540 processor (four 2.53 GHz CPUs) with 8GB of main memory.
impact on power reduction from quantization effects in the reduction in resolution and finite maximum delay of the delay lines. Indeed, a glitch filter whose delay line is infinitely precise and has infinite range would offer the greatest flexibility, and thus the greatest opportunity to reduce power. On the other hand, a delay line with large range and fine granularity would also require many stages and SRAM cells, thus presenting an area overhead. Our experiments shed light on an appropriate choice for these parameters.

### 3.5.1 Maximum Power Reduction Assuming Ideal Delay Lines

The first set of experiments we conducted were to assess the maximum possible power reductions assuming an ideal delay line (i.e., infinite precision and infinite maximum range). The experiments provide an upper bound on the achievable power reduction, prior to our optimization of the range and precision of the delay-line. Table 3.3 summarizes the power reduction and critical path degradation results for the case where a glitch-filter uses an ideal delay line (which has no limits on range or resolution, and does not have any area/power overhead), and for two specific delay-line implementations which will be discussed in the next section. The table lists glitch power reduction along with the resulting reduction to core dynamic power for each circuit in the benchmark set. It should be noted that the amount of dynamic power dissipated in logic (i.e., BLEs and FFs) and routing (i.e., parts of the FPGA which are affected by glitches) versus that dissipated in other parts of an FPGA (those that are not affected by glitches) varies from one design to another. For instance, in the UMass benchmark set, the turboSram benchmark’s core dynamic power is dominated by the power dissipated in memory blocks, while logic and routing power contributes a small percentage to overall power dissipation. In contrast in other benchmarks, such as jpeg, power dissipated in logic and routing power is dominant. Note that virtually no glitches were observed while simulating the ava benchmark from the UMass benchmark set, as such the table entries corresponding to glitch power reduction for this benchmark are left empty.

#### Table 3.2: Benchmark resource utilization and dynamic power statistics

<table>
<thead>
<tr>
<th>Circuit</th>
<th># LUTs</th>
<th># BRAM bits</th>
<th># DSP blocks</th>
<th>Logic [%]</th>
<th>Routing [%]</th>
<th>Clock [%]</th>
<th>Memory [%]</th>
<th>DSP [%]</th>
<th>Glitch [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>1522</td>
<td>0</td>
<td>0</td>
<td>69.7</td>
<td>30.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>18.9</td>
</tr>
<tr>
<td>apex2</td>
<td>1878</td>
<td>0</td>
<td>0</td>
<td>66.4</td>
<td>33.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>16.3</td>
</tr>
<tr>
<td>apex4</td>
<td>1262</td>
<td>0</td>
<td>0</td>
<td>55.6</td>
<td>44.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>26.0</td>
</tr>
<tr>
<td>bigkey</td>
<td>1699</td>
<td>0</td>
<td>0</td>
<td>33.3</td>
<td>45.6</td>
<td>21.1</td>
<td>0.0</td>
<td>0.0</td>
<td>12.7</td>
</tr>
<tr>
<td>clma</td>
<td>8365</td>
<td>0</td>
<td>0</td>
<td>52.5</td>
<td>34.1</td>
<td>13.4</td>
<td>0.0</td>
<td>0.0</td>
<td>16.9</td>
</tr>
<tr>
<td>des</td>
<td>1591</td>
<td>0</td>
<td>0</td>
<td>32.3</td>
<td>67.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>30.2</td>
</tr>
<tr>
<td>diffq</td>
<td>1494</td>
<td>0</td>
<td>0</td>
<td>20.7</td>
<td>1.1</td>
<td>78.3</td>
<td>0.0</td>
<td>0.0</td>
<td>1.2</td>
</tr>
<tr>
<td>dsp</td>
<td>1362</td>
<td>0</td>
<td>0</td>
<td>37.1</td>
<td>48.9</td>
<td>14.0</td>
<td>0.0</td>
<td>0.0</td>
<td>12.6</td>
</tr>
<tr>
<td>elliptic</td>
<td>3602</td>
<td>0</td>
<td>0</td>
<td>34.0</td>
<td>29.8</td>
<td>36.3</td>
<td>0.0</td>
<td>0.0</td>
<td>11.4</td>
</tr>
<tr>
<td>exa10</td>
<td>4598</td>
<td>0</td>
<td>0</td>
<td>43.1</td>
<td>56.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>26.5</td>
</tr>
<tr>
<td>ex5p</td>
<td>1064</td>
<td>0</td>
<td>0</td>
<td>50.0</td>
<td>50.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>42.2</td>
</tr>
<tr>
<td>frisc</td>
<td>3539</td>
<td>0</td>
<td>0</td>
<td>30.7</td>
<td>18.0</td>
<td>51.3</td>
<td>0.0</td>
<td>0.0</td>
<td>11.1</td>
</tr>
<tr>
<td>memc3</td>
<td>1397</td>
<td>0</td>
<td>0</td>
<td>60.9</td>
<td>39.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>23.1</td>
</tr>
<tr>
<td>pdc</td>
<td>4575</td>
<td>0</td>
<td>0</td>
<td>45.6</td>
<td>54.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>30.6</td>
</tr>
<tr>
<td>s298</td>
<td>1930</td>
<td>0</td>
<td>0</td>
<td>47.8</td>
<td>29.9</td>
<td>22.4</td>
<td>0.0</td>
<td>0.0</td>
<td>8.5</td>
</tr>
<tr>
<td>s38417</td>
<td>6042</td>
<td>0</td>
<td>0</td>
<td>39.7</td>
<td>38.6</td>
<td>21.7</td>
<td>0.0</td>
<td>0.0</td>
<td>1.8</td>
</tr>
<tr>
<td>s38584_1</td>
<td>6177</td>
<td>0</td>
<td>0</td>
<td>52.0</td>
<td>34.3</td>
<td>13.7</td>
<td>0.0</td>
<td>0.0</td>
<td>4.8</td>
</tr>
<tr>
<td>seq</td>
<td>1750</td>
<td>0</td>
<td>0</td>
<td>62.8</td>
<td>37.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>17.8</td>
</tr>
<tr>
<td>spia</td>
<td>3600</td>
<td>0</td>
<td>0</td>
<td>45.2</td>
<td>54.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>27.9</td>
</tr>
<tr>
<td>tseng</td>
<td>1046</td>
<td>0</td>
<td>0</td>
<td>26.2</td>
<td>0.0</td>
<td>73.8</td>
<td>0.0</td>
<td>0.0</td>
<td>2.4</td>
</tr>
<tr>
<td>ava</td>
<td>5978</td>
<td>0</td>
<td>0</td>
<td>29.9</td>
<td>0.1</td>
<td>70.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>fditc</td>
<td>18325</td>
<td>0</td>
<td>0</td>
<td>48.4</td>
<td>34.3</td>
<td>17.3</td>
<td>0.0</td>
<td>0.0</td>
<td>12.0</td>
</tr>
<tr>
<td>fir_filter</td>
<td>235</td>
<td>0</td>
<td>25</td>
<td>19.8</td>
<td>11.3</td>
<td>65.2</td>
<td>0.0</td>
<td>0.0</td>
<td>3.7</td>
</tr>
<tr>
<td>jpeg_encoder</td>
<td>18600</td>
<td>0</td>
<td>0</td>
<td>52.2</td>
<td>35.9</td>
<td>11.8</td>
<td>0.0</td>
<td>0.0</td>
<td>20.8</td>
</tr>
<tr>
<td>top_mdll</td>
<td>17502</td>
<td>48384</td>
<td>8</td>
<td>29.0</td>
<td>6.1</td>
<td>36.7</td>
<td>27.8</td>
<td>0.3</td>
<td>1.4</td>
</tr>
<tr>
<td>turboSram</td>
<td>1140</td>
<td>23552</td>
<td>0</td>
<td>8.0</td>
<td>12.6</td>
<td>14.3</td>
<td>65.1</td>
<td>0.0</td>
<td>1.4</td>
</tr>
</tbody>
</table>

---

6 This was achieved by removing the resolution and range constraints for the delay line decision variable ($d_{i,e}$) from Eq. (3.10).
Turning our attention now to the first section of the table, we see that with an ideal delay line, glitch power reductions ranging from 45-97%, with an average reduction of 75% may be obtained. Core dynamic power savings range from 0% (for the ava benchmark) to 22%. Average logic and routing dynamic power reduction is \( \sim 10.4\% \) over the set of benchmark circuits.

For the other two delay-lines shown in the table, we pessimistically assume that these would be composed of the conventional delay cells shown in Figure 3.12(a). In our results, we include the simulated power overhead of a delay-line which comprises these delay-cells, in addition to the increased routing power resulting from the area-overheads of the glitch filtering circuitry. As will be described in the subsequent section, for a given delay-line resolution and maximum delay, we may estimate the resulting area overhead. Given an area overhead of \( x \), we estimate that the tile dimensions will increase by a factor of \( \sqrt{1 + x} \), and thus routing and clock power will (pessimistically) increase by this factor as well. Given the power-breakdown data which we obtained (Table 3.2), we are able to estimate the resulting impact to total core dynamic power.

Table 3.3 also lists the critical path degradation for each circuit. Recall in Section 3.3 we discussed a delay penalty of 30 ps for the glitch-filter circuit even when the its glitch filtering mechanism is unused (e.g. for timing critical signals). This nominal delay penalty results in a slight impact to the critical path delay for the circuits considered in this work. Observe that the critical path degradation ranges from 0.2% to 2.8%, with an average critical path degradation of 0.9%. Note that the different delay-lines in the table have equal impacts to critical path, since the critical path is unaffected by the range or resolution of the delay-lines.

### 3.5.2 Power Reduction with Real Delay Lines

After establishing the maximum possible power savings for each design under ideal circumstances, we experimented with various resolutions and number of bits (i.e. maximum delay line delay) in a bid to identify the delay-line parameters to realize maximum power savings/minimum area overhead. The results of these experiments are shown in Figure 3.16.
Table 3.3: Detailed glitch and dynamic power reductions for glitch filters with three delay line variants.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Critical Path [ns]</th>
<th>Glitch Reduction [%]</th>
<th>Core Dynamic Power Savings [%]</th>
<th>Critical Path Degradation [%]</th>
<th>Glitch Reduction [%]</th>
<th>Core Dynamic Power Savings [%]</th>
<th>Critical Path Degradation [%]</th>
<th>Glitch Reduction [%]</th>
<th>Core Dynamic Power Savings [%]</th>
<th>Critical Path Degradation [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>22</td>
<td>70</td>
<td>13</td>
<td>1.3</td>
<td>68</td>
<td>11</td>
<td>1.3</td>
<td>65</td>
<td>12</td>
<td>1.3</td>
</tr>
<tr>
<td>apex2</td>
<td>26</td>
<td>81</td>
<td>13</td>
<td>1.1</td>
<td>79</td>
<td>12</td>
<td>1.1</td>
<td>76</td>
<td>12</td>
<td>1.1</td>
</tr>
<tr>
<td>apex4</td>
<td>24</td>
<td>73</td>
<td>19</td>
<td>1.3</td>
<td>71</td>
<td>17</td>
<td>1.3</td>
<td>70</td>
<td>17</td>
<td>1.3</td>
</tr>
<tr>
<td>bigkey</td>
<td>18</td>
<td>63</td>
<td>8</td>
<td>0.5</td>
<td>56</td>
<td>6</td>
<td>0.5</td>
<td>50</td>
<td>6</td>
<td>0.5</td>
</tr>
<tr>
<td>clma</td>
<td>22</td>
<td>65</td>
<td>11</td>
<td>2.2</td>
<td>60</td>
<td>9</td>
<td>2.2</td>
<td>50</td>
<td>8</td>
<td>2.2</td>
</tr>
<tr>
<td>des</td>
<td>28</td>
<td>63</td>
<td>19</td>
<td>0.5</td>
<td>59</td>
<td>16</td>
<td>0.5</td>
<td>59</td>
<td>17</td>
<td>0.5</td>
</tr>
<tr>
<td>diffeq</td>
<td>18</td>
<td>93</td>
<td>1</td>
<td>0.2</td>
<td>92</td>
<td>-0.1</td>
<td>0.2</td>
<td>92</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>dsip</td>
<td>17</td>
<td>85</td>
<td>11</td>
<td>0.4</td>
<td>74</td>
<td>8</td>
<td>0.4</td>
<td>53</td>
<td>6</td>
<td>0.4</td>
</tr>
<tr>
<td>elliptic</td>
<td>17</td>
<td>74</td>
<td>8</td>
<td>0.2</td>
<td>70</td>
<td>7</td>
<td>0.2</td>
<td>60</td>
<td>6</td>
<td>0.2</td>
</tr>
<tr>
<td>ex1010</td>
<td>28</td>
<td>82</td>
<td>22</td>
<td>1</td>
<td>79</td>
<td>19</td>
<td>1</td>
<td>78</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>ex5p</td>
<td>26</td>
<td>45</td>
<td>19</td>
<td>1.2</td>
<td>42</td>
<td>16</td>
<td>1.2</td>
<td>42</td>
<td>17</td>
<td>1.2</td>
</tr>
<tr>
<td>frisc</td>
<td>17</td>
<td>63</td>
<td>7</td>
<td>0.3</td>
<td>60</td>
<td>5</td>
<td>0.3</td>
<td>55</td>
<td>5</td>
<td>0.3</td>
</tr>
<tr>
<td>misex3</td>
<td>24</td>
<td>83</td>
<td>19</td>
<td>1</td>
<td>65</td>
<td>13</td>
<td>1</td>
<td>60</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>pdc</td>
<td>31</td>
<td>62</td>
<td>19</td>
<td>1</td>
<td>62</td>
<td>17</td>
<td>1</td>
<td>45</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>s298</td>
<td>21</td>
<td>59</td>
<td>5</td>
<td>2.8</td>
<td>57</td>
<td>3</td>
<td>2.8</td>
<td>50</td>
<td>3</td>
<td>2.8</td>
</tr>
<tr>
<td>s38417</td>
<td>17</td>
<td>96</td>
<td>2</td>
<td>0.7</td>
<td>96</td>
<td>-0.2</td>
<td>0.7</td>
<td>95</td>
<td>1</td>
<td>0.7</td>
</tr>
<tr>
<td>s38584.1</td>
<td>24</td>
<td>84</td>
<td>4</td>
<td>1</td>
<td>80</td>
<td>2</td>
<td>1</td>
<td>58</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>seq</td>
<td>23</td>
<td>83</td>
<td>15</td>
<td>1</td>
<td>82</td>
<td>13</td>
<td>1</td>
<td>79</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>spla</td>
<td>28</td>
<td>71</td>
<td>20</td>
<td>1.1</td>
<td>70</td>
<td>18</td>
<td>1.1</td>
<td>66</td>
<td>17</td>
<td>1.1</td>
</tr>
<tr>
<td>tseng</td>
<td>18</td>
<td>75</td>
<td>2</td>
<td>0.2</td>
<td>55</td>
<td>0.3</td>
<td>0.2</td>
<td>53</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>ava</td>
<td>4.3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>-1</td>
<td>1</td>
<td>-</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>f dct</td>
<td>7.2</td>
<td>97</td>
<td>12</td>
<td>0.7</td>
<td>94</td>
<td>10</td>
<td>0.7</td>
<td>94</td>
<td>11</td>
<td>0.7</td>
</tr>
<tr>
<td>fir_filter</td>
<td>12.6</td>
<td>76</td>
<td>0</td>
<td>0.2</td>
<td>72</td>
<td>-1</td>
<td>0.2</td>
<td>72</td>
<td>-0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>jpeg</td>
<td>7.8</td>
<td>93</td>
<td>19</td>
<td>0.5</td>
<td>85</td>
<td>16</td>
<td>0.5</td>
<td>85</td>
<td>17</td>
<td>0.5</td>
</tr>
<tr>
<td>RS decoder</td>
<td>7.7</td>
<td>73</td>
<td>1</td>
<td>1.2</td>
<td>71</td>
<td>0.3</td>
<td>1.2</td>
<td>66</td>
<td>1</td>
<td>1.2</td>
</tr>
<tr>
<td>turboSram</td>
<td>4.2</td>
<td>73</td>
<td>1</td>
<td>0.8</td>
<td>65</td>
<td>0.4</td>
<td>0.8</td>
<td>65</td>
<td>1</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Mean: | 75.3 | 10.4 | 0.9 | 70.6 | 8.4 | 0.9 | 65.5 | 8.3 | 0.9 |
The figure shows 16 different combinations of delay-line parameters, as resolution is varied from 150 ps to 750 ps, while the maximum delay of the delay line is varied from 1.6 ns to 2.8 ns. As expected, the results show monotonic decreases in power consumption as the resolution is increased; reduced resolution is associated with reduced flexibility to eliminate glitches. At the same time, we observe that for the same resolution, the ability to eliminate glitches increases monotonically as the maximum delay of the delay line (and thus, number of configuration bits) increases.

However, the plot reveals an interesting trend: while the delay-line in this study with the finest resolution (150 ps) and longest delay (2.8 ns) allowed us to achieve a power reduction of just over 72%, approaching the maximum theoretical glitch power reduction of 75%, the other delay lines with coarser resolution and reduced maximum delay were still able to achieve glitch power reductions within \( \sim 5\text{-}10\% \) of the theoretical maximum. This compels us to further investigate the area-power trade-offs of these various delay-lines.

We begin by establishing an estimate for the area overhead of the proposed glitch-filter circuit. The number of minimum-width transistors introduced as overhead by this circuit (including configuration memory) for a delay-line constructed with current-starved delay cells is approximately:

\[
A_{CS} = 8 + 12n + 6S \quad (3.20)
\]

Where \( n \) is the number of bits, and \( S \) is the number of stages in the delay line. For a conventional delay cell based delay-line, the area overhead is approximately:

\[
A_{Conv} = 8 + 12n + 10S \quad (3.21)
\]

We further estimate that a suitably-sized 6-input LUT has an area of \( \sim 1110 \) minimum-width transistors, by using transistor sizing data obtained through area-delay optimization of a conventional island-style FPGA architecture [26] (i.e. similar to Stratix III). Also, assuming that the area of an FPGA tile is broken down as follows: 50% routing, 30% LUTs and 20% for miscellaneous circuitry [27], we can form an estimate for the area overhead of the proposed glitch-filters for varying delay-line resolutions and maximum delay-line delays. For the 16 delay-line parameter combinations explored, we may then obtain the greatest power reduction for a given area overhead. The maximum power-reductions, for the two different delay-lines and over the range of area-overheads considered in this work are shown in Figure 3.17.

Interestingly, the plot shows that for an area overhead of less than 3%, a glitch power reduction of \( \sim 70\% \) is obtained (which is within 5% of the theoretical maximum), even with the larger conventional delay cell based delay line. This corresponds to a glitch filter with a resolution of 350 ps and maximum delay of 2.4 ns (this requires 3 bits for configuration of the delay-line). The total core dynamic power dissipation in this case is reduced by up to \( \sim 19\% \), and 8.4% on average over the benchmark set. Another candidate delay-line, with resolution of 550 ps and maximum delay of 1.6 ns (requiring 2 bits for configuration of the delay-line) offers slightly reduced glitch power reduction of just over 63%, but this is at an area overhead of under 2%, again for both delay-line types. However, the actual reduction in core dynamic power is 8.3% on average in this case (core dynamic power reductions reach up to \( \sim 20\% \)), thus making this combination of parameters particularly attractive given its low-cost and a net \( \sim 2\% \) decrease in dynamic power savings compared to the theoretical maximum! Interestingly, even though the first candidate glitch filter is able to offer \( \sim 7\% \) greater glitch power reduction than the second candidate
Figure 3.17: Max glitch power reduction vs. area overhead.

glitch filter, the increased power costs arising from a larger delay line results in the two glitch filter candidates offering nearly identical dynamic power savings on average. Detailed power reductions data for the two candidate glitch filters are provided in Table 3.3.

3.5.3 Tool Run-time

For the majority of the circuits in our benchmark set, the run-time of our CAD flow was dominated by the time spent in the glitch filter setting optimization phase where we search for an optimal solution to the MILP described in Section 3.4.2. Table 3.4 shows the run-time of the Gurobi Optimizer for a representative subset of the circuits in our benchmark set when targeting an architecture with ideal delay lines, as discussed in Section 3.5.1. The table also shows the number of variables, constraints, and the number of non-zero entries in the constraint matrix for the MILP associated with each circuit, which together indicate the complexity of each optimization problem. Naturally, a more complex MILP (more variables, constraints, and/or non-zero entries in the constraint matrix) will require greater run-time to compute an optimal (or near optimal) solution. Due to constraints on the allotted compute time for our experiments, we imposed a time limit of 21000 seconds (just under 6 hours) to the Gurobi Optimizer; if the tool was unable to find an optimal solution within this time limit it would return an incumbent solution while also reporting the gap between the cost of the incumbent solution and the estimated lower bound of the cost function. Therefore, Table 3.4 also shows the gap to the estimated lower bound for

\[ \text{minimize} \quad c^T x \]

subject to \[ A x \leq b \]

and \[ x \geq 0 \]

The matrix \( A \) is known as the constraint matrix.

The Gurobi Optimizer uses a branch-and-bound algorithm to search for optimal solutions, a common approach used by MILP solvers, and as such will seek a solution to the MILP over many iterations. At the end of each iteration, the solver produces an incumbent solution – i.e a legal solution that may be suboptimal – and also estimates a global lower bound of the cost function. The branch-and-bound algorithm seeks to find better incumbent solutions with each new iteration while also updating the estimated lower bound; the algorithm converges to a solution when the cost of the incumbent solution is equal to or is within an acceptable threshold of the estimated lower bound.
circuits whose optimal solution could not be found within 21000 seconds, which helps to understand the degree to which the optimizer is close to an optimal solution for each circuit. Observe that for the larger circuits shown in the table, such as jpeg, encoder and top, rs, decode, the MILP solver seem to be further away from their and optimal solutions after 21000 seconds than for smaller circuits such as alu4.

Table 3.4: Circuit-by-circuit run-time data for MILP solver.

<table>
<thead>
<tr>
<th>Circuit</th>
<th># Variables</th>
<th># Constraints</th>
<th># Non-zero Entries</th>
<th>Run Time (seconds)</th>
<th>Gap to lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>38002</td>
<td>17941</td>
<td>86150</td>
<td>21000</td>
<td>7.80%</td>
</tr>
<tr>
<td>apex2</td>
<td>50183</td>
<td>23529</td>
<td>116195</td>
<td>21000</td>
<td>37.84%</td>
</tr>
<tr>
<td>apex4</td>
<td>40510</td>
<td>19441</td>
<td>93056</td>
<td>21000</td>
<td>25.70%</td>
</tr>
<tr>
<td>bigkey</td>
<td>37779</td>
<td>18372</td>
<td>83952</td>
<td>48</td>
<td>0.00%</td>
</tr>
<tr>
<td>clma</td>
<td>107783</td>
<td>41998</td>
<td>282339</td>
<td>21000</td>
<td>10.82%</td>
</tr>
<tr>
<td>des</td>
<td>82644</td>
<td>39679</td>
<td>178476</td>
<td>21000</td>
<td>57.89%</td>
</tr>
<tr>
<td>diffeq</td>
<td>9989</td>
<td>3650</td>
<td>26232</td>
<td>751</td>
<td>0.01%</td>
</tr>
<tr>
<td>ex1010</td>
<td>149817</td>
<td>71963</td>
<td>344507</td>
<td>21000</td>
<td>63.88%</td>
</tr>
<tr>
<td>ex5p</td>
<td>48772</td>
<td>23334</td>
<td>108391</td>
<td>21000</td>
<td>53.30%</td>
</tr>
<tr>
<td>frisc</td>
<td>32129</td>
<td>11502</td>
<td>86316</td>
<td>21000</td>
<td>20.25%</td>
</tr>
<tr>
<td>jpeg, encoder</td>
<td>596827</td>
<td>247185</td>
<td>1501038</td>
<td>21000</td>
<td>98.17%</td>
</tr>
<tr>
<td>s298</td>
<td>29303</td>
<td>12965</td>
<td>71230</td>
<td>3552</td>
<td>0.01%</td>
</tr>
<tr>
<td>s38584,1</td>
<td>67822</td>
<td>30215</td>
<td>163769</td>
<td>21000</td>
<td>26.01%</td>
</tr>
<tr>
<td>seq</td>
<td>42971</td>
<td>20331</td>
<td>99087</td>
<td>21000</td>
<td>13.32%</td>
</tr>
<tr>
<td>top, rs, decode</td>
<td>214556</td>
<td>76220</td>
<td>540538</td>
<td>21003</td>
<td>96.22%</td>
</tr>
</tbody>
</table>

The table shows that while optimal solutions may be attained with relatively short run-times for a few circuits, the vast majority of the circuits shown require more than 6.5 hours to find optimal solutions, which may be unacceptable for a commercial CAD flow given the relatively small sizes of the circuits studied in this work. Naturally, an MILP-based CAD flow suffers from poor run-time as MILPs have been shown to be very difficult to solve, even for relatively small problem sizes [31]. The goal of this study was to identify an upper-bound on the power savings made possible by the proposed glitch reduction technique, and it is for that reason we developed an MILP-based CAD flow since we believed it would yield high-quality results, potentially at the cost of increased tool run-time. Therefore, run-time efficient heuristics would need to be developed to replace the MILP-based glitch optimization phase in a commercial implementation of our CAD flow; this is an absolute necessity for large commercial designs in the 100k-1M LE range, and we leave the development of a run-time efficient glitch optimization algorithm for future work. However, we may extend the utility of our proposed MILP-based glitch optimization through two optimizations, as discussed in the proceeding sections.

Early Solver Termination

While Table 3.4 has shown that the Gurobi Optimizer requires significant time to find optimal solutions for the circuits considered in this work, we observed that near optimal solutions could be obtained in much shorter spans of time. Therefore, a mechanism to which is able to terminate the solver after a desired level of glitch reduction is achieved may lead to more acceptable run-times. As a motivating example, Fig. 3.18 shows the progression of the incumbent solution’s normalized cost (normalized to the worst-case cost) and the estimated normalized lower bound of the optimal cost for the MILP associated with the apex2 circuit. The plot shows the progression of these two quantities over the first 10000 seconds as the optimizer seeks a solution to the MILP, as well as the cost of the final incumbent solution reported
Chapter 3. Glitch Reduction

by the optimizer after 21000 seconds. Observe that the final cost is \(\sim 75\%\) lower than the worst-case cost, which closely matches the simulated \(\sim 80\%\) reduction in glitch power observed for this circuit when the glitch filter settings corresponding to final incumbent solution\(^9\) are applied.

![Figure 3.18: Incumbent solution normalized cost and estimated lower bound of optimal cost (also normalized) versus time for \(\text{apex}^2\).](image)

If we assume the incumbent cost is a reliable predictor for the final post-simulated glitch power reduction, then we may estimate from the figure that the optimizer is able to find a solution which offers glitch power reduction within \(\sim 7\%\) of the maximum glitch reduction in under 10 seconds, a run-time value which is significantly more acceptable. Similarly, by monitoring the progression of the incumbent cost for the MILP associated with each circuit shown in Table 3.4 we may estimate the time taken to reach 50\% and 90\% of the maximum glitch power savings, which is shown in Table 3.5. Note that the geometric mean runtime to achieve 50\% and 90\% of the maximum glitch power reduction is 22.8 and 142.3 seconds, respectively, which is a significant run-time reduction compared to a geometric mean run-time of 11616.9 seconds for the data shown in Table 3.4. As such, we believe early solver termination may be a practical approach in reducing run-time of our MILP-based glitch optimizer. Moreover, users which demand the greatest levels of power optimization and can afford to incur run-time penalties may choose to allow the optimizer to run for longer periods of time to maximize glitch reduction.

Circuit Partitioning

Another optimization of the MILP-based glitch optimization phase of our CAD flow is to find independent sub-problems which comprise the larger MILP. For instance, consider the digital system shown in Fig. 3.19 which is comprised of four combinational networks \(N_1-N_4\). In the figure, we see that while a

\(^9\)As described in Section 3.4.2, the optimizer’s estimate of glitch power reduction is pessimistic by design, since the optimizer over-estimates the adverse-effects which arise from having unequal delay push-outs in the fan-in cone of a gate. The optimizer is also oblivious to the correlation between glitching in gates along the same path; for example it is not uncommon for a single glitch to propagate through multiple gates \([16]\), as such eliminating a glitch at an upstream gate may reduce glitching on downstream nodes, which is an effect the optimizer does not model. It is for these reasons that optimizer will underestimate power savings, although in this case the difference between the power estimate of the optimizer (i.e. the value of the cost function) and the post-simulation value is relatively small.
Table 3.5: Solver run-time when terminated early.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Time to reach 50% of Max Reduction (seconds)</th>
<th>Time to reach 90% of Max Reduction (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>apex2</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>apex4</td>
<td>8</td>
<td>1688</td>
</tr>
<tr>
<td>bigkey</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>clma</td>
<td>39</td>
<td>517</td>
</tr>
<tr>
<td>des</td>
<td>21</td>
<td>531</td>
</tr>
<tr>
<td>diffeq</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>ex1010</td>
<td>153</td>
<td>153</td>
</tr>
<tr>
<td>ex5p</td>
<td>15</td>
<td>6260</td>
</tr>
<tr>
<td>frisc</td>
<td>200</td>
<td>234</td>
</tr>
<tr>
<td>jpeg_encoder</td>
<td>70</td>
<td>2933</td>
</tr>
<tr>
<td>s298</td>
<td>10</td>
<td>47</td>
</tr>
<tr>
<td>s38584_1</td>
<td>23</td>
<td>526</td>
</tr>
<tr>
<td>seq</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>top_rs_decode</td>
<td>920</td>
<td>920</td>
</tr>
<tr>
<td>Geomean</td>
<td>22.8</td>
<td>142.3</td>
</tr>
</tbody>
</table>

connection from $N_4$ to $N_3$ exists, paths between $N_1$ or $N_2$ and any of the other networks must cross a flip-flop. If $N_5 = N_3 \cup N_4$, we may say that $N_1$, $N_2$, and $N_5$ are mutually disjoint combinational logic networks. Observe that glitch optimization problems corresponding to two mutually disjoint combinational networks are entirely independent of one another since the glitch characteristics and arrival times in one network are unaffected by glitch filter settings in the other, and vice-versa. Consequently, we may divide the circuit in Fig. 3.19(a) into three partitions, as shown in Fig. 3.19(b), and the glitch filter optimization problems for each partition may be solved independently.

In general, if a digital system may be partitioned into a set of mutually disjoint combinational networks, we may formulate independent MILPs for each combinational network, and solve the MILPs in parallel. Run-time improvements would arise from the fact that the sub-problems may be solved in parallel, and from the smaller problem sizes of each individual sub-problem.

### 3.6 Summary

This chapter presented a glitch reduction circuit and the associated CAD flow/optimization framework needed to maximize glitch power reduction in an FPGA architecture comprising the proposed circuitry. In contrast to prior works, the proposed circuitry offers the ability to reduce glitch power over a wide range of PVT corners because of the nature in which it suppresses glitches, while incurring minimal area/delay overheads. Variations of the proposed circuitry allow glitch power to be reduced from 60-71%, which corresponds to a reduction in core dynamic power of up to $\sim20\%$, and $\sim8.4\%$ on average at an area cost of 1.5-3%.

---

$^{10}$Even if the problems cannot be solved in parallel (for instance, because of hardware constraints), if an MILP is decomposed into $n$ independent sub-problems and these are solved serially, we would still anticipate a run-time improvement since there is a super-linear relationship between an MILP's runtime and the number of constraints and variables forming the MILP.
Figure 3.19: Digital circuits may be partitioned to yield multiple independent combinational circuits whose corresponding MILPs may be solved independently.
Chapter 4

Leveraging Unused Resources for Energy Optimization of FPGA Interconnect

4.1 Introduction

Process scaling has served as the linchpin of the unprecedented growth the semiconductor industry has seen over the past 50 years. Through process scaling, the performance, energy efficiency, and density of a digital system can all be increased, and these trends combine to offer significant increases in the total computational power of chips with each passing process node. However, as the industry continues to venture into deep submicron territory, several challenges have emerged which are beginning to degrade the expected returns from process scaling. One of the most significant bottlenecks has been the poor scalability of interconnect power; for example, [75] and [56] show that interconnect is becoming an ever more dominant factor in the performance and power dissipation of a digital system. As such, the optimization of interconnect is becoming an ever more critical aspect of digital system design.

In this chapter, we present techniques to reduce static and dynamic power dissipated in the interconnect of FPGAs, by taking advantage of unused resources. As described in Chapter 2, the bulk of prior work which addresses the issue of power consumption of unused resources in FPGAs have specifically targeted static power reductions via variations of coarse-grained power gating strategies [20, 35, 95], which are costly from an area point-of-view. As previously discussed, coarse-grained power reduction strategies are ultimately limited in the amount of power that can be saved due to their inherent inflexibility. In this chapter, we propose two fine-grained techniques to reduce the power consumption in the routing networks of island-style FPGAs. This chapter begins by showing that the routing networks of island-style FPGAs are typically heavily underutilized, leaving a large number of unused routing resources which needlessly dissipate leakage power and increase the dynamic power of neighbouring used conductors due to increased coupling capacitance. In this chapter, we propose to leverage the unused routing conductors in a design to strategically reduce both dynamic and static power dissipated in the interconnect. The first technique seeks to minimize the total parasitic capacitance of FPGA routing conductors. We are motivated by the observation that the majority of the capacitance in the interconnect lies in the coupling capacitance, $C_C$, between adjacent routing conductors, and moreover, it is expected that $C_C$ will become an even more dominant contributor to total wire capacitance in newer process nodes [49]. We observe that if an unused routing conductor is left in a floating state, the adjacent routing conductors will see...
reduced capacitive loading. This is because (as we will demonstrate) the effective capacitance seen by an adjacent conductor is that of the series combination of the coupling capacitance between the two conductors, and the total capacitance to ground of the conductor left in a floating state. Since the series combination of two capacitors results in an equivalent capacitance which is less than the capacitance of either of the two original capacitors, it follows that the total capacitance seen between a conductor and an adjacent floating conductor is less than in the case where the adjacent conductor is tied to a rail (\(V_{DD}\) or \(GND\)). To implement this desired capacitance reduction technique, we propose a lightweight buffer topology which allows for tri-state operation, while requiring minimal area overhead. We propose to augment a conventional FPGA routing architecture with these tri-state buffers.

The second power reduction technique follows from an analysis of the leakage currents present in the interconnect routing multiplexers. Our analysis shows that by ensuring certain critical pins of a routing multiplexer are unused (and thus can be tri-stated in our proposed routing architecture), significant leakage power dissipated in routing multiplexers can be reduced. We further propose CAD techniques to strategically encourage routing conductors to be left unused if they are adjacent to conductors used by power-critical nets (and thus can be tri-stated) to reduce coupling capacitance, or if they are connected to the specific pins of routing multiplexers which are expected to be leaky. Results show that for the circuits considered in this study, average interconnect dynamic power reductions of greater than \(\sim 25\%\) can be achieved as \(C_C\) approaches a value three times as large as \(C_P\) (the plate capacitance – the capacitance between a wire and adjacent metal layers/substrate). In addition, our experiments show that that, on average, between \(\sim 34.8-81\%\) of static power may be saved in the routing network for the circuits considered in this work, which combined with the aforementioned dynamic power savings leads to total energy savings in the routing network of between 14.9\% to 42.7\%. These savings are achieved at a critical path degradation of between \(\sim 1.2-1.8\%\) (on average), and a total area overhead of between \(\sim 2.6-4.9\%\), depending on the tri-state buffer topology employed.

4.2 Background

4.2.1 Interconnect Dynamic Power and Capacitance Scaling

Due to the large capacitances of the routing wires in a modern FPGA, the dynamic power dissipated in the routing network is a dominant component of the total dynamic power dissipated, between 50-60\% \[74,95\]. As such, any effort to reduce the power consumption in the routing network will potentially allow for an appreciable reduction in overall power.

Fig. 4.1 shows the two primary sources of capacitance in a routing conductor: \(C_P\), the plate capacitance, which is the capacitance between a routing conductor and conductors in adjacent layers and/or the substrate, and \(C_C\), the capacitance between a routing conductor and adjacent routing conductors within the same metal layer. Due to the evolution of the parameters related to wire geometries through process scaling, wires have considerably greater height than width, as depicted in the figure. This leads to a growing disparity in the capacitances of \(C_C\) and \(C_P\). While the specific ratio between \(C_C\) and \(C_P\) will vary from one foundry to the next, there are several estimates for this ratio in literature that shed light on the dominance of \(C_C\) over \(C_P\). ITRS projections indicate that for modern processes, the ratio between \(C_C\) and \(C_P\) is approximately equal to 2 \[49\], although prior work has estimated that \(C_C/C_P\) is over 5.7 in a 90nm process \[90\]. Moreover, in FPGAs specifically, it is believed that \(C_C\) can be up to three times larger than \(C_P\), depending on wire width and spacing \[9\]. Thus, it is widely acknowledged that \(C_C\) is the dominant component of the total capacitance of interconnect, and as such, any technique
4.2.2 Leakage Power in FPGA Interconnect

Previous studies which have investigated the leakage power dissipated in FPGAs have shown that combined, the leakage dissipated in routing switches and in configuration memory accounts for over two-thirds of the total leakage of an FPGA [95]. However, given the fact that performance of an FPGA is effectively independent of the performance characteristics of the configuration memory, the transistors within the configuration memory can be optimized to reduce leakage (by increasing transistor gate length, employing high-$V_T$ transistors, etc.). In doing so, routing switch leakage becomes the dominant contributor to the total leakage in the FPGA [92].

Fig. 4.2 illustrates the leakage paths in a routing switch comprising a two-stage routing multiplexer having 16 inputs. This particular multiplexer topology has been shown to offer the best area-delay trade-off [25] and is used by Intel in their FPGAs [67], as such we assume all routing switches comprise such a multiplexer topology in this work. As shown in the figure, leakage paths exist between different inputs of the routing multiplexer, between the inputs of the multiplexer and the feedback PMOS weak keeper of the output stage, and between the supply rails of the output buffer. While transistors have multiple sources of leakage, our experiments with the commercial 65 nm models used in this work indicate that sub-threshold leakage dominates over all other forms of leakage, and this holds true for newer process nodes which use high-K gate dielectrics [15], allowing for drastic reduction in gate leakage. As such for this work, we primarily seek to optimize sub-threshold leakage. Alternate routing multiplexer structures, such as that depicted in [53] employ a fully decoded first stage multiplexer cascaded with an encoded second stage; this multiplexer will have similar leakage paths to the structure depicted in Fig. 4.2 and thus the leakage reductions proposed in this work will likely yield similar results with this alternative structure. In addition, full CMOS multiplexers are topologically similar to the NMOS-based topology studied in this work [26], as such we expect similar levels of leakage reduction using our proposed technique if FPGA vendors begin adoption of full CMOS multiplexers in their products. A detailed study of the proposed power reduction technique with alternative routing multiplexer structures is left for future work.

Given the dominance of sub-threshold leakage on the total leakage of the routing multiplexer, we can observe that for a given setting of the configuration bits of the multiplexer, the leakage paths are
predominantly through the selected inputs of the multiplexer (i.e. the pass transistors in the multiplexer that are on), since any other leakage path in the multiplexer would traverse at least two off transistors, and paths which traverse two or more off transistors are known to have significantly lower leakage than paths which traverse a single off transistor [85]. As we will show in Section 4.3.2, we leverage this property to reduce leakage in this work.

4.2.3 Wire Planning in ASICs and FPGAs

Due to the increasing sensitivity of key system-level metrics (such as speed and power) on interconnect parameters, much work has been done on interconnect optimization in the ASIC domain (e.g. [38, 79]). Broadly speaking, the interconnect optimization problem serves to optimize a chip’s speed and/or power consumption through judicious selection of wire width and spacing, under constraints of total wiring area. For example, a timing-critical net would be routed with an intermediate wire width (to reduce interconnect resistance and thus reduce interconnect delay) and increased wire spacing to reduce wire capacitance (and thus further reduce delay). On the other hand, a high activity signal would be routed with both minimum width and maximum spacing to ensure minimum routing conductor capacitance, thus minimizing dynamic power consumption.

Previous work has investigated the optimization of interconnect in FPGAs [17], however it only considered the optimization of wire spacing and width of the prefabricated routing network. These optimizations provided some control over the wire width and spacing of conductors used by a particular net. For example, if a net were timing critical, the router would endeavour to route the net using the portion of the interconnect network which had favorable wire width/spacing. However, depending on placement and wiring congestion, such low-capacitance conductors may not be readily accessible, and moreover, the prior work considered only timing optimization, not power.

4.2.4 Routing Resource Utilization in FPGAs

FPGA vendors develop FPGA families (such as Stratix from Altera or Virtex from Xilinx) which are a set of different sized FPGAs all having the same fundamental routing and logic architecture, but have different numbers of complex logic blocks (CLBs), hard blocks, I/O capabilities, etc. While the different members of a family may have vastly differing logic capacities (for instance, the difference in logic capacity varies by an order of magnitude between the smallest and largest members of the Stratix
Chapter 4. Leveraging Unused Resources for Energy Optimization of FPGA Interconnect

X family), their routing architecture, and thus channel width remains constant. This is a practical approach since it allows a vendor to allocate design resources to develop a single tile (for example containing a single CLB and associated routing channels), and then stitch a varying number of these tiles together to form the different members of each family. However, given the need to target a wide range of applications, vendors typically have to architect an FPGA’s interconnect network to ensure worst-case routing problems can be accommodated. This leads to the routing network containing many more wires than is actually needed for many designs. Moreover, the channel width of an FPGA is dictated by the worst-case congestion present on the chip, which is typically not predictable and can deviate significantly from one region to the next \[18\]. This also leads to significant variation in the utilization of routing resources across the chip. Fig. 4.3 shows a distribution of an FPGAs individual channel utilizations, averaged over the VTR benchmarks \[74\], shown when applications are targeted to an architecture whose channel width is 1.3 times larger than the minimum channel width required to route each design (i.e. for a “low-stress” routing case). The figure shows significant under utilization of routing resources, as the mean channel utilization over the benchmarks is \(\sim 28\%\).

In addition to the speed and area consequences arising from conservatively setting the channel width of FPGAs, there are power implications as well. In the prefabricated routing network of FPGAs, wire spacing between adjacent conductors is proportional to the ratio between the physical width of the channel (as dictated by tile layout) and the number of conductors occupying that channel (i.e. the channel width). As will be discussed in the next section, the capacitance of routing conductors in modern nanoscale processes is dominated by the capacitance between adjacent routing conductors, which is inversely proportional to wire spacing. Consequently, incorporating a large number of conductors per channel will necessitate reduced wire spacing, and thereby higher capacitance and power. Moreover, since routing static power is directly proportional to the number of interconnect buffers present in the array, a conservatively architected routing network invariably leads to excess static power consumption as well.

### 4.3 Power Optimization Using Tri-State Buffers

#### 4.3.1 Dynamic Power Optimization

The proposed parasitic capacitance reduction technique is inspired by the observation that two capacitors connected in series results in a total capacitance that is smaller than either of the two capacitors. More specifically, if two capacitors, \(C_1\) and \(C_2\) are connected in series, the total resulting capacitance is \(C_{eq} = (C_1 \cdot C_2)/(C_1 + C_2)\). We aim to reduce coupling capacitance by combining this capacitance with a second capacitance in series.

Fig. 4.4(a) shows three routing conductors, with their respective driver circuits. Routing conductor 1 is coupled with routing conductor 2 through coupling capacitor \(C_{C1}\) and similarly, routing conductor 2 is coupled with routing conductor 3 through \(C_{C2}\). From the perspective of routing conductor 1, routing conductor 2 and coupling capacitors \(C_{C1}\) and \(C_{C2}\) together form the equivalent, approximated circuit shown in Fig. 4.4(b). In the figure, \(R_{eq}\) is the effective output resistance of the driver driving routing conductor 2, and it is assumed (pessimistically, as will later become apparent) that the effective driver resistance for the buffer driving routing conductor 3 is near 0 (the conductor is grounded). The input
impedance of this circuit has the following $s$-domain transfer function:

$$Z_{in}(s) = \frac{R_{eq}(2 \cdot C_C + C_P)s + 1}{C_C s[R_{eq}(C_C + C_P)s + 1]}$$

(4.1)

where it is assumed that $C_{C1} = C_{C2} = C_C$. While the effective impedance is frequency dependent, we note that if $R_{eq} >> T_{PW,AVG}/2\pi (C_C + C_P)$, where $T_{PW,AVG}$ represents the average signal pulse width, $Z_{in}(s) \approx (2 \cdot C_C + C_P)/(C_C(C_C + C_P))$. This implies that if the driver impedance can be set to be very large, the effective impedance seen from routing conductor 1 looking towards routing conductor 2 is that of capacitor $C_{C1}$ in series with the parallel combination of capacitors $C_{C2}$ and $C_P$, which means that the effective parasitic loading on routing conductor 1 can therefore be reduced. Permanently setting the driver resistance to be large enough to realize reductions in effective parasitic capacitance is not a practical solution, as this will make the buffer excessively slow. Note however that a buffer with a reasonably small effective output resistance can be made to have a much larger output resistance if that buffer can be put into a tri-state mode. Putting a buffer into tri-state mode requires disconnecting all paths to either $V_{DD}$ or $GND$, which ensures that there are no low impedance paths to either rail.

In this work, we assume that a routing conductor can in general be put into a tri-state mode only if it is unused. Parasitic capacitance reduction can be observed only when an active routing conductor is adjacent to one or more unused routing conductors. The observed power reductions are therefore a function of how a specific design has been routed, and in Section 4.6.2 we describe enhancements to a conventional FPGA router to encourage leaving unused conductors adjacent to used conductors, where possible.

Returning now to routing conductor 3 in Fig. 4.4(a), the analysis above assumed the conductor was grounded. The reason this is a pessimistic assumption as far as capacitance reduction is concerned is as follows: if instead conductor 3 were tri-stated, the consequence would be that capacitor $C_{C2}$ would be in series with the plate capacitance of conductor 3 ($C_P$). This would mean that the value of the capacitor labeled $C_{C2} + C_P$ in Fig. 4.4(b) would in fact be $C_P + C_{C2}[C_P/(C_{C2} + C_P)]$ which is less than $C_{C2} + C_P$. This would provide a further reduction in coupling capacitance between conductor 1
and conductor 2 (for the case of conductor 2 being tri-stated). Fig. 4.5 illustrates a metal cross section showing a used conductor $i$, with unused adjacent conductors which we call the first-level neighbours of $i$—these are the conductors immediately adjacent to $i$. Adjacent to the first-level neighbours of $i$ are the second-level neighbours (i.e. not immediately adjacent to $i$), adjacent to these conductors are the third-level neighbours of $i$, and so forth. In light of the previous discussion, the capacitance seen from $i$ looking towards its first-level neighbours is minimized if all first-level to $n$th-level neighbours are unused and tri-stated. However, any attempt to ensure all first to $n$th-level neighbours are unused may require considerable router complexity, and so we wish to determine the value of optimizing the occupancies of conductors within $n$ levels of $i$.

The trade-off between router effort and capacitance reduction that arises from the number of levels that are tri-stated can be investigated analytically. Let $C_{\text{eff}}^n$ represent the effective capacitance seen by $i$ when all first-level to $n$th-level neighbours are tri-stated, while all $(n + 1)$st-level to $\infty$-level neighbours are not tri-stated, normalized to $C_P$. In this chapter, we refer to $C_{\text{eff}}^n$ as an $n$th-order effective capacitance model. It can be shown with simple circuit analysis that $C_{\text{eff}}^1$ is given by:

$$C_{\text{eff}}^1 = \frac{\rho C(2 + \rho C)}{2(\rho C + 1)},$$

(4.2)

where $\rho C$ is the coupling factor and is equal to $C_C/C_P$. Similarly, it can be shown that $C_{\text{eff}}^2$ is given by:

$$C_{\text{eff}}^2 = \frac{\rho C(\rho C^2 + 6\rho C + 4)}{(\rho C + 2)(3\rho C + 2)},$$

(4.3)

In fact, this analysis can be extended to find the limit as $n \to \infty$, as this is an instance of the well-known semi-infinite resistor ladder problem [29]:

$$C_{\text{eff}}^\infty = \sqrt{1 + 2\rho C} - 1.$$  

(4.4)
This expression represents the lower-bound on the effective capacitance of conductor $i$. The three expressions shown above are plotted for different values of $\rho C$ in Fig. 4.6.

As $\rho C$ ranges from 1.5 to 3, we see that while there is a significant difference between $C_{\text{eff}}^1$ and $C_{\text{eff}}^\infty$, the difference between $C_{\text{eff}}^2$ and $C_{\text{eff}}^\infty$ is relatively small over this range of $\rho C$ values. As such, for this work, we deem it sufficient to only minimize the occupancy of the first-level and second-level neighbours of conductor $i$ if this conductor is used by timing and/or power critical nets, since any attempt to optimize the occupancy of neighbours at other levels will yield diminishing capacitance reductions, while requiring substantially greater computational effort.

### 4.3.2 Static Power Optimization

The use of tri-state buffers in the interconnect of FPGAs can also allow for reduction in leakage power. As discussed in Section 4.2.2 there are two dominant sources of leakage in FPGA interconnect: routing buffer leakage and leakage in the routing multiplexers. The use of tri-state buffers allows for a reduction of leakage current for both of these paths. It is trivial to show that the former leakage path can be reduced/eliminated using tri-state buffers since when a buffer operates in tri-state mode, both the pull-up and pull-down paths of the buffer are simultaneously shut-off, ensuring that all low impedance paths to either supply rail are eliminated.

Fig. 4.7 shows how leakage through the second (and more dominant) set of paths may also be reduced with the use of tri-state buffers. The figure shows a routing multiplexer $S_1$ (which is a 2:1 multiplexer for simplicity) whose selected input is driven by buffer $B_1$, and whose unselected input is driven by an unused buffer $B_2$, which in this case is tri-stated. Conventionally, the unused buffer $B_2$ would be programmed to drive its output to some predetermined state to reduce leakage. We wish to show that by tri-stating unused buffers, we reduce the amount of leakage through a routing multiplexer versus the case where unused buffers are driven to some predetermined state. This optimal predetermined state varies from one process to the next [13,92]; for this analysis we assume that the optimal state is logic-"1", however, it can be shown that the arguments presented here also apply if the optimal state for leakage reduction in a conventional architecture is logic-"0".

In Fig. 4.7(a) $B_1$’s input is at $V_{DD}$ and thus $V_X = V_Y = 0$. Since $B_2$ is tri-stated, $V_Z < V_{DD}$. This then means that the voltage across $M_6$ is equal to $V_Z - V_Y < V_{DD}$; we therefore conclude that because of Drain Induced Barrier Lowering (DIBL) effects, the leakage current through $M_6$ is less than...
Figure 4.6: Capacitance reduction vs. $\rho_C$ for different levels of modelling accuracy.

Figure 4.7: Leakage through routing multiplexer when unselected input tri-stated.

in a conventional architecture where, as we have assumed, $V_Z = V_{DD}$. Depending on the leakage characteristics of $M_3$, significant leakage reduction may be obtained versus the baseline case.
In Fig. 4.7(b) $B_1$’s input is connected to GND and thus $V_X = V_{DD}$ while $V_Y \approx V_{DD} - V_{in}$, where $V_{in}$ is the threshold voltage of $M_6$. Since $B_2$ is tri-stated, $V_Z > 0$. This then means that the gate-source voltage of $M_6$ is equal to $-V_Z$ volts; if $V_Z$ is greater than the sub-threshold swing of the device (typically < 100mV), then the leakage current $i_1 \approx 0$. While the leakage current in this case is marginally larger than in the conventional case (when $B_2$’s output is $V_{DD}$), the difference in leakage between the two cases is insignificant for all practical purposes. Moreover, the difference is markedly outweighed by the leakage reduction in the case where the active buffer drives the select input pin of $S_1$ to logic-“0”. Therefore, by tri-stating unused buffers, the average leakage through routing multiplexers can be significantly reduced. In addition, given that the leakage in either state as shown in this example is dictated by the leakage characteristics of the tri-state buffer, we can reduce the multiplexer’s sub-threshold leakage by close to an order of magnitude by optimizing the leakage of the tri-state buffer. In Section 4.4, we show how such a low-leakage tri-state buffer may be designed in the context of FPGAs with a minimal area/delay penalty.

We now consider, for a used routing multiplexer configured to select a certain input pin $in_i$, the set of input pins which lie along the most significant leakage paths of the routing multiplexer. We call this set of pins the leaky pins of $in_i$. Referring again to the 16:1 routing multiplexer shown in Fig. 4.2 we observe that the routing multiplexer consists of four first-stage multiplexers $X_1 - X_4$ and a single second-stage multiplexer $X_5$. If $in_1$ (which resides in MUX $X_1$) is used, then clearly all other inputs in $X_1$ ($in_2 - in_4$) belong to the set of leaky pins of $in_1$. Moreover, all inputs sharing the same configuration bit as $in_1$ will also be a significant contributor to the total leakage of the routing multiplexer. In this case, this corresponds to inputs $in_3$, $in_5$, and $in_{13}$. In general, if input $in_i$ is used, the set of leaky pins of $in_i$ includes all inputs residing in the same first stage multiplexer as $in_i$, as well as all inputs which share the same configuration bit as $in_i$. We can therefore attempt to optimize routing leakage by ensuring that when some net $n_j$ attempts to use routing conductor $r_k$ through pin $in_i$ of $r_k$’s routing multiplexer, the set of leaky pins of the input pin $in_i$ are unused and thus can be put into a low leakage state. This ensures that the dominant leakage paths of $r_k$’s routing multiplexer are optimized. Furthermore, if the unused input pins of the routing multiplexer (specifically the unused leaky pins corresponding to selected pin $in_i$) are tri-stated, the leakage through the routing multiplexer can be further reduced. This technique therefore allows us to achieve fine-grained power gating, since we are able to shut-off specific leakage paths (the most critical ones). We describe router enhancements for multiplexer leakage reduction in Section 4.6.3

### 4.4 Tri-state Buffers

Two conventional tri-state buffers are shown in Fig. 4.8 (Fig. 4.8a) shows a buffer with a header transistor which has been added to cut off all paths from $V_{DD}$ to the input and output stages of the buffer when input signal $TS$ is high. Analysis of this circuit indicates that when $TS$ is low, the circuit operates as a conventional buffer. On the other hand when $TS$ is high, assuming that the input is held at logic-“1” by the half latch formed by $M_2$ and $M_6$, both $M_4$ and $M_5$ are simultaneously shut off, thus allowing the output to be tri-stated. In this work, we refer to this topology as a Supply-Gated Tri-state Buffer (SG-TSB), since tri-state operation is achieved by effectively severing the connection to the positive supply rail. This conventional approach to designing tri-stateable buffers is costly because of the transistor stacking in the pull-up paths. To maintain the same buffer delay as that of a conventional buffer, the transistor widths need to be increased by at least a factor of two, and as such this topology...
suffers from significant area costs.

An alternative conventional tri-stateable buffer topology is shown in Fig. 4.8(b) - this topology consists of a split inverter (formed by transistors $M_1$ and $M_3$), separated by a transmission gate (transistors $M_2$ and $M_5$) which together form the input stage, and a conventional output stage consisting of transistors $M_7$ and $M_8$. Note that in this topology, there is no transistor stacking in the output stage, which is a key design feature enabling tri-state operation with drastically reduced area overhead. The buffer works as follows: when $TS$ is low, the transmission gate consisting of transistors $M_2$ and $M_5$ is on which effectively shorts the gate of $M_7$ to the gate of $M_8$. Thus, in this mode $M_1$ and $M_3$ are connected to form a single inverter which directly drives the output stage. When $TS$ is high, $M_2$ and $M_5$ are both off, which leads to the connection between the gates of $M_7$ and $M_8$ being severed; $M_4$ and $M_6$ are both turned on pulling the gates of $M_7$ and $M_8$ to GND and $V_{DD}$, respectively. Thus, in this mode both $M_7$ and $M_8$ are off, leaving the output tri-stated. This topology therefore uses a clever arrangement of transistors in the input stage of the buffer to ensure that tri-state mode is achieved without transistor stacking in the output stage, which results in area savings compared to the SG-TSB discussed previously. For the remainder of this chapter, we refer to the circuit shown in Fig. 4.8(b) as an Input-Gated Tri-state Buffer (IG-TSB), as this circuit operates in tri-state mode by disconnecting the output stage from the input stage, and ensuring the appropriate voltages are applied to the transistors forming the output stage.

We proposed to further optimize the IG-TSB topology for use in FPGAs. Referring to Fig. 4.8(b), we can note some potential optimizations. We first observe that in FPGAs, an unused routing buffer will always have its input set to $V_{DD}$; this means that in tri-state mode, $M_1$ is always guaranteed to be on, which makes transistor $M_6$ unnecessary. Furthermore, because the input is guaranteed to be set to $V_{DD}$, the drain of $M_3$ can safely be connected directly to the gate of $M_7$; thus $M_1$ and $M_3$ would form a conventional inverter driving the gate of $M_7$. This is advantageous because previously the path to $V_{DD}$ from the gate of $M_7$ was through two stacked PMOS transistors: $M_2$ and $M_5$. By eliminating this stack of transistors, we are able to reduce the sizes of the PMOS transistors in the input stage, which results in appreciable area reduction since the area of PMOS transistors dominates the total area.
of the input stage. Finally, transistor $M_5$ can be repositioned to have its source connected to $V_{DD}$ and its drain connected to the gate of $M_8$ – again this eliminates any stacking in the pull-up path from the gate of $M_8$ to $V_{DD}$, and also marginally reduces the parasitic capacitance connected to the gate of $M_7$ (we have eliminated the diffusion capacitance arising from the drain of $M_5$ at this node). Our optimized tri-stateable buffer topology is shown in Fig. 4.9(a), and together with the set of optimizations we discussed above, aims to offer a very area efficient tri-stateable buffer topology.

A detailed description of our proposed buffer is as follows: When the buffer is used (i.e. it is not tri-stated), since $TS$ is low, transistor $M_4$ is off, while $M_2$ is on. During a rising transition at the input, $M_1$ turns on, and the gates of $M_7$ and $M_8$ are both pulled to ground, which turns $M_7$ off, $M_8$ on, and allows the output to be pulled to $V_{DD}$. During a falling transition at the input, $M_3$ and $M_5$ both turn on, which results in $M_7$ turning on and $M_8$ shutting off, thus causing the output to be pulled to $GND$.

When operating in tri-state mode, $TS$ is high and therefore, $M_2$ is off, $M_4$ is on, which keeps the gate of $M_8$ pulled to $V_{DD}$ (thus keeping it turned off). Since an unused buffer has its input tied to $V_{DD}$ (the input stage of the buffer acts as a latch when the input is at $V_{DD}$, since $M_9$ forms a feedback loop allowing the input to hold its state at $V_{DD}$), $M_1$ turns on, thus pulling the gate of $M_7$ to $GND$ and turning it off. For the remainder of this chapter, we refer to the proposed circuit as a PMOS-Gated
Tri-state Buffer (PG-TSB), since only the PMOS transistor of the output stage (transistor $M_8$) is truly disconnected from the input stage – and thus gated – while the NMOS transistor of the output stage ($M_7$) is shut off by virtue of node voltage conditions which can be guaranteed for an unused routing buffer.

The three tri-stateable buffer topologies presented thus far all offer the benefit of buffer leakage reduction when operating in tri-state mode. However, further gains in leakage reduction can be made with simple circuit optimizations, which we propose to complement the routing multiplexer leakage reductions discussed previously. We therefore propose an additional variant of the proposed tri-state buffer which includes leakage reducing optimizations. In this work, we refer to this proposed low power variant as a Leakage Optimized PMOS-GATED Tri-state Buffer (LOPG-TSB). Typically, the leakage in the routing network is dominated by leakage in the routing multiplexers [92], and as such any effort to optimize buffer leakage would have limited impact on total routing network leakage in a conventional FPGA. However, with a reduction in routing multiplexer leakage – as we propose to accomplish in this chapter with the power reduction strategy outlined in the previous section – buffer leakage will emerge as a more significant contributor to total routing leakage. As such, in the context of this work, it is beneficial to seek means to further reduce buffer leakage; we therefore propose an additional variant of the proposed tri-state buffer which includes leakage reducing optimizations.

To optimize the leakage of the proposed PG-TSB, we first observe that both the size and leakage of the buffer are dominated by the output stage, which is natural to expect given that the output stage needs to drive a considerably larger load than the input stage. Therefore, the greatest gains in leakage reduction will be enabled through leakage optimizations which target the output stage. Common circuit-level leakage reduction techniques – such as the use of High Threshold Voltage (HVT) transistors – inevitably come with an inherent performance penalty which may be compensated for through an increase in transistor sizes, however this is undesirable since this would increase the size of the output stage which, as we have previously established, will lead to a noticeable increase in the total area of the buffer. An alternative means to mitigate the performance penalty of using HVT transistors (and/or other circuit-level leakage reduction techniques) would be to apply a boosted voltage at the gate of all active low-leakage transistors. Figs. 4.10(a) and 4.10(b) illustrates this concept, where we have an HVT NMOS transistor ($M_2$) and a Standard Threshold Voltage (SVT) PMOS transistor ($M_1$) that form an inverter, which is connected to a voltage supply operating at $V_{DD}$ volts. Assume for this example that the HVT NMOS transistor has a threshold voltage of $V_{tnom} + \Delta V_t$, where $V_{tnom}$ is the magnitude of the nominal threshold of an SVT transistor. First consider the leakage and drive current characteristics of the inverter when the input is 0 volts. In this case, the NMOS transistor is off and in a low-leakage state due to its increased threshold voltage, while the PMOS transistor is on. Observe that because the PMOS transistor has a nominal threshold voltage, it does not suffer from degraded drive current. Next, we consider the leakage and drive current characteristics when the input is $V_{DD}$ volts. Now, the NMOS transistor is on, and given that $V_{DDB} \geq V_{DD} + \Delta V_t$, its overdrive voltage and drive current will be equal to or greater than an SVT NMOS transistor. Interestingly, the PMOS transistor $M_1$ in this case is off and operating in a low leakage state because its source-to-gate voltage is negative. In fact, if $V_{DDB} = V_{DD} + \Delta V_t$, then the overdrive voltage of $M_1$ is equal to $V_{SG} - V_{tnom} = -(V_{tnom} + \Delta V_t)$. In other words, $M_1$ would exist in a low-leakage state, similar to an HVT transistor with a threshold voltage of $V_{tnom} + \Delta V_t$. As such, this particular circuit configuration presents an attractive means to

\[\text{The increased } V_{DDB} \text{ will increase gate leakage, however the overall impact to total transistor leakage would be}\]
reduce leakage of the buffer without incurring significant performance penalties.

Under normal circumstances, the cost of a second supply rail may be of some concern, however, commercial FPGAs typically already have an additional supply rail used to boost the gate voltages of NMOS pass transistors to improve performance \[26\]. We propose to connect the first stage of our proposed buffer to a rail carrying this boosted voltage \((V_{DDB})\), while the second stage is connected to a rail at the nominal supply voltage \((V_{DD})\); Fig. 4.9(b) shows the proposed LOPG-TSB. It is worth noting that traditionally the boosted rail in an FPGA is meant to support static signals and thus has a relatively low current requirement. Using the boosted supply in the manner we have proposed will increase the current demand on this rail and thus will require that the power distribution network associated with this supply is adequately bolstered. However, it is very likely that the resulting current demand on the boosted rail will remain orders of magnitude smaller than the demand on a conventional supply (i.e. \(V_{DD}\)) given the relatively small dynamic load (the internal capacitance of the routing buffer) that will need to be driven. As such we anticipate the resulting additional overhead will be acceptable, although future work will need to be conducted to investigate this issue in detail. Another issue is the potential increased leakage in the input stage of the buffer due to the fact that the input to the buffer will swing between 0 and \(V_{DD}\) volts, while the supply connected to the input stage is at \(V_{DDB}\) volts. Using a single PMOS transistor for the pull up network of the input stage in this situation is potentially hazardous because the source-gate voltage of the device will be \(V_{DDB} - V_{DD}\) volts instead of 0 volts when the input is a logic-“1”. However this may be compensated for by either employing a stack of PMOS transistors in the pull-up network, or by employing a single PMOS transistor with increased gate length and/or threshold voltage. In this work, we used a PMOS stack to combat this issue, as shown in Fig. 4.9(b) \((M_{S1} \text{ and } M_{S2} \text{ are used to build PMOS stacks})\). Fig. 4.11 shows the leakage dependence of an inverter in such a scenario as the boost factor \((V_{DDB}/V_{DD})\) is varied. The figure shows that while an exponential increase in leakage is to be expected as the boost factor is increased, using a stack of two transistors is sufficient to realize acceptable leakage levels (i.e. better than or equal to baseline leakage) for boost factors greater than 1. We may subsequently increase the gate width of the transistors \(M_{S1}, M_3 \text{ and } M_5\) to overcome any performance penalty arising from increased gate length/threshold voltage, and given that this transistor is in the first stage of the buffer, this will have a minimal impact on total buffer area.

---

[15] insignificant given that it is dominated by other sources. This is especially true in modern processes which have High-k gate dielectrics [15].
The proposed LOPG-TSB is therefore topologically similar to the PG-TSB shown in Fig. 4.9(a); the leakage reductions are obtained by ensuring \( M_f \) is an HVT transistor, the input stage is connected to a boosted supply voltage \( V_{DDB} \), and PMOS transistors stacks are used to optimize leakage in the face of a non-zero \( V_{SG} \) when the input is at logic-"1". These optimizations, when combined, lead to a drastically reduced leakage, as will be detailed in the following section, with minimal delay and area overheads. For the remainder of this work, we assume \( V_{DDB} \) is 10% higher than \( V_{DD} \), in accordance with [26].

Finally, there is one issue regarding the tri-state buffers that warrants additional discussion, which is regarding the feasibility of “floating” unused conductors. This practice is acceptable in the FPGA context because such unused conductors feed downstream switches and pins via input NMOS-based multiplexers (see Fig. 2.3). Naturally, the select inputs of such downstream multiplexers would not be configured to pass a signal from an incoming unused conductor. Note that the ability to tri-state wires is unique to the inherent interconnect architecture of FPGAs; tri-stating signals in general CMOS logic is not acceptable, owing to the potential for significant short-circuit current if a CMOS gate’s input floats to a voltage mid-way between the two rails.

### 4.4.1 Buffer Comparison

To assess the merits of the proposed tri-state buffers over the two conventional topologies, we sized the buffers to allow for a target delay of approximately 185 ps - representing a ~10% delay overhead over a conventional buffer (not tri-stateable). Comparisons of the area overhead (in minimum transistor widths) and tri-state leakage power reduction (relative to a conventional, non-tri-stateable buffer) are shown in Table 4.1.

The comparison shows that the conventional SG-TSB topology (Fig. 4.8(a)) offers superior leakage power reductions when operating in a tri-state mode compared to the IG-TSB and PG-TSB topologies. This is expected since the SG-TSB topology allows leakage to be reduced in both the input and output stages of the buffer, since the path from each buffer to \( V_{DD} \) is severed when in tri-state mode. However, the SG-TSB topology suffers from increased area overhead and active leakage because of the larger PMOS transistors employed in the buffer. The IG-TSB topology (Fig. 4.8(b)) offers dramatic reductions in area overhead, owing to the fact that there are no stacked transistors in the output stage of the buffer (recall that the transistors in the output stage of the buffer are the dominant component of the total buffer area). Thus, the area overhead is reduced from 26 minimum width transistors to ~6.5 minimum width transistors. However, since only the leakage of the output stage is reduced, the IG-TSB has inferior tri-state mode leakage power compared to the SG-TSB (we observed an 53% reduction in leakage power compared to a conventional buffer). The proposed PG-TSB offers further reductions in area overhead compared to the IG-TSB, as the area overhead is reduced from ~6.5 minimum width transistors to 3 minimum width transistors. The proposed buffer topology also shows decreased tri-state mode leakage power compared to the IG-TSB. This is because the proposed topology is able to employ narrower transistors in the input stage and allows for simpler gating circuitry, both of which lead to reduced leakage current.

However, the proposed PG-TSB buffer is still unable to match the low leakage power of the conventional SG-TSB. On the other hand, the proposed LOPG-TSB offers substantial leakage reduction – in excess of 90% – offering improved leakage reduction in comparison to the SG-TSB, although this comes at an area cost of an additional ~6.5 minimum-width transistors compared to the PG-TSB topology, which was incurred because of the PMOS transistor stacks employed in the input stage of the buffer. It
is worth noting that whereas the active mode leakage of the SG-TSB, IG-TSB, and PG-TSB circuits is approximately equal to a conventional buffer’s leakage, the LOPG-TSB’s active leakage is \( \sim 74\% \) lower than a conventional buffer; the leakage optimizations applied in this topology benefit both the active and tri-state leakage of the buffer. One final consideration is the fact that the dynamic power dissipated in the LOPG-TSB topology will be marginally greater than the PG-TSB topology. This is because of the increased gate and junction capacitances of transistors \( M_3 \) and \( M_5 \) as well as the increased voltage swing of the internal nodes of the buffer (the drains of \( M_3 \) and \( M_5 \)). However, given that the load capacitance at the output of the routing buffer dwarfs the capacitances of the buffer’s internal nodes, the increase in dynamic power due to the aforementioned reasons is relatively small – our simulations indicate a \( \sim 1.65\% \) dynamic power increase for the LOPG-TSB compared to the PG-TSB. We include this penalty in all dynamic power results presented in this work pertaining to the LOPG-TSB.

Thus, the proposed PG-TSB and LOPG-TSB circuits are two attractive low cost/low power options allowing for the implementation of tri-state routing buffers in FPGAs. However, our proposed power reduction reduction technique does not rely on any specific topology; indeed a conventional tri-state buffer topology may be employed, albeit at a much larger area penalty.
4.5 Interconnect Modelling

In this work, we assume that the layout of the routing conductors forming the interconnect network matches that of the standard VPR-style staggered interconnect model. Fig. 4.12 depicts the assumed organization of the routing conductors comprising a routing channel in this work.

![Routing Channel Diagram](image)

**Figure 4.12:** Assumed layout of routing conductors within a channel.

Fig. 4.12 depicts the assumed staggered organization of the routing conductors comprising a routing channel in this work. In this organization, adjacent conductors do not necessarily have complete overlap with one another. Rather, because the start points of interconnect are staggered, some adjacent conductors will only partially overlap with one another. Fig. 4.13 shows a detailed view of the coupling capacitances between adjacent conductors. Note that because of the staggered nature of the routing layout considered in this work, all wires (except for those at the boundaries of a channel) will have exactly three neighbours. In the figure, wire \((i, j)\) (where \(i\) is the track number in the channel and \(j\) is the segment number within track \(i\)) is coupled with routing wires \((i - 1, j)\), \((i - 1, j + 1)\), and \((i + 1, j)\). Thus, routing wire \((i, j)\) has three neighbours which for this work we number as follows: wire \((i - 1, j)\) is neighbour 1, wire \((i - 1, j + 1)\) is neighbour 2, and wire \((i + 1, j)\) is neighbour 3. We note that wires \((i, j)\) and \((i + 1, j)\) completely overlap with one another, and model the coupling between these two tracks with a capacitor with capacitance \(C_{C}\). As previously mentioned, some wires do not overlap completely with adjacent tracks, and this is the case with wires \((i - 1, j)\) and \((i - 1, j + 1)\), which have a 75% and 25% overlap with wire \((i, j)\), respectively. As such, we model the coupling between wires \((i - 1, j)\) and \((i, j)\) and between wires \((i - 1, j + 1)\) and \((i, j)\) with capacitances of \(0.75 \cdot C_{C}\) and \(0.25 \cdot C_{C}\), respectively. Note that this example showed the various overlaps with neighbouring wires when \(i\) is an odd number. When \(i\) is an even number, wire \((i, j)\) will have complete overlap with wire \((i - 1, j)\), 25% overlap with wire \((i + 1, j - 1)\) and 75% overlap with wire \((i + 1, j)\). Future work may consider alternate segmentation patterns, including those which employ wire swizzling to minimize crosstalk-related delay uncertainty [37].

4.6 Tool Support

In order to ensure that specific routing conductors remain unused to optimize power and timing we have made modifications to a conventional FPGA router. The router is modified to encourage power and timing critical signals to be routed with unused adjacent tracks, while the leaky pins of any used multiplexer are connected to unused routing conductors. The router modifications are described in the
The modifications to the router in this work bears some similarity to a previously proposed crosstalk-aware router for FPGAs \cite{101}, in that signals are routed such that they have unused adjacent tracks (if they have high activity or are timing critical). However, the difference between this work and prior work is that here we proposed to leave adjacent conductors unused to reduce the effective capacitance of used routing conductors, thereby reducing power and improving speed. This is in contrast to previous work, where adjacent conductors were left unused (and were grounded, not tri-stated as is the case in our work) to reduce coupling between active routing conductors, thereby mitigating the delay penalty due to crosstalk.

### 4.6.1 Power Optimization Overview

The VPR router incorporates the PathFinder algorithm \cite{77}. PathFinder routes individual driver/load connections one at a time. A cost function is used to find a low-cost path through the routing fabric from a driver to a load. The cost function incorporates metrics of timing performance and routability (i.e. the ability to legally route all of the connections in the circuit) in order to yield a routed circuit with a favourable quality of results. The cost function implemented in VPR is:

\[
\text{Cost}_n = (1 - C_{\text{Crit}}) \cdot \text{cong.cost}_n + C_{\text{Crit}} \cdot \text{delay.cost}_n
\]  

(4.5)

where \(n\) is the routing resource (e.g. wire segment) being considered for addition to a partially-completed route, \(i\) is the driver/load connection being routed, \(C_{\text{Crit}}\) is the timing criticality of the connection (equal to 1 for connections on the critical path, and decreasing to 0 as the slack of the connection increases), \(\text{cong.cost}_n\) is the congestion cost of routing resource \(n\), which gives an indication of the demand for the routing resource among nets, while \(\text{delay.cost}_n\) is the delay of routing resource \(n\).
The rationale for cost function (4.5) is that when connection $i$ is being routed to a routing resource $n$, if the connection has high criticality (i.e. $\text{Crit}_i$ is close to 1), then the cost of using the routing resource is principally given by the delay of that resource (equal to $\text{delay}_n$). For such timing-critical connections, the principal concern should be to route the connections using the fastest routing resources, with less focus on the demand for such resources by other nets. On the other hand, if the criticality of the connection being routed is low (i.e. $\text{Crit}_i$ is close to 0), then the cong$_n$ is the dominant part of the cost of using a resource. As such, connections with sufficient timing slack are encouraged to use resources with less demand, potentially pursuing more circuitous routes to reduce demand on over-congested conductors (nodes).

To optimize the interconnect power of a design, we augmented the conventional cost function with terms to maximize the likelihood that dynamic and/or static power can be saved using the tri-state-based optimizations described previously. The augmented cost function is as follows:

\[
\text{Cost}_n = (1 - \text{Crit}_i) \cdot [\text{cong}_n + DP \cdot \text{dp}_n(i,n) + SP \cdot \text{sp}_m(m,n)] + \text{Crit}_i \cdot \text{delay}_n,
\]

(4.6)

where $m$ is the routing node immediately upstream from $n$ for the given connection being routed, $DP$ is a scalar dynamic power optimization tuning parameter, and $SP$ is a scalar static power optimization tuning parameter. The primary modification to the cost function is the addition of two terms $(1 - \text{Crit}_i) \cdot DP \cdot \text{dp}_n(i,n)$ and $(1 - \text{Crit}_i) \cdot SP \cdot \text{sp}_m(m,n)$, which attempt to optimize dynamic and static power, respectively, if net $i$ has sufficient slack. Critical nets ignore these optimizations to maximize circuit performance. Also, observe that while the term $\text{Crit}_i \cdot \text{delay}_n$ is the same as in 4.5 we modified $\text{delay}_n$ to account for the delay impact of having used or unused adjacent routing conductors (which can be tri-stated to allow for capacitance reduction which improves delay). The details on the terms $\text{dp}_n(i,n)$ and $\text{sp}_m(m,n)$ are described in the following sections.

### 4.6.2 Dynamic Power Optimization

The term $\text{dp}_n(i,n)$ is used to optimize dynamic power by steering signals to use routing conductors with unoccupied adjacent routing conductors, while also steering signals away from unused conductors if they happen to be adjacent to used routing conductors which carry power-critical nets. $\text{dp}_n(i,n)$ is given by:

\[
\text{dp}_n(i,n) = \alpha_i \cdot \text{cap}_n(n) + \text{dp}_n\text{infringe}_n(n)
\]

(4.7)

where $\alpha_i$ is the activity factor for connection $i$ being routed (normalized to a [0:1] range). The term $\text{cap}_n(n)$ determines the capacitance of node $n$ given the occupancies of the first and second level neighbours of node $n$, and is equal to:

\[
\text{cap}_n(n) = \sum_{j \in \text{nbrs}(n)} C^2_{\text{eff}}(n,j)
\]

(4.8)

where $\text{nbrs}(n)$ gives the set of routing conductors which are first level neighbours of $n$. $C^2_{\text{eff}}(n,j)$ is
given by:

$$C_{\text{eff}}^2(n,j) = \begin{cases} 
C_C(n,j) & \text{if } j \text{ used} \\
\frac{C_C(n,j) + \sum_{k \in \text{nbrs}(j), k \neq n} C_{\text{eff}}(j,k)}{C_C(n,j) + \sum_{k \in \text{nbrs}(j), k \neq n} C_{\text{eff}}(j,k)} & \text{otherwise}
\end{cases} \quad (4.9)$$

Where $C_P(j)$ is the plate capacitance of conductor $j$. Similarly, $C_{\text{eff}}^1(j,k)$ is given by:

$$C_{\text{eff}}^1(j,k) = \begin{cases} 
C_C(j,k) & \text{if } k \text{ used} \\
\frac{C_C(j,k) - (C_{\text{tot}}(k) - C_C(j,k))}{C_{\text{tot}}(k)} & \text{otherwise}
\end{cases} \quad (4.10)$$

In the above equations, $C_C(i,j)$ gives the coupling capacitance between conductors $i$ and $j$ and $C_{\text{tot}}(i)$ gives the total plate and coupling capacitance of routing conductor $i$. The above equations indicate that the values of $C_{\text{eff}}^1(i,j)$ and $C_{\text{eff}}^2(i,j)$ are dependent on the state of the neighbouring conductors – i.e. which of the first and second-level neighbour conductors are occupied. The term $C_{\text{eff}}^1(i,j)$ is solely a function of the occupancy of node $j$, while the term $C_{\text{eff}}^2(i,j)$ is a function of the occupancies of the neighbours of $j$. Since for typical track layouts, the number of neighbours any given track will have is unlikely to exceed four [43], the total number of possible values for $C_{\text{eff}}^2(i,j)$ is relatively small.

Moreover, the number of possible values for $C_{\text{eff}}^1(i,j)$ is exactly two. All possible values of these two terms for each conductor may therefore be pre-computed and looked-up during routing. This allows the router to have access to an accurate estimate of the capacitance of each routing conductor while avoiding the need to perform many additional computations within its inner-loop.

The number of possible states to pre-compute $C_{\text{eff}}^1(i,j)$ increases exponentially with $n$, since the number of possible values for this term is $O(2^m)$, where $n$ is the number of levels of neighbours, and $m$ is the average number of distinct neighbours each routing conductor has for particular track layout. Therefore, it is evident that any effort to pre-compute and cache the values of $C_{\text{eff}}^1(i,j)$ for large values of $n$ is prohibitive, while attempting to compute this quantity within the inner loop of the router is simply intractable due to the number of computations required to compute a single capacitance value. As described in Section 4.3.1, the difference between $C_{\text{eff}}^2(i,j)$ and $C_{\text{eff}}^0(i,j)$ is within $\sim 5\%$ for $\rho_C$ values between 1 and 3, as such we believe that for this work, considering two-levels of neighbours provides a suitable balance between accuracy and tool runtime/memory requirements.

Referring again to Equation 4.7, the term $dp_{\text{infringe\_cost}}(n)$ is a term used to guide nets away from unused conductors which are adjacent to used conductors carrying power critical nets. This term is given by:

$$dp_{\text{infringe\_cost}}(n) = \sum_{j \in \text{nbrs}(n)} \Delta_{DP}(j,n) \quad (4.11)$$

Where $\Delta_{DP}(j,n)$ is the increase in power for the net with the highest activity factor which uses conductor $j$, if $n$ becomes a used conductor. If $j$ is unused, then $\Delta_{DP}(j,n)$ considers the worst case power impact on the neighbours of $j$ apart from $n$, if $n$ is used (i.e. considers the impact to power on

\footnote{It can be shown that the number of possible values for $C_{\text{eff}}^0(i,j)$ is equal to $2^{m-1} + 1$, where $m$ is the number of first-level neighbours for conductor $j$.}
second-level neighbours of $n$). Thus, $\Delta DP(j, n)$ is given by:

$$\Delta DP(j, n) = \begin{cases} \frac{\alpha_{\text{max}}(j) \cdot C^2_{\text{eff}}(j, n)}{\sum_{i \in \text{nbrs}(n)} C^2_{\text{eff}}(i, j) \cdot C_{\text{eff}}(n, i)} & \text{if } j \text{ used} \\
\sum_{k \in \text{nbrs}(j)} \frac{\alpha_{\text{max}}(k) \cdot C^2_{\text{eff}}(k, j) \cdot \Delta C_{\text{eff}}(j, n)}{C^2_{\text{eff}}(j, n, k)} \cdot C_{\text{eff}}(n, m) & \text{otherwise} \end{cases} \quad (4.12)$$

where $C^k_{\text{eff}}(j, n, k) = C_p(j) + C_C(k, j) + C_C(j, n) + \sum_{l \in \text{nbrs}(j), l \neq k, l \neq n} C^1_{\text{eff}}(j, l)$ is the first order capacitance at $j$ when we assume that $n$ and $k$ are used, $\Delta C^1_{\text{eff}}(j, n) = C^1_{\text{eff}}(j, n, m)$ represents the change in the first order capacitance seen by $j$ when looking towards $n$ if $n$ becomes a used conductor, and $\alpha_{\text{max}}(j)$ represents the worst case activity factor amongst all nets currently vying for conductor $j$. As such, the term $dp\_infringe\_cost(n)$ effectively determines the worst case impact to routing dynamic power arising from a net attempting to use routing conductor $n$ given the occupancies of first and second degree neighbours of $n$.

### 4.6.3 Static Power Optimization

In a manner similar to how we optimize dynamic power, we optimize static power by steering signals to use routing multiplexers which are expected to have maximum leakage reduction, and avoiding situations where we increase the static power of a routing multiplexer.

A high-level overview of how we accomplish these goals is as follows: while routing net $i$, when expanding from node $m$ to node $n$, if $n$ is a routing conductor then $m$ connects to $n$ through a routing multiplexer $s$. Given knowledge of the connection pattern of the multiplexer $s$ (i.e. the mapping between input pins of $s$ and routing tracks which fan-in to $s$), we can determine the set of leaky pins of $s$ when connecting $m$ to $n$, which we denote $\text{leaky}(m, s)$. Clearly, if the occupancies of the tracks connected to the pins $p \in \text{leaky}(m, s)$ is low, we have a high probability of operating this routing multiplexer in a low-leakage state (by setting the unused tracks to operate in tri-state mode), and so we should favour expanding the route from $m$ to $n$. Conversely, if a large number of tracks connected to the pins $p \in \text{leaky}(m, s)$ are used, the leakage of routing multiplexer $s$ will be higher, and therefore, if $i$ has sufficient timing slack (low criticality), we should try to avoid $s$. Furthermore, when expanding from $m$ to $n$, routing conductor $n$ may connect to a large number of routing multiplexers, and there may be cases where using track $n$ increases the static power of routing multiplexers in the fanout of $n$ but not used by net $i$, if track $n$ happens to connect to a “leaky pin” of a routing multiplexer used by another net.

The expression for $sp\_cost(m, n)$ is therefore as follows:

$$sp\_cost(m, n) = \text{leakage}(m, n) + \text{leakage\_infringe}(n) \quad (4.13)$$

where $\text{leakage}(m, n)$ effectively counts the number of pins $p \in \text{leaky}(m, s)$ which connect to used routing conductors. The term $\text{leakage\_infringe}(n)$ is given by:

$$\text{leakage\_infringe}(n) = \sum_{\substack{\text{max } s \in \text{fanout}(n) \\text{off path leakage}(n, s)}}$$

Where $\text{fanout}(n)$ gives the routing multiplexers in the fanout of $n$. The term $\text{off path leakage}(n, s)$ estimates the resulting increase in the leakage of routing multiplexer $s$ if $n$ is used, by determining whether or not $n$ connects to a leaky pin of $s$. Recall that the leaky pins of $s$ are determined by the particular
Chapter 4. Leveraging Unused Resources for Energy Optimization of FPGA Interconnect

pin which multiplexer \( s \) selects. However, during routing, there may be many nets vying for the output node of a multiplexer, as such, given contention for a particular routing node during routing, it may be difficult to identify whether or not track \( n \) connects to a leaky pin of \( s \) in a manner which is not too pessimistic. To handle this, in this work we keep track of all signals vying for every routing conductor, and sort them in order of criticality. Intuitively in a negotiated congestion router, the nets with the least criticality are the first to try to find other routing options when they face competition for routing nodes along a current routing solution. As such, it is the most critical nets that are most likely to “win” as different nets compete for the same routing resources. By keeping track of the nets vying for each routing conductor and then sorting them by criticality, we are able to estimate which net is most likely to prevail for each routing conductor, and use this to estimate which set of pins will be leaky during routing.

4.7 Experimental Study

To assess the merits of the proposed energy reduction techniques, we used our modified version of VPR to place and route the set of benchmark circuits packaged with VTR 7.0 [74], as well as circuits from the MCNC benchmark suite. Resource statistics for the benchmarks considered in this work are shown in Table 4.2. We performed post-routing analysis to estimate the power reductions achievable using our proposed technique versus a baseline architecture. Our baseline architecture uses the Wilton switch block [100], has logic blocks with ten 6-LUTs/FFs per CLB, and contains unidirectional wire segments (direct-drive) which span 4 CLB tiles. For our experiments, we estimate that the total load capacitance of a length-4 wire in a commercial FPGA is near 200 \( fF \) as follows: Based on the tile area data in [103], we estimated a single tile length in each dimension is 170 \( \mu m \) for an FPGA in 65 \( nm \) CMOS which implies a wire capacitance of 170 \( fF \) for a length-4 wire, assuming a wire capacitance per length of 0.25 \( fF/\mu m \).

We anticipate that the additional parasitic capacitance of a length-4 wire (resulting from the diffusion capacitance of the many multiplexers connected to a length-4 wire) will push the total load capacitance close to 200 \( fF \) in 65 \( nm \) CMOS.

All benchmark circuits were initially routed on the baseline architecture to determine the minimum number of tracks per channel needed to route each circuit successfully (\( W_{min} \)). For each circuit, we then computed \( W_{med} = 1.3 \times W_{min} \) to reflect a medium-stress routing scenario for that particular circuit. We also found the largest \( W \) for all circuits considered in the benchmark set, multiplied this value by 1.1, and set \( W_{max} \) to this value. \( W_{max} \) in this case corresponds to the situation where the designs in the benchmark set are routed in the same routing architecture (same channel width). We believe this may be a more realistic representation for the results one would expect if a commercial FPGA used the techniques described in this work. Thus, we have performed three sets of experiments to reflect power reduction under a spectrum of different routing congestion scenarios; in the first set of experiments we set \( W = W_{min} \) which represents the maximum possible congestion; in the second we set \( W = W_{med} = 1.3 \times W_{min} \) to reflect a medium-stress scenario, and finally in our last set of experiments we assumed \( W = W_{max} \). We further divide our experiments to study the achievable power reductions for the two different tri-state buffer topologies considered in this work (PG-TSB and LOPG-TSB).

We use the ACE switching activity estimator tool [59] to compute switching activity for each signal.

---

\(^{3}\)It is important to note that since \( W_{min} \) represents the maximum possible congestion, it is highly likely a circuit will be unroutable if the router seeks to optimize additional objectives, as is the case in this work. As such for the set of experiments where \( W = W_{min} \) we increased the router iteration limit to a very large number such that these circuits would be routable in the presence of the additional power optimization objectives.
Chapter 4. Leveraging Unused Resources for Energy Optimization of FPGA Interconnect

Table 4.2: Resource statistics for VTR and MCNC circuits.

<table>
<thead>
<tr>
<th>Circuit</th>
<th># LUTs</th>
<th># BRAM</th>
<th># DSP</th>
<th>Circuit</th>
<th># LUTs</th>
<th># BRAM</th>
<th># DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>1522</td>
<td>0</td>
<td>0</td>
<td>bgm</td>
<td>32384</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>apex2</td>
<td>1878</td>
<td>0</td>
<td>0</td>
<td>blob merge</td>
<td>6019</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>apex4</td>
<td>1262</td>
<td>0</td>
<td>0</td>
<td>boundtop</td>
<td>3054</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>bigkey</td>
<td>1699</td>
<td>0</td>
<td>0</td>
<td>ch intrinsics</td>
<td>425</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>clma</td>
<td>8365</td>
<td>0</td>
<td>0</td>
<td>diffeq1</td>
<td>485</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>des</td>
<td>1591</td>
<td>0</td>
<td>0</td>
<td>diffeq2</td>
<td>322</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>diffeq</td>
<td>1494</td>
<td>0</td>
<td>0</td>
<td>LU32PEEng</td>
<td>76281</td>
<td>5251</td>
<td>32</td>
</tr>
<tr>
<td>dsp</td>
<td>1362</td>
<td>0</td>
<td>0</td>
<td>LU8PEEng</td>
<td>22675</td>
<td>1409</td>
<td>8</td>
</tr>
<tr>
<td>elliptic</td>
<td>3602</td>
<td>0</td>
<td>0</td>
<td>mcm</td>
<td>101751</td>
<td>348</td>
<td>30</td>
</tr>
<tr>
<td>ex1010</td>
<td>4598</td>
<td>0</td>
<td>0</td>
<td>mkDelayWorker32B</td>
<td>5595</td>
<td>1336</td>
<td>0</td>
</tr>
<tr>
<td>ex5p</td>
<td>1064</td>
<td>0</td>
<td>0</td>
<td>mkPktMerge</td>
<td>232</td>
<td>459</td>
<td>0</td>
</tr>
<tr>
<td>frisc</td>
<td>3539</td>
<td>0</td>
<td>0</td>
<td>mkSMAdapter4B</td>
<td>1977</td>
<td>153</td>
<td>0</td>
</tr>
<tr>
<td>misex3</td>
<td>1397</td>
<td>0</td>
<td>0</td>
<td>or1200</td>
<td>3054</td>
<td>64</td>
<td>1</td>
</tr>
<tr>
<td>pdc</td>
<td>4575</td>
<td>0</td>
<td>0</td>
<td>raygentop</td>
<td>2147</td>
<td>21</td>
<td>18</td>
</tr>
<tr>
<td>s298</td>
<td>1930</td>
<td>0</td>
<td>0</td>
<td>sha</td>
<td>2277</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s38417</td>
<td>6042</td>
<td>0</td>
<td>0</td>
<td>stereovision0</td>
<td>11460</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s38584.1</td>
<td>6177</td>
<td>0</td>
<td>0</td>
<td>stereovision1</td>
<td>10289</td>
<td>0</td>
<td>152</td>
</tr>
<tr>
<td>seq</td>
<td>1750</td>
<td>0</td>
<td>0</td>
<td>stereovision2</td>
<td>29942</td>
<td>0</td>
<td>564</td>
</tr>
<tr>
<td>spla</td>
<td>3690</td>
<td>0</td>
<td>0</td>
<td>stereovision3</td>
<td>180</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tseng</td>
<td>1046</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In each benchmark circuit, the tool was run with assumed static probabilities of 0.5 and switching probabilities of 0.2 for all primary inputs. Activity data is required by our modified CAD flow (see (6.2)) to optimize routing for power reduction, as well as for post-routing dynamic power estimation. Routing static power was estimated by first building look-up table models to relate the output voltage and input pin leakage of a single-stage routing multiplexer to the voltages of its input pins. This can then be used to estimate the total leakage of a two-stage routing multiplexer for any specific combination of input voltages across all of its input pins. We used SPICE simulations with a commercial 65nm process to determine the added delay of our proposed tri-stateable buffer over a conventional buffer, to assess the capacitance reductions achievable for the various values of \( \rho_C \) considered in this study, and to study leakage power under various modes of operation. All simulations assumed a \( V_{DD} \) of 1.0 V, and were run at a temperature of 85°C at the TT process corner. Finally, all experiments were run on the SciNet high performance computing system.

4.7.1 Router Parameter Optimization

The cost function in Equation 4.6 includes two scalar parameters \( DP \) and \( SP \) which must be appropriately selected for each circuit. Ideally, we would want a circuit’s routing energy to be minimized through the power optimizations proposed in this work with minimal impact to performance. We selected these parameters by doing the following: First, each circuit was routed using a range of different combinations of \( DP \) and \( SP \). For each such parameter combination, we were then able to obtain estimates for the reduction of interconnect dynamic power, reduction of interconnect static power, and degradation of

---

4 This approach is taken because the number of simulations required to characterize the leakage of a multiplexer grows exponentially with the number of input pins. As such, any attempt to fully characterize a two-stage multiplexer was deemed to be intractable.

5 Experiments were run on SciNet’s General Purpose Cluster (GPC) whose nodes contain an Intel Xeon E5540 processor (four 2.53 GHz CPUs) with 8GB of main memory.
Let $F_{\text{max}}$ (the maximum operating frequency of the circuit). From these results we built for each circuit two-dimensional piecewise-linear (PWL) models to relate the reduction of dynamic power, reduction of static power, and degradation of $F_{\text{max}}$ for each specific combination of $DP$ and $SP$. We denote these PWL models as $\Delta_D(DP,SP)$, $\Delta_S(DP,SP)$, and $\Delta_F(DP,SP)$, respectively. For any given circuit with baseline interconnect dynamic power $D_B$, baseline interconnect static power $S_B$, and baseline maximum operating frequency $F_{\text{max}}$, the energy dissipated in the interconnect, $E_{\text{int}}$, is given by:

$$E_{\text{int}} = \frac{D_B(1 - \Delta_D(DP,SP)) + S_B(1 - \Delta_S(DP,SP))}{F_{\text{max}}B(1 - \Delta_F(DP,SP))}$$

(4.15)

Therefore, for all subsequent experiments we used Equation (4.15) to find values of $DP$ and $SP$ which minimized energy while also ensuring that the impact to performance was minimal. In this work, we ensured that $\Delta_F(DP,SP) < 5\%$. Fig. 4.14 shows the variation in interconnect dynamic power reduction as $DP$ is varied between 0 and 10 assuming that $\rho_C = 2$; the family of curves shown in the figure reflect different values of $SP$, also ranging between 0 and 10. Note that the plots only show dynamic power reduction for an architecture employing the PG-TSB topology; the data corresponding to the LOPG-TSB topology would have identical trends as those depicted in this figure, with the exception that the dynamic power reductions for the LOPG-TSB topology are $\sim$2.2-2.3% lower because of the dynamic power overhead of this circuit as discussed in Section 4.4.1. Note that three sets of curves depicting the relationship between dynamic power reduction and the router’s parameters are shown in the figure for the three different channel widths assumed in this work.

The results show some interesting trends: First, as $DP$ is varied from 0 to 2, we see a steady increase in the achieved dynamic power reductions for both channel width assumptions, however as $DP$ is further increased, it becomes apparent that additional increases in dynamic power reduction are not possible. This is to be expected because when a net is being routed, the router has a limited scope when searching for tracks with unused adjacent routing conductors (which may be used to reduce capacitance), since any attempt to veer too far away from the natural route of that net (i.e. how the net would be routed in the absence of these power optimizations) would result in among other things, increased wirelength. The increase in wirelength results in increased total wire capacitance, defeating the purpose of searching for capacitance reduction opportunities in the first place. Also note the considerably greater dynamic power reductions possible when $W = W_{\text{max}}$; for most of the circuits in our benchmark set (with the exception of the largest circuits), $W_{\text{max}} > 1.3 \times W_{\text{min}}$, as such when $W = W_{\text{max}}$ the routing channels become less congested, giving the router greater opportunity to space routing conductors away from one another which reduces capacitance. At the other end of the spectrum, when $W = W_{\text{min}}$, the router finds very little opportunity to apply our proposed dynamic power reduction technique, regardless of the value of $DP$. Again, this is expected behaviour since a highly congested design offers few degrees of freedom for the router to explore alternative, more profitable routes.

The plot also shows competition between the part of the router cost function which seeks to minimize routing multiplexer leakage and the part of the cost function which seeks to minimize routing dynamic

---

6Observe that in a commercial CAD flow, such circuit-specific PWL models would not be available in general; rather, it is more likely that a commercial CAD flow would employ a single model which relates combinations of $DP$ and $SP$ to power reduction and performance degradation averaged over a set of benchmark circuits. Given that there will be some variation between an averaged model and a circuit-specific model (the averaged model will in general yield inferior quality-of-results), the results presented in this study represent an effective upper-bound on the achievable benefits and a lower-bound on associated costs of our proposed technique. However, we believe that for power-critical applications, a designer would be motivated to experiment with different $SP$ and $DP$ combinations, thereby searching for circuit-specific optimal parameters and thus approaching the results presented in this work.
Figure 4.14: Impact of DP and SP on interconnect dynamic power reduction and critical path.

power, as there is a degradation in the reduction of dynamic power as SP is increased (in fact there is a small increase in routing dynamic power when DP = 0 and SP ≥ 6). Recall from Section 5.4 to optimize dynamic power, one part of the router’s cost function attempts to make sure that the occupancies of the first and second-level neighbours of a used conductor are minimized. In contrast, to optimize static power, another part of the router’s cost function seeks to ensure the occupancies of routing conductors connected to the “leaky” pins of a used routing multiplexer is minimized. These different goals will complement one another if a large number (if not all) of the first and second-level neighbours for any routing conductor i are also connected to the set of pins p ∈ \(\bigcup_{s \in \text{fanout}(i)} \text{leaky}(i, s)\). In this case, both of these parts of the router’s cost function will seek to ensure the first and second level neighbours of i are unused. However, an analysis of the routing architecture (which includes switch connection pattern, track layout, and track-to-routing multiplexer connection pattern) assumed in this study reveals that only \(\sim 3\%\) of all pairs of routing conductors which are first or second-level neighbours of one another meet this condition, meaning that the optimization of static and dynamic power are evidently quite uncorrelated with one another. This would explain why increases in SP negatively impact routing dynamic power reduction, since the two objectives of dynamic and static power reduction are in competition with one another. This seems to indicate that a routing architecture which maximizes the number of conductors which meet the aforementioned condition would better serve the power reduction strategies discussed in this work. The search for such a routing architecture may be a potential avenue for future research.

Fig. 4.15 shows the variation in active routing static power reduction as SP is varied between 0 and 5; the family of curves shown in the figure reflects different values of DP which also ranges between 0 and 5. The plot contains three sets of curves which show the relationship between active static power reduction and the router’s parameters for the three different channel widths assumed in this work. Note that the power reductions shown in this figure correspond to an architecture which employs the PG-

---

7Active routing static power only accounts for the static power dissipated in used routing resources. Total routing static power, which is reported in detail in Table 4.3, includes static power dissipated in used and unused resources.
Figure 4.15: Impact of SP and DP on active routing leakage reduction.

TSB topology. As will be detailed in Table 4.3, the LOPG-TSB offers greater leakage reduction (as is expected), however the set of curves corresponding to this topology show similar overall trends as those depicted in this figure.

Fig. 4.15 reaffirms that the static and dynamic power reduction objectives of the router are in competition with one another, since increases in DP result in decreases in the amount of static power reduced. The figure also shows that the router is able to offer the greatest levels of power reduction when $W = W_{\text{max}}$, while power reduction is least when $W = W_{\text{min}}$. Again, this is because increased routing congestion naturally leads to reduced opportunities for the router to optimize for power, as such more relaxed channel widths will naturally lead to greater levels of power reduction. Finally, the figure shows that while the achievable static power reduction is an increasing function of SP (the number of opportunities to reduce routing multiplexer leakage increases by up to $\sim 12\%$ as SP approaches a value of 5), the gains in static power reduction diminish for larger values of SP. This is because again, power reduction opportunities must be weighed against increases in wirelength: observe that since the leakage of an unused routing switch is always less than that of an used routing switch (even one that is in a low leakage state), any effort to reduce static power will naturally try to minimize total wirelength. Consequently, the router must find opportunities to reduce routing multiplexer leakage while ensuring that wirelength is not negatively impacted. This places a constraint on the potential power reduction opportunities, which is what is observed in the plot. Because of the limitations in the leakage power model used by the router (Equations 4.13 and 4.14), large values of SP may result in a degraded reduction of routing leakage as the router trades-off wirelength with routing multiplexer leakage reduction opportunities in an unfavourable manner, which results in the non-monotonic behaviour of some of the curves shown in the figure.

4.7.2 Area Overhead

While the area overheads of the proposed circuits are relatively small, we wish to account for the power overheads resulting from increased tile area, since increased tile area will result in longer wires, and thus greater wire capacitance. The proposed PG-TSB topology uses three additional transistors: $M_3$, $M_4$, $M_5$. 

<table>
<thead>
<tr>
<th>SP</th>
<th>Active Routing Leakage Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>23%</td>
</tr>
<tr>
<td>1</td>
<td>28%</td>
</tr>
<tr>
<td>2</td>
<td>33%</td>
</tr>
<tr>
<td>3</td>
<td>38%</td>
</tr>
<tr>
<td>4</td>
<td>43%</td>
</tr>
<tr>
<td>5</td>
<td>43%</td>
</tr>
</tbody>
</table>
Table 4.3: Detailed power reductions assuming $\rho_C = 2$

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Blocks</th>
<th>PG-TSB</th>
<th>LOPG-TSB</th>
<th>PG-TSB</th>
<th>LOPG-TSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>128</td>
<td>5.4</td>
<td>327.1%</td>
<td>10.9</td>
<td>3.2</td>
</tr>
<tr>
<td>apex</td>
<td>170</td>
<td>3.4</td>
<td>390.8%</td>
<td>8.1</td>
<td>1.3</td>
</tr>
<tr>
<td>apex+</td>
<td>171</td>
<td>40.4</td>
<td>31.1%</td>
<td>8.8</td>
<td>1.8</td>
</tr>
<tr>
<td>bigsky</td>
<td>547</td>
<td>10.9</td>
<td>438.1%</td>
<td>14.8</td>
<td>8.7</td>
</tr>
<tr>
<td>clima</td>
<td>690</td>
<td>49.9</td>
<td>29.4%</td>
<td>13.7</td>
<td>2.6</td>
</tr>
<tr>
<td>dse</td>
<td>605</td>
<td>6.4</td>
<td>357.1%</td>
<td>12.6</td>
<td>4.2</td>
</tr>
<tr>
<td>disp</td>
<td>518</td>
<td>8.0</td>
<td>39.4%</td>
<td>11.0</td>
<td>5.8</td>
</tr>
<tr>
<td>eilpict</td>
<td>469</td>
<td>49.9</td>
<td>31.4%</td>
<td>13.5</td>
<td>2.7</td>
</tr>
<tr>
<td>ex010</td>
<td>316</td>
<td>5.1</td>
<td>321.1%</td>
<td>11.6</td>
<td>2.8</td>
</tr>
<tr>
<td>ex00p</td>
<td>146</td>
<td>5.0</td>
<td>32.1%</td>
<td>9.9</td>
<td>2.8</td>
</tr>
<tr>
<td>fpe</td>
<td>370</td>
<td>46.1</td>
<td>31.0%</td>
<td>14.1</td>
<td>2.3</td>
</tr>
<tr>
<td>neuro3</td>
<td>314</td>
<td>41.1</td>
<td>31.8%</td>
<td>10.7</td>
<td>1.8</td>
</tr>
<tr>
<td>pde</td>
<td>392</td>
<td>42.9</td>
<td>32.3%</td>
<td>11.7</td>
<td>1.9</td>
</tr>
<tr>
<td>d28h</td>
<td>131</td>
<td>5.2</td>
<td>32.5%</td>
<td>6.0</td>
<td>0.5</td>
</tr>
<tr>
<td>sl417</td>
<td>511</td>
<td>5.8</td>
<td>34.7%</td>
<td>14.4</td>
<td>3.6</td>
</tr>
<tr>
<td>sl4041</td>
<td>714</td>
<td>5.6</td>
<td>321.1%</td>
<td>12.7</td>
<td>3.4</td>
</tr>
<tr>
<td>sqx</td>
<td>397</td>
<td>3.7</td>
<td>30.4%</td>
<td>8.8</td>
<td>2.2</td>
</tr>
<tr>
<td>spla</td>
<td>328</td>
<td>3.7</td>
<td>30.2%</td>
<td>10.3</td>
<td>1.5</td>
</tr>
<tr>
<td>towng</td>
<td>237</td>
<td>6.9</td>
<td>356.1%</td>
<td>14.5</td>
<td>4.7</td>
</tr>
<tr>
<td>LUT2FEEng</td>
<td>7732</td>
<td>6.4</td>
<td>343.1%</td>
<td>18.1</td>
<td>4.1</td>
</tr>
<tr>
<td>LUT3FEEng</td>
<td>2435</td>
<td>46.1</td>
<td>31.0%</td>
<td>26.2</td>
<td>2.3</td>
</tr>
<tr>
<td>bgm</td>
<td>3417</td>
<td>5.6</td>
<td>331.1%</td>
<td>8.9</td>
<td>3.3</td>
</tr>
<tr>
<td>block_meta</td>
<td>677</td>
<td>36.2</td>
<td>31.5%</td>
<td>11.0</td>
<td>1.4</td>
</tr>
<tr>
<td>boundsp</td>
<td>792</td>
<td>6.0</td>
<td>32.6%</td>
<td>12.6</td>
<td>3.8</td>
</tr>
<tr>
<td>cl_kerns</td>
<td>367</td>
<td>6.1</td>
<td>470.1%</td>
<td>20.0</td>
<td>3.9</td>
</tr>
<tr>
<td>disp2cl</td>
<td>300</td>
<td>6.9</td>
<td>351.1%</td>
<td>15.7</td>
<td>4.6</td>
</tr>
<tr>
<td>disp2cl</td>
<td>294</td>
<td>9.3</td>
<td>43.3%</td>
<td>25.7</td>
<td>6.1</td>
</tr>
<tr>
<td>nand</td>
<td>2088</td>
<td>5.3</td>
<td>321.1%</td>
<td>26.0</td>
<td>3.1</td>
</tr>
<tr>
<td>mDiDapWorker32B</td>
<td>1554</td>
<td>6.9</td>
<td>407.1%</td>
<td>17.0</td>
<td>4.6</td>
</tr>
<tr>
<td>mDiPMerge</td>
<td>487</td>
<td>23.3</td>
<td>438.1%</td>
<td>21.8</td>
<td>11.1</td>
</tr>
<tr>
<td>mDiSMAdapter4B</td>
<td>570</td>
<td>6.0</td>
<td>351.1%</td>
<td>15.9</td>
<td>3.8</td>
</tr>
<tr>
<td>or1300</td>
<td>10.0</td>
<td>6.3</td>
<td>360.1%</td>
<td>17.4</td>
<td>4.1</td>
</tr>
<tr>
<td>avyangle</td>
<td>792</td>
<td>5.3</td>
<td>330.1%</td>
<td>11.3</td>
<td>3.1</td>
</tr>
<tr>
<td>ssa</td>
<td>282</td>
<td>6.5</td>
<td>366.1%</td>
<td>17.6</td>
<td>4.3</td>
</tr>
<tr>
<td>stereo2c</td>
<td>1267</td>
<td>31.6</td>
<td>350.1%</td>
<td>11.9</td>
<td>4.9</td>
</tr>
<tr>
<td>stervision3</td>
<td>1262</td>
<td>48.3</td>
<td>30.0%</td>
<td>12.7</td>
<td>2.5</td>
</tr>
<tr>
<td>stereo2v1</td>
<td>2892</td>
<td>65.3</td>
<td>369.1%</td>
<td>22.5</td>
<td>4.3</td>
</tr>
<tr>
<td>stereo2v2</td>
<td>96</td>
<td>9.5</td>
<td>409.1%</td>
<td>14.8</td>
<td>7.3</td>
</tr>
</tbody>
</table>

Mean: 9.0 34.8 14.9 3.8 69.2 25.2 12.0 41.0 22.4 9.8 75.2 33.6 17.0 49.1 30.2 14.7 81.0 42.7
and $M_5$. These transistors were sized such that the proposed buffer had a worst-case delay overhead of $\sim 10\%$. The resulting area overhead of the additional transistors is 3 minimum-width transistors ($M_3$, $M_4$, and $M_5$ were all minimum width) in addition to an SRAM cell needed for the mode configuration of each routing buffer. The LOPG-TSB on the other hand has a larger area overhead since it requires the use of PMOS stacks (which need to be adequately sized) to ensure low leakage operation, while not sacrificing performance relative to the PG-TSB. Consequently, this buffer has an area cost of approximately 15.5 transistors (9.5 transistors for the routing buffer, and an additional 6 for the required SRAM cell). The area of the baseline routing switch employed in this study, estimated through VPR’s standard area model is $\sim 98$ minimum width transistors. This results in a total area overhead of 2.6-2.9% for the PG-TSB and an area overhead of between 4.5-4.8% for the LOPG-TSB, depending on the assumed value of $W$. Assuming a square tile layout, we estimate a wirelength increase between $\sqrt{1.026} \approx 1.013x$ and $\sqrt{1.029} \approx 1.014x$ for the PG-TSB topologies and between $\sqrt{1.045} \approx 1.022x$ and $\sqrt{1.048} \approx 1.024x$ for the LOPG-TSB topologies. We include these power overheads in estimates of dynamic power reduction in this work.

**Dynamic Power Reduction**

In lieu of a specific value for $\rho_C$ which would vary with process node, and would depend on the specific layout of an FPGA tile, we assess the available power reductions as $\rho_C$ is varied from 1-3. In addition, we seek to determine the benefits of a second-order effective capacitance model (i.e. one which considers the occupancies of first and second-level neighbours) over a first-order model (one that only considers occupancies of first-level neighbours). Fig. 4.16 shows the variation in dynamic power reduction versus $\rho_C$ for three different scenarios. In the first scenario, we assume $W = W_{\text{max}}$, and the CAD flow has a second-order effective capacitance model. For the other two other scenarios, we assume $W = W_{\text{med}}$, and consider the effect of using a second-order model versus a first-order model. As shown in the figure, there is a clear increase in the achieved power reduction by employing a second-order model. As $\rho_C$ approaches a value of 3, the difference in the routing dynamic power reduction achieved between a CAD flow which uses a first-order model versus one which used a second-order model is $\sim 1.7\%$; relatively speaking, this represents an $\sim 11\%$ reduction in routing dynamic power between the two CAD flows.

Turning now to the effect of $\rho_C$ on interconnect dynamic power reduction, we observe that the power reductions afforded by the proposed technique increase as $\rho_C$ increases. Given that ITRS data shows the ratio $\rho_C$ increases with every passing technology node, it follows that the proposed technique offers excellent scalability – it is expected to allow for improved quality of results with each passing technology node. The interconnect power reductions exceed 17% as $\rho_C$ approaches a value of 3 for the case where $W = W_{\text{med}}$ (and a second order model is employed), while the power reduction exceeds 25% if $W = W_{\text{max}}$. Table 4.3 shows a circuit-by-circuit breakdown of the dynamic power reductions for the circuits considered in this work for the three different assumed values of $W$, given $\rho_C = 2$, which we believe is a reasonable estimate for the value of this parameter in modern processes. For this value of $\rho_C$, the results indicate that over the benchmark set, routing dynamic power reduction ranges between 3.8-17%, depending on the assumed value of $W$. Note that the results indicate that the dynamic power reductions achieved by employing LOPG-TSB topology are on average $\sim 2.3\%$ lower that the reductions achieved by employing the PG-TSB topology. This is because of a $\sim 1.6\%$ overhead arising from an increased voltage swing and increased parasitic capacitance of LOPG-TSB’s internal nodes, as well as increased interconnect capacitance due to a larger tile area (as discussed in Section 4.7.2). As a consequence of this penalty, very little dynamic power is saved using LOPG-TSBs when
W = W_{min}.

As was discussed in Section 4.2.1, the dynamic power dissipated in the routing network represents a significant portion of the total dynamic power dissipated in an FPGA. Consequently, we believe the proposed dynamic power reduction technique will yield a significant impact on the total dynamic power dissipated in an FPGA.

**Leakage Reduction**

Table 4.3 also indicates the routing static power reductions for each benchmark for the case when \( W = W_{min}, \ W = W_{med}, \) and \( W = W_{max}. \) For our baseline static power measurements, we assume that all unused routing conductors will have all configuration bits set to "0", and the routing multiplexer associated with unused routing conductors will thus operate in a low-leakage state. We assume all unused routing conductors will be driven to logic-"1", which, for the transistor models we have used in this study, leads to the lowest leakage state over all input vectors of used input pins of the routing multiplexer.

As mentioned previously, we also study the effects of the two different buffer topologies considered in this work, PG-TSB and LOPG-TSB, on leakage reduction. For the PG-TSB topology, the average routing static power reduction ranges between 34.8-49.1%, depending on the assumed value of \( W. \) Given the LOPG-TSB offers reduced routing buffer leakage in both tri-state and active modes, naturally we expect even greater static power reduction relative to the baseline. Our experiments indicate that depending on the value of \( W, \) the average routing static power reduction ranges between 69.2-81%. Comparing the two sets of numbers, it would appear that the LOPG-TSB topology allows an additional \( \sim 32-34\% \) reduction of total routing static power. Given that routing leakage comprises a significant portion of total FPGA leakage, we believe this technique will allow for an appreciable reduction in total FPGA leakage.
Overall Energy Reduction

The combination of reduced dynamic and static power with our proposed tri-state buffer topologies and CAD flow yield significant reductions in the total energy dissipated in the routing network. Assuming that each circuit operates at its maximum operating frequency $F_{\text{max}}$, our experiments indicate that routing energy may be reduced by 14.9-30.2% as $W$ ranges from $W_{\text{min}}$ to $W_{\text{max}}$ if we employ the PG-TSB topology. The significant leakage reduction offered by the LOPG-TSB topology allows for even greater energy reductions (despite lower reductions of dynamic power), ranging between 25.2-42.7% for the different assumed values of $W$. These results represent a significant improvement to the overall energy efficiency of FPGAs since the energy dissipated in the routing network is known to be a dominant component of total energy consumption [74].

From the point of view of energy reduction, the LOPG-TSB is the favourable topology given that it consistently offers $\sim$11-12% greater energy reduction across all channel width assumptions. Moreover, the benefit to energy consumption of the LOPG-TSB may become more pronounced in other applications. For the experiments conducted in this work, dynamic power dominates over static power for the majority of the circuits in our benchmark set. However, observe that not all applications require operation at the maximum operating frequency. For example, Internet-of-Things (IoT) applications typically require clock frequencies in the 100KHz-10MHz range and have extended periods in which the circuit is completely idle [89]. For such applications as these, minimization of static power will yield considerable improvement to overall energy dissipation.

However, the LOPG is more costly from the point of view of area overhead, and may have other costs which we have not been studied in this work (specifically, the cost of using the boosted supply rail, $V_{\text{DDB}}$ to drive a dynamic load). Ultimately, the two buffer topologies therefore offer different trade-offs, the superior option will therefore depend on the target application. The PG-TSB offers a smaller area cost and superior dynamic power savings. This topology is therefore favoured if it is desired to reduce energy with a minimal impact to tile area. On the other hand, the LOPG-TSB offers significantly greater leakage reduction and, despite inferior dynamic power savings, our experiments indicated that a greater amount of total interconnect energy may be reduced with this topology. This topology is therefore attractive for the most power-critical applications.

4.7.3 Router Run-Time

The modifications to the router’s cost function which we introduced in this work will naturally lead to degraded router run-time since the router now has additional objectives to evaluate when routing connections. In addition to finding paths which are fast and have low congestion, the router will now seek low-power paths as well, which adds complexity to its search algorithm. Table 4.4 shows the degradation to router run-time which arises as a result of our power optimization technique for a subset of the circuits (of varying sizes) from our benchmark set. For the circuits shown, we see router run-time degradation ranging between 5-100X. While such degradation may not be tolerated in main-stream commercial CAD tools, observe that the proposed power reduction technique may be presented as an optional optimization, i.e. users which require maximum power efficiency would choose to use the option while others may opt to disable it to reduce run-time.

The primary objective of our work was to evaluate the effectiveness of our proposed power optimiza-
Table 4.4: Router run-times for baseline router and power-aware router.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Run-time (seconds)</th>
<th>Run-time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU8PEEng</td>
<td>1316.2</td>
<td>60414.6</td>
</tr>
<tr>
<td>blob_merge</td>
<td>53.1</td>
<td>3667.9</td>
</tr>
<tr>
<td>ch_intrinsics</td>
<td>1.2</td>
<td>5.9</td>
</tr>
<tr>
<td>mkSMAAdapter4B</td>
<td>19.3</td>
<td>277.2</td>
</tr>
<tr>
<td>stereovision0</td>
<td>36.7</td>
<td>4435.2</td>
</tr>
</tbody>
</table>

Note that while we modified the router’s path cost function to encourage it to seek low-power paths, the router’s expected cost function was not modified. This approach therefore increased the extent by which the expected cost underestimates total path cost (since it does not model a large component of the path cost), which makes the router’s search largely breadth-first oriented, which degraded run-time. As such, an enhanced expected cost function which is able to predict the power cost to a reasonable degree of accuracy is likely to improve run-time. Recall that our power reduction techniques depend on the ability to ensure that certain tracks are left unused. Consequently, a net which is routed in a region with high congestion is likely to benefit less from our power optimization than a net which is routed in a region with very sparse track usage. Therefore, one possible approach which could be experimented with to improve the expected cost would be for the router to store the X and Y track utilizations at each grid location of the FPGA, which would be updated whenever a net is routed (or re-routed). The expected cost function could then estimate, for a given node-to-sink connection whether or not it would pass through regions of high congestion, and then form an estimate of power savings accordingly. Again, we leave enhancements to the router which improve run time, including an update to the expected cost function, for future work.

4.8 Summary

In this chapter, we presented a novel technique to reduce the capacitance of routing conductors in FPGAs, as well as reduce leakage power in the interconnect. We have shown that keeping unused routing conductors tied to a rail ($V_{DD}$ or $GND$) is suboptimal and imposes an unnecessary increased coupling capacitance to used routing conductors. The unused routing conductors may be exploited, since when they are left floating, they result in decreased capacitance to adjacent routing conductors. We also showed how unused routing conductors can also be used to minimize the leakage of routing multiplexers. We then presented two low-cost tri-stateable buffer topologies (with different area and power trade-offs) to allow for our proposed capacitance/power reduction technique with minimal area overhead. We further present CAD techniques to optimize the routing of a design to maximize capacitance and...
leakage reduction opportunities. Results have shown interconnect dynamic power reductions of up to
\(\sim 25\%\) when the ratio between coupling capacitance and plate capacitance of a wire, \(C_C/C_P\), reaches
a value of 3, while routing leakage reductions of up to \(\sim 81\%\) are possible, resulting in an reduction of
routing energy ranging from \(14.9-42.7\%\) at an average performance penalty of \(\sim 1.2-1.8\%\) and area cost
of \(\sim 2.6-4.8\%\) depending on the choice of tri-state buffer topology and channel width assumptions.
Chapter 5

Charge Recycling for Power Reduction in FPGA Interconnect

5.1 Introduction

As detailed in Chapter 2, the majority of prior research dedicated to dynamic power in FPGAs has attempted various multi-$V_{DD}$ approaches. There are two main drawbacks to a multi-$V_{DD}$ FPGA:

- The inherent area and timing penalties paid in a multi-$V_{DD}$ architecture, including the costs associated with on-chip power distribution.
- The off-chip costs associated with supplying multiple $V_{DDS}$.

The inherent area and timing trade-offs associated with a multi-$V_{DD}$ architecture are evident in Fig. 2.17. Note that in this figure, the power supply is not directly connected to the circuits; instead, supply transistors are used to select between different $V_{DD}$ rails. The inclusion of these supply transistors, which act in series with the pull-up-networks of the circuits being supplied, results in performance degradation. To counter the degradation in speed, the supply transistors and the transistors in the pull-up networks of these circuits need to increase in size. While some of the delay penalty introduced by the supply transistors can be recouped, in reality due to the increased loading from a larger pull-up-network on upstream circuits, there will always be some residual performance penalty. As such, from an area and delay point of view, there is an unavoidable penalty in a configurable multi-$V_{DD}$ architecture.

Other architectures with fixed supply blocks have been attempted [69][70], where intelligent placement and routing were proposed to “select” a circuit’s $V_{DD}$, and the area/delay penalty resulting from supply transistors is avoided. However, results from the authors’ work showed that this approach also had inherent area (and performance) penalties. Moreover, the cost of level-converters to interface between different voltage domains also imposes a non-negligible area cost [34]. The off-chip costs of a multi-$V_{DD}$ architecture cannot be ignored as well, since multiple supply voltages require the use of multiple off-chip voltage regulators, which in turn require additional expensive passive components, increasing cost and board area.

In this chapter we propose charge recycling (CR) as a dynamic power reduction technique for FPGAs. While this technique may be used in conjunction with other techniques – such as a multi-$V_{DD}$ architecture – it also provides an alternative path to reducing dynamic power while avoiding some of
the overheads of previous approaches, which have been reportedly quite large \cite{34, 71}. We specifically target interconnect dynamic power since, as was highlighted in Section 2.3.1, the dynamic power dissipated in the interconnect network of FPGAs accounts for a significant portion of an FPGA’s total power consumption. As with the set of power reduction techniques presented in Chapter 4, our approach leverages the prevalent unused routing conductors present in FPGAs. In modern commercial FPGAs, the unused routing conductors are wasted and pulled to a constant voltage. We propose instead that (some) conductors be programmably available as charge reservoirs (when they are unused), wherein they are each paired with a used conductor in close proximity. Given a used conductor with an available reservoir, charge from the used conductor is transferred to the reservoir on a falling transition (instead of dumping all charge to the ground rail). Conversely, charge from the reservoir is transferred to the used conductor on a rising transition, thereby reducing the amount of charge that needs to be drawn from the supply rail to bring the used conductor to rail $V_{DD}$. We propose a dual-mode interconnect switch design capable of CR. In the first (high-speed) mode, the switch functions as a traditional switch; in the second (low-power) CR mode, special circuitry within the switch realizes the charge recycling behavior as described. Note that this power reduction technique is complimentary to other dynamic power reduction techniques, such as operating with multiple power supplies, or $V_{DD}$ scaling. As such, combining charge recycling with other dynamic power reduction techniques will lead to further power reduction.

CAD tool support is required for CR to be applied successfully and we propose enhancements to a standard FPGA CAD flow to fully take advantage of the proposed power reduction technique. We altered the routing algorithm within the publicly-available VPR framework \cite{84} to encourage high-activity non-timing-critical signals to be routed using CR-capable switches that have available (free) reservoirs. We also implemented a post-routing mode selection pass that chooses the mode for CR-capable switches, subject to user-supplied speed performance constraints. Our mode selection is implemented as a mixed-integer linear program (MILP). The CAD tools are informed by models of the power, speed and area of CR-capable routing switches, derived from HSPICE simulation results for a 65nm commercial process. We studied the power reductions afforded by the proposed techniques, as well as impacts to speed and area. Results show that power in the FPGA interconnect can be reduced by up to 15-18.4%, depending on the architectural parameters and speed-performance constraints. While a number of circuit-level approaches to FPGA power reduction have been considered, including dual $V_{DD}$ \cite{69}, power-gating sleep mode \cite{95}, low-swing signaling \cite{14}, subthreshold techniques \cite{86}, and body-bias leakage control \cite{10}, to our knowledge, this is the first work to consider charge recycling.

5.2 Background

5.2.1 Charge Recycling

Charge recycling is a technique that has been explored in the ASIC domain, particularly in the application of on-chip interconnect \cite{91, 99}. The general idea is as follows: In conventional logic circuits, when a conductor makes a transition from logic “1” to “0”, the stored energy on the wire – equal to $CV_{DD}^2/2$ where $C$ is the wire capacitance – is completely dissipated. In contrast, charge recycling attempts to provide a mechanism with which the charge dissipated during a falling transition may be reused, thereby reducing the amount of charge which must be drawn from the power supply and thus reducing power consumption as well. One method to recycle charge would be to store some of the (normally) dissipated charge in secondary nodes (capacitors). This charge can then be delivered to a wire which is making a
transition from logic “0” to “1”, reusing some of the energy that would have been otherwise lost in a conventional circuit.

The charge recovery and recycling process is illustrated in Figs. 5.1(a) and 5.1(b) respectively. The figures show two capacitors: \( C_L \), the load capacitor, and \( C_R \), a reservoir capacitor used to store recovered charge. Switches connect \( C_L \) to either \( V_{DD} \) or \( GND \), and a third switch can be used to connect the two capacitors together. In Fig. 5.1(a), we see that \( C_L \) is initially connected to \( V_{DD} \), and there is no charge stored in \( C_R \). During a logic “1” to “0” transition at \( C_L \), initially the circuit undergoes a charge recovery phase, where \( C_R \) and \( C_L \) are connected (both \( V_{DD} \) and \( GND \) are disconnected from \( C_R \)). Given that the two capacitors have equal capacitance, through charge sharing, half of the charge initially stored in \( C_L \) will be transferred over to \( C_R \), and thus the voltages of these capacitors will settle to \( V_{DD}/2 \). After the charge recovery phase, the two capacitors are disconnected, and \( C_L \) is connected to \( GND \), and thus fully discharged.

In Fig. 5.1(b), \( C_L \) is initially connected to \( GND \), and the voltage at \( C_R \) is equal to \( V_{DD}/2 \) as a consequence of the charge recovery phase of a preceding logic “1” to “0” transition. In a “0” to “1” transition, the circuit initially undergoes a charge recycling phase where \( C_R \) and \( C_L \) are connected to one another while both the \( V_{DD} \) and \( GND \) rails are disconnected. Through charge sharing, the voltage at \( C_L \) will rise (while the voltage at \( C_R \) will fall) to \( V_{DD}/4 \). After the voltage at \( C_L \) has settled, \( C_L \) and \( C_R \) are disconnected from one another, and the \( C_L \) is connected to \( V_{DD} \) to complete the full transition to logic “1”. Note that in this example, the amount of charge actually drawn from the \( V_{DD} \) rail is equal to \( 0.75 \cdot C_L \cdot V_{DD} \) as opposed to \( C_L \cdot V_{DD} \). That is, there is an immediate 25% reduction in energy.

Figure 5.2 shows waveforms for the falling and rising transitions of a signal wire whose charge is being recovered and recycled. During a rising transition, we see two distinct phases: a first phase where the routing conductor is initially charged to an intermediate low voltage value \( V_{l_R} \) as charge is shared between the routing conductor and the charge reservoir, and a second phase where the routing conductor is fully charged to \( V_{DD} \). Similarly, during a falling transition, the output undergoes two distinct phases while discharging. In the first phase, the routing conductor is partially discharged to an intermediate voltage, \( V_{h_R} \), through charge sharing with a charge reservoir. In the second phase of discharging, the routing conductor is fully discharged to \( GND \).

In the steady state after several transitions have occurred, \( V_{l_R} \) and \( V_{h_R} \) will assume constant respective values. Assuming that the reservoir and load capacitors have equal capacitance, we note the following two relations that must hold:

\[
V_{h_R} = \frac{V_{DD} + V_{l_R}}{2} \quad (5.1)
\]

This condition states that during the recovery phase, the reservoir voltage after charge sharing is equal to the average of \( V_{DD} \) and the initial voltage of the reservoir capacitor prior to charge sharing, \( V_{l_R} \). The second relation is:

\[
V_{l_R} = \frac{V_{h_R}}{2} \quad (5.2)
\]

This second condition states that during the recycling phase, the reservoir voltage after charge recycling is simply half of the initial voltage of the reservoir capacitor prior to charge sharing, \( V_{h_R} \), since half of the charge on the reservoir capacitor would be transferred to the load capacitor (which initially is fully discharged). Substitution of (5.2) into (5.1) results in:

\[
V_{h_R} = \frac{2}{3} V_{DD} \quad (5.3)
\]
Figure 5.1: Charge recovery and recycling in a CMOS circuit with an auxiliary reservoir capacitor.

Figure 5.2: Charge recovery/recycling on transitions: output waveform during a rising transition (shown on left); output waveform during a falling transition (shown on right).
Substituting \((5.3)\) into \((5.2)\) leads to: \(V_R^l = \frac{1}{3}V_{DD}\). This implies that the theoretical maximum energy reduction from the proposed charge recycling interconnect is \(1/3\) (33%).

This analysis may be extended to consider the more general case where there may be more than one reservoir, and the reservoirs do not have the same capacitance as the load. Consider the case where \(n\) reservoirs are used to recycle charge in the manner described in this section; that is, denoting the \(i\)th reservoir \(C^i_R\), a rising transition at the output begins with \(C^1_R\) being temporarily connected to the output (allowing charge recovery from \(C^1_R\)), followed by \(C^2_R\), then \(C^3_R\), and so on until \(C^n_R\), after which the output is finally connected to \(V_{DD}\). Similarly, during a falling transition, the output is first temporarily connected to \(C^n_R\) (\(C^n_R\) recovers charge from the output), then \(C^{n-1}_R\), and so on until \(C^1_R\), after which the output is connected to \(GND\). To summarize, during a rising transition, reservoir \(C^i_R\) is temporarily connected to the output before \(C^{i+1}_R\), while during a fall transition, \(C^i_R\) temporarily connects to the output before \(C^{i-1}_R\).

Also assume that the load capacitance of the output node is \(C_L\), and that the capacitance of each reservoir \(C^i_R\) is \(\alpha_R \cdot C_L\) (for simplicity, we assume all the reservoirs have the same capacitance). Given the preceding discussion, we know that after every charge recovery event, the voltage of each reservoir will be at a maximum (since charge has been deposited into the reservoir); we denote the voltage of each reservoir \(i\) following a charge recovery event \(V^h_R^i\). Similarly, after each charge recycling event, the voltage of each reservoir will be at a minimum (charge has been drawn from the reservoir), and we denote the corresponding voltage \(V^l_R^i\). Thus, we may generalize Equations \((5.3)\) as follows:

\[
V^h_R^i = \frac{V^h_R^{(i+1)} + \alpha_R \cdot V^l_R^i}{1 + \alpha_R} \tag{5.4}
\]

This is a simple charge redistribution equation between the load and reservoir capacitors. Similarly, Equation \((5.2)\) may be generalized as follows:

\[
V^l_R^i = \frac{V^l_R^{(i-1)} + \alpha_R \cdot V^h_R^i}{1 + \alpha_R} \tag{5.5}
\]

From Equations \((5.4)\) and \((5.5)\), we may deduce the following:

\[
V^h_R^i - V^l_R^i = V^h_R^{(i+1)} - V^l_R^{(i+1)} = \frac{V^h_R^{(i+1)} - V^l_R^i}{\alpha_R + 1} \tag{5.6}
\]

\[
V^l_R^{(i+1)} - V^h_R^i = (\alpha_R - 1) \cdot \frac{V^h_R^{(i+1)} - V^l_R^i}{\alpha_R + 1} = (\alpha_R - 1) \cdot (V^h_R^i - V^l_R^i) \tag{5.7}
\]

Given that we have \(n\) reservoirs, we note that \(V^h_R^{(n+1)} = V_{DD}\) and \(V^l_R^0 = 0\). Consequently, the relation in Equation \((5.6)\) reduces to:

\[
V^h_R^i - V^l_R^i = \frac{V_{DD}}{\alpha_R(n + 1) + 1} \tag{5.8}
\]

This implies that \(V^h_R^n = (\alpha_R \cdot n \cdot V_{DD})/[(\alpha_R(n + 1) + 1] \), and represents the voltage of the output after charge has been recycled from the \(n\)th (and thus final) reservoir when the output undergoes a rising
transition. Observe the amount of charge which has been recycled is $C_L \cdot V_{Rn}$, thus $C_L \cdot (V_{DD} - V_{Rn})$ coulombs of charge need to be drawn from the supply to complete the rise transition. In contrast, $C_L \cdot V_{DD}$ coulombs of charge would need to be drawn from the supply during a rising transition for a conventional CMOS circuit. The amount of energy that has been recycled, and thus saved, is therefore:

$$E_R(\alpha_R, n) = \frac{\alpha_R \cdot n}{\alpha_R(n + 1) + 1}$$  \hspace{1cm} (5.9)

As previously alluded to, in this work we study the power savings that may be achieved if $\alpha_R = 1$ and $n = 1$; we previously calculated (and can now reconfirm with Equation (5.9)) that the theoretical maximum energy savings using this approach is 33%. However, we may use Equation (5.9) to investigate the maximum theoretical energy savings using other schemes where multiple reservoirs and/or reservoirs with larger capacitances are employed. We plot $E_R$ for these two different scenarios in Fig. 5.3. In Fig. 5.3(a), we study the theoretical maximum power savings for the case when $n = 1$ and $\alpha_R$ is varied – i.e. the impact of reservoir capacitance on power savings in a single reservoir charge recycling scheme. Fig. 5.3(b) shows the power savings for the case when $\alpha_R = 1$ and $n$ is varied, allowing us to study the impact of having a multitude of charge reservoirs (each having the same capacitance as the load).

The plots show that using a charge recycling scheme employing many reservoirs leads to greater power savings than one using a single large reservoir. Analytically, the maximum theoretical power reduction of either approach may be derived by considering the limits as either $n$ or $\alpha_R$ approach infinity:

$$\lim_{n \to \infty} E_R(\alpha_R, n) \bigg|_{\alpha_R = 1} = 1$$  \hspace{1cm} (5.10)

$$\lim_{\alpha_R \to \infty} E_R(\alpha_R, n) \bigg|_{n = 1} = \frac{1}{2}$$  \hspace{1cm} (5.11)

In other words, using a single reservoir allows for a maximum power savings of 50% if a sufficiently large reservoir is used. In contrast, using a large number of reservoirs will eventually allow all of the switching energy to be saved, which is effectively a form of adiabatic switching through a series of quasistatic charge transfers [110]. In fact, using just two reservoirs whose capacitances are the same as the load, we may theoretically predict a power savings of 50%, thus equalling the maximum theoretical power savings of a single reservoir charge-recycling approach (which uses an infinitely large reservoir capacitance). On the other hand, the downside to a multi-reservoir charge recycling approach is the increased complexity in the circuitry needed to stage the sequence of events required to recover and recycle charge from multiple reservoirs, in addition to increased switching delay. We revisit this discussion in Section 5.6.

5.3 CR-Capable FPGA Interconnect

To reduce power consumption in the routing network, we propose an FPGA architecture with a routing network containing charge reservoirs (i.e. spare capacitors) which can be used to store the charge from a signal wire which is being discharged, and then reuse the stored charge when the signal wire is later being charged to $V_{DD}$. In our case, the charge reservoirs are actually unused routing resources within the routing network. That is, we are not proposing to create “new” capacitors to be used as reservoirs; rather, we propose to use unused routing resources as reservoirs. In the next section, we describe changes to FPGA routing algorithms that steer towards routing solutions wherein high-activity signals are routed
Figure 5.3: Theoretical power savings for different charge recycling scenarios.
Chapter 5. Charge Recycling for Power Reduction in FPGA Interconnect

using resources that have available reservoirs.

Figure 5.4: CR buffer circuit.

The challenge in designing a charge-recycling FPGA interconnect switch is in realizing the dynamic behavior that must happen on rising and falling transitions. The switch must first disable the driving buffer, then it must allow charge sharing between the load and reservoir, and then, after charge sharing is complete, the switch must re-enable the driving buffer to complete the transition, bringing the load to one of the two rails. Note that the reason we must first disable the driving buffer is that otherwise, we may draw charge from the supply rail to charge both the load and the reservoir!

To implement the desired behavior, we designed the buffer circuit shown in Fig. 5.4 which shows a pair of routing buffers – labelled Routing Buffer A and Routing Buffer B – which consist of conventional buffers that have been augmented with additional circuit blocks labelled Gating Circuitry, Delay Line, and CR Circuit (the latter two blocks are shared by the two routing buffers); these blocks are needed to implement our proposed charge recycling technique. Together, the conventional buffer circuitry coupled with these additional circuit blocks comprise what we call a CR Buffer. The Gating Circuitry is used to disable the output stage so that charge may transfer from the output conductor to its paired reservoir, the CR Circuit is responsible for connecting the two paired routing conductors to one another during a charge recovery/recycling event, and the Delay Line is used to self-time the various phases of a CR-enabled rise or fall transition. As shown in Fig. 5.4 two additional SRAM configuration cells produce
signal $CR$ (charge recycling), which controls whether the two switches are operating in normal mode (high-speed) or CR mode, and signal $TS$ (tristate control) which determines which of the two switches is tristated (and thus which wire is acting as a reservoir) when operating in CR mode.

A high level description of the operation of the charge recycling buffer is as follows. Assuming that the switches shown in Fig. 5.4 are operating in CR mode, and that Routing Buffer A is active while Routing Buffer B is tri-stated, whenever there is a transition at $V_{INA}$, the calibrated delay of the Delay Line leads to a difference in signal levels between $V_{INA}$ and $DL_{OUT}$ (the output of the delay line). This difference results in the output stage of Routing Buffer A being momentarily tri-stated by the Gating Circuitry, and activates the CR Circuit which will connect the output load to the reservoir conductor – thus allowing charge to be shared between the two conductors. After the transition at $V_{INA}$ has propagated to the output of the delay line, and $V_{INA}$ and $DL_{OUT}$ are now equal, the CR Circuit will disconnect the output load from the reservoir, while the output stage will be re-enabled, allowing the output to completely rise (fall) to $V_{DD}$ ($GND$).

The Delay Line (Fig. 5.5) consists of two stages (as shown in the figure): the first stage is a current-starved inverter which serves as the main delay cell in the Delay Line, while the second stage (inverter) serves to simply buffer the output of the first stage. Also observe an input 2:1 multiplexer (formed by transistors $MD_7$-$MD_{10}$) is used to either select $V_{INA}$ or $V_{INB}$. Note that transistors $MD_1$ and $MD_4$ act as clamped current sources, as their gates are biased with voltages $V_{PB}$ and $V_{NB}$, respectively. The bias voltages are set to limit the amount of current that may flow through these transistors, intentionally slowing down the inverter. The delay of the delay line can be calibrated by adjusting the values of bias voltages on the clamped current sources and in fact, this delay must be calibrated to match the settling time of the signal and reservoir nodes during charge sharing for maximum power savings. This circuit also provides the signals $DL_{INT}$ (the gate of $MD_2$) and $\overline{DL_{INT}}$, which are needed by the CR Circuit as shown in Fig. 5.4. Finally, transistors $MD_{11}$-$MD_{13}$ are used to shut-off the delay line when not operating in CR mode.

![Figure 5.5: Circuit implementation of Delay Line in CR buffer.](image)

The description of the Gating Circuitry (Fig. 5.6), when operating in CR mode, is as follows: When
Figure 5.6: CR Buffer with Gating Circuitry exposed.

$V_{IN}$ makes a transition from logic “1” to “0”, $M1$ and $M4$ will both turn on immediately, raising the gate of $M13$ to $V_{DD}$ and thus turning it off. Due to the delay line, node $DL_{OUT}$ will initially continue to hold logic “1”. This combination of logic values results in $M1$ turning on, but $M7$ remaining turned off; this prevents the gate of $M14$ from being raised to $V_{DD}$ immediately following a “1” to “0” transition at the input. Thus, the output stage is momentarily tri-stated as both $M13$ and $M14$ are off. Note that the gate of $M14$ remains low during this period as it effectively retains the node voltage prior to the transition at $V_{IN}$. As the transition at $V_{IN}$ propagates through the delay line, $DL_{OUT}$ will make the transition from “1” to “0”, and this causes $M7$ to turn on, which results in $M14$ turning on, thus allowing the output to fall to logic “0” ($GND$ rail). A rising transition on $V_{IN}$ results in analogous behavior – temporary charge sharing with the reservoir before connecting the load to the $V_{DD}$ rail.

Regarding the mode selection, note that transistor $M9$ is controlled by $CR$, transistors $M10$ and $M12$ are controlled by $\overline{CR}$, while $M6$ and $M11$ are controlled by $TS$. When $CR$ is high and $TS$ is low $M9$, $M10$, and $M11$ are all off while $M6$ is on, resulting in the Gating Circuitry operating in the manner described above. When $CR$ is high and $TS$ is high, $M6$ will be off thus severing the connection between the drain of $M2$ and the gate of $M13$ while $M11$ will turn on thus driving the gate of $M13$ to $V_{DD}$. Assuming that an unused buffer will have its $V_{IN}$ held at $V_{DD}$ this ensures that $M14$'s gate will be driven to $GND$, thus placing the buffer in tri-state mode. Finally, when $CR$ is high, the buffer will operate in high-speed mode, since $M9$ and $M10$ (which are on when $CR$ is high) allow signal transitions to bypass transistors $M5$ and $M7$.

We now turn to the circuit that performs the charge sharing between the load and reservoir, the CR circuit. The details of the CR circuit are shown in Fig. 5.7. The CR circuit is implemented using pass transistor logic and drives the gates of the transistors connecting the switch output and charge
reservoir nodes. The description of the circuit is as follows: When $V_{IN}$ makes a transition from “0” to “1”, since $DL_{OUT}$ will remain at logical “0” for a brief period of time, $M_{17}$ will be turned on, resulting in transistor $M_{22}$ turning on, which allows charge to be recycled from the reservoir capacitor. After a brief period of time, $DL_{OUT}$ will make the transition to logical “1”, and this results in $M_{18}$ turning on and thus $M_{22}$ being shut off. This disconnects the reservoir node from the output. When $V_{IN}$ makes a transition from “1” to “0”, since again $DL_{OUT}$ will momentarily remain at logical “1”, the gate of $M_{21}$ will be pulled to GND through $M_{16}$, thus turning it on and allowing charge to be recovered from the load by the reservoir capacitor. Again, after the signal $V_{IN}$ has completely propagated through to $DL_{OUT}$, $M_{15}$ will turn on, $M_{16}$ will be shut off, which results in both $M_{21}$ and $M_{22}$ being shut off, again disconnecting the output from the reservoir capacitor.

![Charge recycling (CR) circuit.](image)

We simulated the proposed switch design using commercial 65nm STMicroelectronics models. Fig. 5.8 shows the simulated waveforms as the output node of a switch and a reservoir node when the input to the switch is driven by a signal with a high toggle rate. Note that, as expected, during the charge recovery/recycling phases, the output and reservoir node voltages converge to the same voltage – this is expected since during these phases the nodes are effectively short-circuited to one another.

As per the discussion in Section 4.7, we estimate a length-4 wire in a 65 nm process to have a capacitance of approximately 200 $fF$; for this load our simulations show a power savings of 27% through charge recycling. The 6% difference in simulated power savings and the theoretically-expected 33% reduction is because of the power consumption of the additional circuitry we introduced to facilitate charge recycling in addition to the increase in the worst case short circuit current of downstream routing buffers which arises because charge recycling and recovery leads to increased rise and fall transition times, respectively. This last point requires further discussion.

In general, a slow transition time at the input of a gate results in increased energy dissipated by the short-circuit current between the gate’s supply rails. This is because the gate will be in a state where both the pull-up and pull-down networks of the gate are simultaneously conducting for a longer period of time. The input voltage range over which both the pull-up and pull-down networks of an inverter are simultaneously conducting is between $V_{tn}$ (when the NMOS transistor begins to conduct) and $V_{DD} - |V_{tp}|$.
(when the PMOS transistor begins to conduct); we refer to this voltage range as an inverter’s power critical voltage region. Fig. 5.9 shows an abstraction of the output rise and fall transitions of a CR buffer. Observe that while the voltage transition during charge recycling or charge recovery is slower than that of a conventional CMOS buffer, the rate of the voltage transition after either charge has been recovered or recycled to/from the reservoir matches the rate of transition of a conventional CMOS buffer. Consequently, we will only see an increase in power consumption due to short circuit current if the phase during which charge is recycled or recovered by a CR buffer encroaches upon the power critical voltage regions of downstream buffers. Since the output voltage of a CR buffer ranges between 0 and $V_{DD}/3$ volts when charge is being recycled, and between $2V_{DD}/3$ and $V_{DD}$ when charge is being recovered, and given...
that the magnitude of the threshold voltages for the NMOS and PMOS transistors used in this work is near $V_{DD}/3$, the encroachment into power critical voltage region of a downstream buffer is minimal and does not result in significant increased power consumption. Moreover, routing buffers will be electrically disconnected, for all practical purposes, from the output of an upstream CR buffer when it recovers charge during a falling transition, since the CR buffer’s output in this phase transitions between $V_{DD}$ and $\frac{2}{3}V_{DD}$ which would lead the transistors in downstream routing multiplexers to effectively operating in cutoff during most of this time.

However, for other charge recycling approaches involving multiple reservoirs and/or larger reservoirs as briefly discussed in the previous section, the voltage range during charge recycling or recovery phases in a CR buffer’s output rise or fall transitions may encroach further into or even completely overlap with a downstream buffer’s power critical voltage region. To ensure in these cases that the resulting increase in power consumption from short circuit current is minimal, we propose two approaches. The most straightforward approach is to either increase the magnitude of the threshold voltages of the transistors comprising the input stage of routing buffers or reduce the supply voltage, as this minimizes the range of the power critical voltage region. In fact, if $V_{tn} \approx |V_{tp}| \geq V_{DD}/2$, the power critical voltage region may be completely avoided. This approach will require sufficiently increased transistor sizes to compensate for the resulting decrease in performance. In addition, it may be difficult to ensure the power critical voltage region is suitably constrained across all process, voltage, and temperature (PVT) corners.

A second approach to combating short circuit power consumption would be to use a low-power Schmitt trigger topology in place of a conventional inverter for the input stage of the CR buffer. A low-power Schmitt trigger was proposed in [7], where the author described a highly tunable hysteresis characteristic and low-power operation with a short circuit current of less than 200 nA. Fig. 5.10 shows a low-power Schmitt trigger topology inspired by [7] which may act as a input stage for a routing buffer in an FPGA. In the circuit, the HVT header and footer transistors $M1$ and $M5$ are designed to throttle the short circuit current when both $M3$ and $M4$ are on (the length of these transistors may be increased to further reduce short circuit current), while $M2$ and $M6$ are LVT transistors which set the switching points of the Schmitt trigger. Simulations using commercial STMicroelectronics models show that the

![Figure 5.10: Low-power Schmitt trigger.](image-url)
power consumed by a Schmitt trigger-based routing buffer when driven by a slow ramp input (with a slew rate of 1 V/ns) is less than 2% greater than a conventional routing buffer when driven by a fast ramp input (a slew rate of 10 V/ns). We therefore believe that while such a topology is not necessary for the single reservoir charge recycling approach specifically studied in this work, future work which may investigate charge recycling with multiple reservoirs and/or large reservoirs may make use of the proposed topology to ensure short circuit power is kept at a minimum.

The circuitry needed for CR functionality also introduces additional static power overheads. While we attempted to minimize static power by ensuring that most of the leakage paths in the added circuitry are through HVT (high threshold voltage) transistors, the primary leakage path is through transistors $M_{21}$ and $M_{22}$ (between the two output nodes of paired routing conductors), which have standard threshold voltage to optimize performance. Through simulations (at 85$^\circ$C), we determined that the static power overhead of the additional circuitry is between $\sim$0.9-3.4%, depending on the state of the routing switch (and the states of downstream routing switches) in which the added circuitry resides. By averaging across the different routing switch states found in a typical design, the average static power overhead is $\sim$1.9%, which is an acceptable cost given sufficient savings in dynamic power are achieved. While this overhead may be reduced further by replacing $M_{21}$ and $M_{22}$ with HVT transistors, this will result in a slower charge recycling/recovery process and/or a greater area overhead (if one were to offset the performance impact of a larger threshold voltage). Since such design decisions are heavily dependent on the exact leakage position of a process, which can vary widely (between process nodes and/or foundries), we leave the leakage optimization of the circuitry needed for CR for future work.

The simulated delay for our designed CR buffer, when used in CR-mode, is approximately 650ps, which represents a two-fold delay hit over a conventional buffer, based on our simulations. While the delay penalty may appear high, prior work shows that over 70% of interconnect switches can be slowed down by 75% in a commercial 90nm Xilinx FPGA, without any critical path delay increase [14]. When the CR buffer is not used in CR-mode however, the simulated delay penalty is approximately 30ps, which is $\sim$10% of the delay of a conventional buffer. This delay penalty is resulting from the fact that in the input stage of the CR buffer, when operating in normal mode, the gate of $M_{14}$ is charged through two series PMOS transistors ($M_1$ and $M_7$), while the gate of $M_{13}$ is discharged through two series NMOS transistors ($M_2$ and $M_{10}$). Nonetheless, since the majority of the total delay through the buffer is given by the delay of the output stage, the delay penalty at the input stage posed by our buffer design results in a modest overall delay penalty. One final point worth noting is that the delay of the charge recycling/recovery process is subject to variation, which will increase the effective delay penalty when operating in CR mode. As discussed in Section 3.3.3, the effective delay penalty may therefore be addressed to some degree through increased sizing of the most relevant transistors (since for example $\sigma_{V_t} \propto 1/\sqrt{WL}$, where $\sigma_{V_t}$ is the standard deviation of a transistor’s $V_t$, while $W$ and $L$ are a transistor’s width and length, respectively). This is yet another power-area trade-off which exists in the design and optimization of the CR circuitry, and we leave an investigation/optimization of this trade-off for future work.

## 5.4 Tool Support

One of the premises of the proposed power saving technique is that switches which are in CR mode will have adjacent tracks which are unused and thus can serve as reservoirs. Therefore, special effort must

---

1 In other words, a routing buffer whose input stage employs the proposed Schmitt trigger
be made during the physical implementation of a circuit to create opportunities for charge recycling, especially for switches carrying signals with high switching activity (as CR will provide the most benefit for high-activity signals). In order to ensure that CR-mode switches have unused adjacent tracks, the first component of our CR-aware CAD flow is to modify VPR’s router to encourage high-activity signals to be routed using CR-capable switches, and moreover, to see that the reservoirs associated with such switches (in the underlying architecture) remain unused. The baseline router cost function employed by VPR was discussed in Section 4.6.

Our modified cost function is as follows:

\[
\text{Cost}_n = (1 - \text{Crit}_i) \cdot \text{cong}_\text{cost}_n + (1 - \alpha_i) \cdot \text{res}_\text{cost}_n + \alpha_i \cdot (PF \cdot \text{res}_\text{occ}_n + GF \cdot \text{not}_\text{cr}_n) + \text{Crit}_i \cdot \text{delay}_\text{cost}_n
\]

(5.12)

where \(\alpha_i\) is the activity factor for connection \(i\) being routed (normalized to a \([0:1]\) range), \(PF\) and \(GF\) are scalar tuning parameters (determined empirically), and \(\text{res}_\text{cost}_n\) is equal to \((1 - \text{Crit}_j) \cdot \alpha_j\), where \(j\) is the index of the connection currently occupying the potential reservoir of node \(n\). Note that \(\text{res}_\text{occ}_n\) and \(\text{not}_\text{cr}_n\) are binary variables; \(\text{res}_\text{occ}_n\) is equal to 1 if node \(n\) is a CR-capable switch whose reservoir is used and is equal to 0 otherwise, while \(\text{not}_\text{cr}_n\) is equal to 1 if node \(n\) is not CR-capable, and is equal to 0 otherwise.

The motivation for the modification to the cost function is as follows: while routing a circuit, we wish to route connections which have high switching activity and sufficient timing margin (i.e. connections which are favourable candidates to use CR-mode switches) such that they use CR-capable switches whose reservoir tracks are unoccupied. A switch with an occupied reservoir track cannot be put into CR mode, and therefore, such a scenario would lead to a lost opportunity for power savings. In general, when routing connection \(i\), we need to consider two cases: 1) connection \(i\) is a favourable candidate to use CR-mode switches, or 2) connection \(i\) is not likely to use CR-mode switches. For the former case, we aim to penalize routes where reservoir tracks are currently occupied, this is given by the \((1 - \text{Crit}_i) \cdot \alpha_i \cdot PF \cdot \text{res}_\text{occ}_n\) term. For the latter case, we aim to avoid situations where connection \(i\) uses a track which could potentially be a reservoir for another connection \(j\) (which may be a favourable candidate to use CR-mode switches). As such, we penalize this case as well, and this is given by the \((1 - \text{Crit}_i)[(1 - \alpha_i) \cdot \text{res}_\text{cost}_n]\) term. Furthermore, the \((1 - \text{Crit}_i) \cdot \alpha_i \cdot GF \cdot \text{not}_\text{cr}_n\) term acts to penalize nets which are favourable candidates to be put into CR mode, but are not using CR-capable switches.

As mentioned, \(PF\) and \(GF\) are scalar terms that were tuned to yield quality routing results. Figs. 5.11 and 5.12 show the variation in the maximum possible dynamic power savings and critical path delay (assuming that CR mode switches incur no extra delay), respectively, as \(PF\) is varied over a range of values, and set \(GF = PF\) as we found this generally yielded the most favourable results. The results shown in these plots are averaged over the VTR benchmark set. We see that a \(PF\) value of around 40 yields the greatest power reduction (approximately 17%) and virtually no impact to critical path. Again, these plots are only used to tune \(PF\) and serve only as estimates, since putting a routing resource in CR mode will result in a delay penalty (considered fully in Section 6.4).
Chapter 5. Charge Recycling for Power Reduction in FPGA Interconnect

5.4.1 Post-Routing CR Mode Selection

After routing is complete, we perform the final mode selection for each CR-capable routing switch. We formulate the problem of mode selection as a mixed-integer linear program (MILP). Our formulation accepts a user-provided critical path delay constraint as input and optimizes power subject to the constraint by placing an (activity-weighted) maximum number of routing conductors into CR mode.

Let \( G(V,E) \) be the circuit’s timing graph where each vertex, \( v \in V \), represents a routing conductor or pin and each (directed) edge, \( e = (u,v) \), represents a programmable switch or a combinational path through a combinational logic element (e.g. a LUT). Let \( C \) be the subset of \( V \) corresponding to CR-capable routing conductors. For each vertex \( v \in C \), we introduce a binary 0/1 decision variable, \( \gamma_v \), indicating whether \( v \) is in CR mode. Then, the delay of the conductor is:

\[
D_v = D_{Intrinsic_v} + \gamma_v \cdot \delta_v
\]  

(5.13)

where \( D_{Intrinsic_v} \) is \( v \)'s delay when CR is turned off, and \( \delta_v \) is the added delay when CR is enabled. Vertices corresponding to input pins on combinational logic elements are assigned zero delay. We lump the delay through a combinational logic element onto the vertex corresponding to its output pin, however,
it is straightforward to model the case of there being multiple different path delays through the gate. No $\gamma$ variables are introduced for non-CR-capable vertices – such vertices only have intrinsic delays.

Having defined the delays for logic and routing, we introduce constraints specifying the worse-case arrival time at any vertex, $v$, $\text{Arr}_v$. Normally, this is done using a max function over vertices that fanin to $v$ in the timing graph. However, as max constraints cannot be handled directly in MILP, we introduce “greater than” constraints for each of $v$’s fanin vertices:

$$\text{Arr}_v \geq \forall_{(u,v) \in E} \text{Arr}_u + D_v \quad (5.14)$$

where the arrival times for vertices corresponding to primary inputs and flip-flop outputs are initialized to 0. Strict equality is used for vertices having a single fanin vertex in $G$ (as no max function is needed in such cases).

Let $CO$ be the subset of vertices in $V$ corresponding to combinational path end-points, i.e. primary outputs of the circuit and flip-flop inputs. Given a user-provided constraint on the critical path delay, $T$, we introduce the following constraints:

$$\forall_{v \in CO} \text{Arr}_v \leq T \quad (5.15)$$

The objective function of the MILP seeks to maximize the switching activity-weighted number of routing conductors placed in CR mode, and, at the same time, contains terms that cause the arrival time constraints in (5.14) to realize a max function:

$$\phi = \sum_{i \in C} \alpha_i \cdot \gamma_i - \sum_{j \in V} \text{Arr}_j \quad (5.16)$$

where $\alpha_i$ is the switching activity of vertex $i$. The first summation counts the number of conductors in CR mode, where each is weighted by its activity. The second summation, whose sign is negative, ensures that the arrival time assigned for each vertex is as small as possible, yet consistent with the constraints of (5.14), thereby realizing the familiar max function seen in timing analysis arrival time propagation. Note that we scale all delay values to be small enough such that the second summation in (5.16) does not dominate (5.16). The objective function (5.16) and the constraints above constitute an MILP that can be solved with a standard MILP solver (we used the open-source lpsolve ver. 5.5 [4]).

5.5 Experimental Study

In order to assess the merits of the proposed circuitry and CAD flow, we used the set of benchmark circuits packaged with VPR 7.0 [74]; statistics for these benchmarks were provided in Table 4.2. Our baseline non-CR-capable architecture contains unidirectional wire segments (direct-drive) which span 4 CLB tiles, uses the Wilton switch block [100], and has logic blocks with ten 6-LUTs/FFs per CLB. All benchmark circuits were routed on the baseline architecture to determine the minimum number of tracks per channel needed to route each circuit successfully ($W_{min}$). For each circuit, we then computed $W = 1.3 \times W_{min}$ to reflect a medium-stress routing scenario – the standard approach in FPGA architecture research. The computed $W$ value for each circuit was used for all experimental runs of the circuit. We use the ACE switching activity estimator tool [59] to compute switching activity for each signal in each benchmark circuit using vectorless estimation. We make the following assumptions about the architecture in our study: (1) each routing segment is paired with one other routing segment and either can serve as the
reservoir for the other, as depicted in Fig. 5.4, and (2) the paired routing conductors have the same start points but run in opposite directions.

To thoroughly investigate the efficacy of our proposed technique, we first studied the relationship between the fraction of routing switches in an FPGA which are CR capable and the corresponding interconnect dynamic power savings afforded by such an architecture. An architecture which is fully populated with CR capable routing switches will naturally offer greater power reduction than an architecture with a small fraction of CR-capable switches. On the other hand, a routing architecture fully populated with CR capable switches will have greater area overheads than a depopulated CR architecture. As such, by varying the percentage of CR capable switches, we will be able to evaluate the power-area trade-offs – at an architectural level – of the proposed power reduction technique.

We also assess the available power reductions as the performance constraint on the worst-case critical path is varied in the post-routing mode selection phase of our CAD flow (see Equation 5.15). Naturally, relaxing the path constraints will allow more switches to be placed in CR mode, yielding improved power reductions at the expense of speed. Fig. 5.13 shows a diagram over-viewing our experimental flow. The power results correspond to dynamic power consumed in the FPGA interconnect. As discussed previously, we assume that CR mode provides a 26% reduction in switch power consumption and results in a $2 \times$ delay hit compared to a conventional buffer. When a CR buffer is not used in CR mode, we assume a 10% delay hit compared to a conventional buffer, as obtained from simulation results discussed previously.

### 5.5.1 Area Overhead

In addition to assessing the merits of the proposed techniques from the perspective of power reduction, we also studied the impact on area incurred by including CR-capable routing switches. We use the number of min-width transistors as the area metric as this is the area metric used in the VPR place/route tool, and thus, it allows us to find the percentage increase in area (over the baseline architecture) of our CR-capable routing architecture.

The total area overhead of the proposed additional circuitry is equal to 62 minimum-width transistors per pair of CR-switches: 50 minimum-width transistors for the CR circuit, gating circuit, and delay line, and an additional 12 transistors for the two additional SRAM configuration cell needed for CR-mode selection. By adding this area-overhead estimate to VPR’s area model, we obtain estimates on routing area and total FPGA tile area (including soft logic) for each of the benchmark designs; these are tabulated in Table 5.1 where column 5 in the table shows the increase in routing area while column 6 shows the increase in overall area.

Table 5.1 shows that the average increase in area over the set of benchmark circuits studied in this work is 7.8%. With a square tile layout, this corresponds to an increase of $\sim 3.8\%$ (i.e. $\sqrt{1.078}$) in each of the $x$ and $y$ tile dimensions. We believe this is a loose upper bound on the additional capacitance that would arise from having a larger tile, as the portion of total capacitance contributed by attached switches is unaffected by the tile size increase. Therefore the power reduction estimates of our proposed technique which exclude the effect of increased tile dimensions serve as an upper-bound on the achievable power reduction. Similarly, estimates which include the $\sim 3.8\%$ penalty to model the effect of increased tile dimensions serve as a pessimistic lower-bound on the achievable power reduction. We include both estimates in the results presented in the following section.

---

2We considered alternative pairings, however, this scheme produced the best results.
5.5.2 Results

Fig. 5.14 shows the variation in average routing area overhead, total area overhead, and interconnect dynamic power savings (both a lower and upper bound estimates) for the circuits in our benchmark set as the percentage of CR-capable switches for each circuit is varied from 10% to 100%. For the case where the FPGA is fully populated with CR-capable switches (which leads to maximum power reduction), the proposed circuitry leads to an $\sim 17.3\%$ increase in routing area with a dynamic power savings of approximately 11.6-15.4%. One important point to note is that while the CR buffer circuit reduced interconnect dynamic power by $\sim 27\%$, only a fraction of all of the routing conductors can be put into CR mode. Moreover, all of the routing conductors which cannot be put into CR mode incur a dynamic power penalty of around 1%, because of the gating circuitry in the input stage (the CR circuitry and delay line are both disabled when the switch is not in CR mode). This is why the total interconnect dynamic power reduction is diminished to 11.6-15.4%.

As the percentage of CR capable switches is reduced, we see diminished power savings; this is because architectures with a reduced number of CR capable switches offer less opportunities to the router to put CR-favourable nets into CR mode. Nonetheless, it is interesting to observe that the reductions in power savings are not directly proportional to the reduction in CR-capable switches. For instance, as the percentage of CR capable switches is reduced to 50%, the total power savings drops to approximately 7.5-10%; thus there is a $\sim 35\%$ reduction in power savings, but a 50% reduction in area overhead. This encourages us to potentially investigate architectures which have a reduced number of CR capable switches but still offer significant power savings.

Fig. 5.15 shows the variation in power savings vs. the relative critical path (compared to the baseline). The leftmost point of the curves shows the minimum power savings that can be achieved with
Chapter 5. Charge Recycling for Power Reduction in FPGA Interconnect

Figure 5.14: Area overhead and interconnect dynamic power savings vs. percentage of CR-capable switches.

this technique under minimum relaxation of performance constraints. Here, we see a power savings of approximately 11.6-15.4% for an architecture fully populated with CR capable switches (as previously stated). Furthermore, we see that the minimum critical path degradation is 1.3% (relative to baseline) which arises because of the intrinsic delay penalty of CR buffers (not operating in CR mode). However, as timing constraints are relaxed, more switches can be put into CR mode. Those switches used for connections with high criticality and thus could not be put into CR mode under strict timing constraints can now be put into CR mode, and thus additional power savings can be achieved. The results show that an additional ≈3% in power savings can be achieved as the critical path delay is allowed to increase by a factor of 2. Power savings in the FPGA interconnect reach over 18% in this case, which we believe will be useful in power-sensitive applications that do not need to run at high speed.

Finally, the power reductions for an architecture fully populated with CR capable switches and a critical path relaxation factor of 1, are detailed on a benchmark-by-benchmark basis in Table 5.1 (columns 2 and 3). Note that in the table, $\text{TF}$ refers to the critical path relaxation factor; a $\text{TF}$ of 1 implies no relaxation of the minimum critical path achievable in the proposed architecture, while a $\text{TF}$ of 1.5 refers to a 50% relaxation of the minimum critical path.

5.6 Discussion

In the previous section, we described a series of experiments with which we attempt to quantify the merits and costs of the proposed power reduction technique. Our experiments revealed that interconnect dynamic power may be reduced by ≈11-15% under the most stringent timing constraints, and up to ≈14.5-18.5% as timing constraints are relaxed. This comes at an area cost of 7.8%, an interconnect static power cost of ≈1.9% (on average), and a performance penalty of ≈1%. We are encouraged by these results, since as discussed in Section 5.2.1 substantially greater power reduction may be achieved by
pursuing charge recycling implementations which target the use of multiple reservoirs and/or employing reservoirs with greater capacitances. To that end, we discuss an enhancement which may allow us to increase the power savings via charge recycling, and predict possible power savings and costs associated with the enhancement in this section. We also discuss other areas in an FPGA (aside from the interconnect network) where we use charge recycling to reduce power, and again predict achievable power reduction and comment on the associated costs. A detailed study and evaluation of the ideas proposed in this section is left for future work.

As previously discussed, Equation (5.9) shows that the use of multiple reservoirs leads to greater power savings than a single-reservoir approach, regardless of the size of the reservoir capacitance. Theoretically, nearly all of the switching power normally dissipated in the interconnect network may be saved if a sufficient number of reservoirs are employed. However, this approach is likely to be impractical since this will lead to excessively long switching times and will require a significantly large number of routing conductors to be unused so that they may act as reservoirs. Recall that for our proposed power technique to be applied to a particular routing conductor, two conditions must be met: (1) The conductor has an unused “friend” conductor which may be used as a charge reservoir and (2) the conductor must be used by a net with sufficient slack such that the timing penalty incurred by operating in CR mode will not lead to a failure to meet timing constraints. Observe that Fig. 5.15 indicates at most, approximately 18.4% of interconnect power may be saved under the most relaxed timing constraints. Given that a routing conductor which operates in CR mode consumed ~27% less power than a baseline routing conductor, we may estimate that, on average, ~71% of the routing conductors in a design had unused “friend” conductors, such that they could operate in CR mode given sufficient slack. Fig. 5.15 also indicates that approximately 78% of the routing conductors which had unused “friend” conductors could operate in

---

**Figure 5.15:** Interconnect dynamic power reduction vs. critical path relaxation factor. Shown for an architecture fully populated with CR capable switches.
CR mode under the most stringent timing constraints. Together, these two constraints prohibited us from operating \(\sim 45\%\) of the used routing conductors in a design in CR mode, thus limiting the amount of power that could be reduced. Consequently, any technique which has a higher delay penalty and/or requires more unused conductors to act as reservoir will be further handicapped, thus further impeding power reduction.

For this reason, we believe that a better approach to increasing the amount of power that can be reduced through charge recycling would be to increase the capacitance of the reservoir capacitor. In the approach we employed in this work, routing conductors were used as reservoirs, which ensures a 1:1 ratio between the load capacitor being driven by a CR Buffer and its corresponding reservoir capacitor. As per the preceding discussion, using routing conductors as reservoirs may not be the most effective way to increase the capacitance of our reservoir, as this will again require more routing conductors to be unused which, as we have previously established, will likely hinder our pursuit for greater power reduction. An alternative approach is to create a single, very large reservoir with which all the used routing conductors may engage in charge recovery/recycling. If this reservoir has a capacitance sufficiently larger than the sum of the capacitances of all of the routing conductors in an FPGA, we will be able to increase the amount of power that may be reduced. Furthermore, the cost associated with providing this single large reservoir may be amortized over the large number of routing conductors that will make use of it to save power. This global reservoir may be routed alongside the power distribution network and can be strapped with on-chip decoupling capacitors, such as deep-trench capacitors which offer very high capacitance density (\(\sim 200 \, \text{fF/\mu m}^2\)) \[24\]. The reservoir may also be strapped with unused routing conductors to increase its capacitance, which is similar conceptually to a previously proposed approach to increase decoupling capacitance of the power distribution network in FPGAs \[81\]. Alternatively, we may fragment the conventional power distribution network of an FPGA into a primary power distribution network and a globally accessible reservoir. In situations when the power supply is burdened by a high current demand (for example in high speed designs with very high signal activity and resource utilization), we may bolster the primary distribution network with the globally accessible reservoir, in which case we will not be allowed to operate any switches in CR mode\[3\]. This approach will require additional configuration memory in addition to transistors allowing us stitch the globally accessible reservoir to the primary distribution network when needed; however we anticipate the additional overhead to be small.

Fig. 5.16 shows a proposed CR buffer which engages in charge recycling and charge recovery with a globally accessible reservoir (in the figure this is labelled \(V_{\text{res}}\)). Observe that the structure and the circuits employed are largely similar to the the architecture presented in Fig. 5.4, the major difference being that instead of being paired with a another routing switch, each CR buffer has its own Delay Line and CR Circuit. Also note that the input stage of the buffer is a Schmitt trigger – employing the same topology presented in Section 5.3 – because the larger reservoir capacitance will result in the voltage transitions during charge recycling and charge recovery encroaching upon the power critical voltage region of a conventional CMOS inverter. Finally, observe that when \(CR\) is 0 and the switch is unused (input is set and latched to hold \(V_{DD}\)), the buffer serves to bolster the supply rail (the routing conductor acts as a decoupling capacitor).

We anticipate that the proposed enhancements will result in an area overhead of between 36–42

\[\text{This can also be done on a region-by-region basis, meaning that if there are “hot spots” on the chip which have high current demand, the local distribution network in that region may be bolstered by a fragment of the globally accessible reservoir. While this prevents us from operating any switches residing in that region from operating in CR mode, we will still be free to operate switches in other parts of the chip in CR mode.}\]
Figure 5.16: Proposed enhanced CR buffer which has access to a global reservoir.

minimum width transistors per CR buffer, depending on the degree to which the configuration bit which outputs the CR signal may be shared between buffers (for example, we may group sets of CR buffers together to share one CR configuration bit such that either all members of the group operate in CR mode or all members in the group operate in “normal” mode). This represents a total area overhead of between 9-10.6% of an FPGA’s core fabric. The timing penalty is expected to be similar to the circuits we studied in this work, i.e. \(\sim 1\%\). However, we estimate that the enhanced CR buffer will allow for a \(\sim 40\%\) reduction in switching power when operating in CR mode. This estimate is based on a theoretical \(\sim 50\%\) power reduction when using a large globally accessible reservoir, and \(\sim 10\%\) power
overhead arising from the circuits which facilitate charge recycling/recovery, as well as the worst case short circuit current because of the encroachment of the recycling/recovery transitions into power critical regions of downstream routing buffers. Assuming that $\sim 78\%$ of all used routing conductors are able to operate in CR mode, we may estimate that between $\sim 24-31\%$ of the dynamic power dissipated in the interconnect network may be reduced (including the effect of tile-length increases resulting from the area overhead of the proposed circuitry) for an architecture fully populated with CR-capable routing switches and assuming that the most stringent timing constraints are imposed. Again, future work can study this proposal in greater depth to assess the power reductions and area/timing costs of the proposed enhanced charge recycling technique.

In the following sections, we briefly discuss how charge recycling may be employed in the clock networks and embedded memory blocks – known as Block RAM (BRAM) in FPGA parlance – of an FPGA to yield further power reductions. We assume that the globally accessible reservoir which we propose to provide to improve charge recycling in the routing network will also be accessible by the clock network and BRAM circuits.

### 5.6.1 Clock Power Reduction Via Charge Recycling

The use of charge recycling in clock networks has been previously explored, such as in [8], which employed an inductor to periodically recycle charge from the parasitic capacitance of the clock network. However, inductors occupy significant die-area; thus it is desirable to identify more area efficient mechanisms to reduce power. Fig. 5.17 shows a CLB whose conventional clock input buffers have been replaced with CR capable buffers, which are similar to the buffer shown in Fig. 5.16. Observe that the clock buffers which feed each BLE must utilize a Schmitt-trigger to ensure that the the transition time of the clock signal at the input to each flip-flop is within acceptable limits (the transition time of the CLB-level clock network when in CR mode is expected to be very slow).

While the globally accessible reservoir allows for a theoretical power savings of $\sim 50\%$ in the CLB-level clock network, it is challenging to predict the actual amount of power which may be saved after considering the incurred power overheads. A large portion of the total overhead is due to the CR Circuit shown in Fig. 5.1. Similarly, the delay penalty associated with charge recycling is dictated by the sizing of the transistors in the CR Circuit. Observe that the maximum supported frequency of a CR-capable clock buffer when operating in CR-mode is $\sim 1/(2 \cdot t_{CR})$, where $t_{CR}$ is the average delay required to recycle/recover charge. Consequently, if the CR Circuit is optimized for power, it will result in a large value for $t_{CR}$, meaning we may not be able to reduce the power of high frequency clock signals through charge recycling. Optimizing the performance of the CR Circuit on the other hand will lead to increased power overhead. Herein lies a trade-off which needs careful consideration in order to truly optimize the amount of power that may be reduced.

We estimate a relatively small total area increase since each CLB has a small number of clock input buffers (for example Stratix V has 3 separate clock networks in each CLB [12]). Other costs associated with this technique, such as the impact to clock skew and clock uncertainty must be assessed in greater detail which we leave for future work.

---

4 For example, the added delay variability when operating a clock buffer in CR mode may make it difficult to employ this technique for high performance designs because these delay variations lead to clock uncertainty/clock skew. However, as indicated in Section 5.3, this issue may be tackled by increasing the sizes of relevant transistors in the CR circuitry comprising the proposed CR-capable clock buffer. This in turn leads to a trade-off between clock power savings, clock skew/uncertainty, and clock network area overhead (although depending on the architecture, the active area of the clock network may be insignificant compared to the rest of the chip) which needs to be evaluated.
5.6.2 BRAM Power Reduction Via Charge Recycling

Another candidate for power reduction via charge recycling is in the embedded memory blocks, commonly referred to as BRAMs, of FPGAs. BRAMs are effectively SRAM arrays whose width and depth are programmable. One of the most significant sources of dynamic power consumption in an SRAM arises when the bit-lines (BLs) are charged/discharged over the course of a write operation. Charge recycling has been previously proposed to reduce the dynamic power dissipated in SRAM write operations in [55], where the authors proposed a technique to minimize the amount of charge that is drawn from the supply by a set of BLs. One issue with this technique, however, is the fact that the differential voltage between a BL and its complimentary signal, BLB, is less than $V_{DD}$ (the voltage differential is at most $V_{DD}/2$), which negatively impacts the write stability of the cells being written to. As such, the previously proposed write technique employing charge recycling may lead to reduced robustness of the array. We are therefore motivated to pursue a charge recycling scheme which allows for a full rail-to-rail voltage differential during a write operation.

The charge recycling technique we studied in this work therefore appears to be a suitable candidate for power reduction in SRAM arrays since it ensures a BL/BLB differential voltage of $V_{DD}$ volts. We propose the the BL driver and pre-charge circuits shown in Fig. 5.18. Fig. 5.19 shows a timing diagram.

---

5BLs are only partially discharged during a read operation, thus the power dissipated during a write operation is typically considerably greater than the power dissipated during a read operation.
Figure 5.18: Charge recycling-enabled BL Drivers in BRAMs.

Figure 5.19: Timing diagram of signals needed to facilitate charge recycling.

of the signals during a write operation to facilitate charge recycling. Four signals, $\phi_{1a}$, $\phi_{1b}$, $\phi_{2a}$ and $\phi_{2b}$ operate collectively to enable power savings via charge recycling during a write operation. The circuit operates as follows: during a write operation $\phi_{2b}$ transitions to logic-“1” at which point the bitlines are temporarily floating. After a short delay, $\phi_{1a}$ transitions to logic-“0” which initiates charge recovery as
charge flows from one of the bitlines (based on the polarity of $DIN$) to the reservoir $V_{res}$. The timing diagram shows the case where $DIN$ is a logic-“0” over the course of the write operation, consequently $BLB$ remains pre-charged to $V_{DD}$, while $BL$ begins to discharge. After a suitable period of time, $\phi_{1a}$ transitions back to logic-“1” which ends the charge recovery phase; note that voltage of the bitline being discharged is $V_{res}$ volts at the end of charge recovery. The wordline, $WL$, and $\phi_{2a}$ may now immediately transition to logic-“1” to fully discharge $BL$ to $GND$. After the write operation is complete, $\phi_{1b}$ and $WL$ will transition to logic-“0” and we may now precharge both bitlines $V_{DD}$. It is during the precharge phase where we recycle the charge that was recovered previously; this occurs when $\phi_{2a}$ transitions to logic-“1”, allowing charge to flow from $V_{res}$ to the previously discharged bitline; again, at the end of the charge recovery phase the voltage of the bitline is $V_{res}$ volts. After yet another suitable period of time, $\phi_{2a}$ and $\phi_{2b}$ transition to logic-“0”, which allows both bitlines to be fully charged to $V_{DD}$.

Since modern SRAM arrays have self-timed circuitry to generate the various signals needed to orchestrate a read or write operation [36], we anticipate that the additional control signals needed to facilitate the proposed charge-recycling enhanced write operation can be accommodated with minimal additional area and/or power overhead. Given that we are assuming access to a globally accessible reservoir, we estimate a $\sim50\%$ reduction in the power dissipated through the charging and discharging of BLs during a write operation. However, observe that four signals, $\phi_{1a}$, $\phi_{1b}$, $\phi_{2a}$ and $\phi_{2b}$ must be distributed among the BL drivers to stage the various phases of charge recycling; these signals will consume additional power which will degrade the overall power savings. The resulting overhead is difficult to predict, since it depends on the aspect ratio of each BRAM array, whether or not columns are multiplexed, the presence and organization of hierarchical bit lines, the most typical width of data being written into the array, etc. Nonetheless, we anticipate that the capacitance of the bit lines will dwarf the capacitance of the wires needed to route $\phi_{1a}$-$\phi_{2b}$. Thus, we expect a near $\sim50\%$ reduction in write power when using this technique. As indicated in [93], the power dissipation of BRAMs can account for up to $\sim20\%$ of the total core dynamic power dissipated in an FPGA; given that write power is a significant component of the total power dissipated in embedded memory blocks, we anticipate an appreciable reduction in total core dynamic power in FPGAs through this technique.

Finally, as shown in Fig. 5.19, the proposed charge recycling write operation imparts timing overhead from the $\phi_{1a}$ and $\phi_{2a}$ phases, which increases the latency of a write operation. However, the performance impact resulting from the proposed charge recycling write operation is difficult to estimate since it is dependent on the mode of the BRAM (whether the BRAM is in single port, simple dual port, or true dual port mode) and the difference between read and write access delays. For instance, if a single clock is used for both read and write ports, it is likely the maximum operating frequency of the BRAM would be dictated by the read latency, in which case increased write latency may not result in a net performance degradation. While there may be performance degradation in other modes, this may still be acceptable in an FPGA context, since there may be many designs which do not require maximum performance and would benefit from the power savings of the proposed technique; for those that do require maximum performance, the proposed BRAM may be programmed to disable the charge-recycling write operation, thus ensuring write latency is minimized. Again, these factors must be studied in further detail and we leave this for future work.
5.7 Summary

In this chapter, we presented a novel technique which specifically targets the reduction of dynamic power in the routing network of FPGAs. The idea is to leverage the unused routing conductors in the FPGA interconnect as charge reservoirs, which receive charge from neighboring conductors on their falling transitions and which share charge on rising transitions. Ultimately, this reduces the amount of charge that needs to be drawn from the supply rail, saving power. We presented a comprehensive description of the proposed technique, from transistor-level description of necessary circuit structures, CAD flows and algorithms which would optimize the possible power reductions afforded by the proposed technique, and assessed the tradeoffs in area, power, and performance. Results show that charge recycling reduces dynamic power in the FPGA interconnect by up to $\sim$15-18.4%, depending on performance constraints. We also provided insight into potential enhancements of the technique, and briefly described how charge recycling may be applied to other areas, aside from an FPGA’s interconnect network in a bid to further reduce power.
Table 5.1: Power reductions and area overhead for each benchmark in an architecture fully populated with CR-capable switches. TF refers to the timing relaxation factor (a value of 1 indicates timing constraints are not relaxed).

<table>
<thead>
<tr>
<th>Circuit</th>
<th>TF = 1</th>
<th></th>
<th></th>
<th>TF = 1.5</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower Bound Power Reduction [%]</td>
<td>Upper Bound Power Reduction [%]</td>
<td>Lower Bound Power Reduction [%]</td>
<td>Upper Bound Power Reduction [%]</td>
<td>Routing Area Overhead [%]</td>
<td>Total Area Overhead [%]</td>
</tr>
<tr>
<td>bgm</td>
<td>9.2</td>
<td>13.9</td>
<td>14.6</td>
<td>19.3</td>
<td>16.7</td>
<td>9.6</td>
</tr>
<tr>
<td>blob_merge</td>
<td>9.4</td>
<td>13.6</td>
<td>9.5</td>
<td>13.6</td>
<td>17.1</td>
<td>8.6</td>
</tr>
<tr>
<td>boundtop</td>
<td>12.9</td>
<td>16.5</td>
<td>13.3</td>
<td>16.9</td>
<td>16.9</td>
<td>7.4</td>
</tr>
<tr>
<td>ch_intrinsic</td>
<td>13.5</td>
<td>17.0</td>
<td>15.2</td>
<td>18.8</td>
<td>18.6</td>
<td>7.2</td>
</tr>
<tr>
<td>diffeql</td>
<td>13.6</td>
<td>17.4</td>
<td>15.3</td>
<td>19.2</td>
<td>18.5</td>
<td>7.8</td>
</tr>
<tr>
<td>diffeq2</td>
<td>9.8</td>
<td>13.3</td>
<td>14.9</td>
<td>18.4</td>
<td>17.7</td>
<td>7.0</td>
</tr>
<tr>
<td>LU32PEEng</td>
<td>10.6</td>
<td>14.5</td>
<td>14.2</td>
<td>18.1</td>
<td>16.5</td>
<td>8.0</td>
</tr>
<tr>
<td>LU8PEEng</td>
<td>11.3</td>
<td>15.9</td>
<td>10.9</td>
<td>15.5</td>
<td>16.6</td>
<td>9.5</td>
</tr>
<tr>
<td>mcml</td>
<td>10.6</td>
<td>14.1</td>
<td>14.0</td>
<td>17.5</td>
<td>17.8</td>
<td>7.1</td>
</tr>
<tr>
<td>mkDelayWorker32B</td>
<td>12.5</td>
<td>16.5</td>
<td>16.0</td>
<td>20.0</td>
<td>16.4</td>
<td>8.2</td>
</tr>
<tr>
<td>mkPktMerge</td>
<td>10.1</td>
<td>12.9</td>
<td>16.8</td>
<td>19.6</td>
<td>15.8</td>
<td>5.7</td>
</tr>
<tr>
<td>mkSMAdapter4b</td>
<td>12.8</td>
<td>16.5</td>
<td>13.8</td>
<td>17.5</td>
<td>17.1</td>
<td>7.5</td>
</tr>
<tr>
<td>or1200</td>
<td>14.7</td>
<td>18.7</td>
<td>14.4</td>
<td>18.4</td>
<td>16.6</td>
<td>8.2</td>
</tr>
<tr>
<td>raygentop</td>
<td>11.9</td>
<td>15.8</td>
<td>12.8</td>
<td>16.6</td>
<td>17.3</td>
<td>7.9</td>
</tr>
<tr>
<td>sha</td>
<td>12.4</td>
<td>15.9</td>
<td>12.4</td>
<td>15.9</td>
<td>17.1</td>
<td>7.0</td>
</tr>
<tr>
<td>stereovision0</td>
<td>13.2</td>
<td>16.6</td>
<td>14.2</td>
<td>17.6</td>
<td>15.9</td>
<td>6.8</td>
</tr>
<tr>
<td>stereovision1</td>
<td>9.9</td>
<td>14.2</td>
<td>12.0</td>
<td>16.4</td>
<td>16.7</td>
<td>8.9</td>
</tr>
<tr>
<td>stereovision2</td>
<td>10.3</td>
<td>14.9</td>
<td>16.1</td>
<td>20.7</td>
<td>16.6</td>
<td>9.5</td>
</tr>
<tr>
<td>stereovision3</td>
<td>12.0</td>
<td>15.1</td>
<td>16.5</td>
<td>19.6</td>
<td>22.3</td>
<td>6.3</td>
</tr>
<tr>
<td>mean</td>
<td>11.6</td>
<td>15.4</td>
<td>14.0</td>
<td>17.9</td>
<td>17.3</td>
<td>7.8</td>
</tr>
</tbody>
</table>
Chapter 6

Pulsed-Signalling

6.1 Introduction

In Chapter 4, we presented a technique to reduce dynamic and static power in the interconnect network of FPGAs. In this chapter, we direct our attention towards interconnect static power consumption, and investigate an extension to the previously proposed static power reduction technique. The technique presented in Chapter 4 proposed to tri-state unused routing conductors, which not only reduced the static power consumption of unused resources, but also allowed for the reduction of static power of used routing multiplexers. In this chapter, we ask if used conductors may be tri-stated as well when they are not active in the transmission of a signal in a bid to further reduce static power consumption. We call this power reduction technique “pulsed-signalling”, since interconnect buffers “pulse” their intended signal value to a receiving buffer for a short time interval, after which they are tri-stated, thus allowing for leakage reduction. To the extent possible, conductors adjacent to used conductors must be kept unused to mitigate noise and ensure the logic value “captured” by the receiving buffer is not disrupted. Again, this approach relies on noise-reduction support in the routing stage of the FPGA CAD flow. Experiments show that static power in the interconnect of FPGAs may be reduced by \( \sim 37-50\% \) on average, depending on channel width assumptions.

6.2 Pulsed-Signalling for Leakage Power Reduction

Routing conductors in FPGAs are connected to a significant number of routing multiplexers which typically have a large number of inputs to ensure sufficient routing flexibility. For the sake of optimizing area-delay tradeoff, routing multiplexers are usually built with pass transistors in a two-stage topology \[26\]. As we discussed in Chapter 4 these routing multiplexers contain multiple leakage paths, resulting in considerable leakage power being dissipated in FPGA interconnect. In the face of impending issues in the sub-10nm era, such as direct source-drain tunneling (DSDT), it has become critical to find means to reduce the leakage dissipated in FPGA interconnect without sacrificing routing flexibility.

To reduce leakage, we begin by observing that the output impedance of the routing buffers in the interconnect network plays a crucial role in the leakage currents which flow from/to routing MUXes. Conventional routing buffers provide low impedance paths (either from \( V_{DD} \) or to \( GND \)), and it is through these paths that leakage currents flow. If instead we are able to increase the impedance of these paths, leakage current may be reduced. Simply increasing the output impedance of routing buffers is not
an acceptable solution, as this would lead to performance penalties. Note however that a buffer’s output impedance only has to be low when the buffer is in the process of transmitting data (and its output is transitioning); if at all other times, the output impedance is kept high, leakage power will be reduced. To achieve this desired behaviour, we propose to replace conventional routing buffers in FPGAs with routing buffers which are tri-stated when they are not in the process of sending data, and dynamically activated following a transition at their inputs (when they need to send data). In the following sections, we detail the circuit-level modifications and CAD support necessary to implement the proposed leakage power reduction technique.

6.2.1 Pulsed-Signalling Circuit

Fig. 6.1 shows the proposed routing buffer, and as depicted, the buffer can loosely be divided into four stages: receiver stage, gating stage, output stage, and low-leakage keeper stage. Starting at the receiver stage, we note that in contrast to conventional FPGA routing buffers whose first stage consists of an inverter with a level-restoring PMOS transistor connected in feedback, the first stage of our buffer consists of an asynchronous fully-regenerative receiver. A regenerative receiver (which is effectively a latch) is made necessary because routing conductors upstream from the routing buffer exist in a low-leakage mode after transmission of data transitions, and as will be made clear later on in this section, the voltage of the routing conductors will tend to drift away from $V_{DD}$ or $GND$. As such, the burden of retaining state is on the receiver, thus the need for it to be fully regenerative. To eliminate leakage paths, we propose to capacitively couple the output of the routing multiplexer to the input of the receiver with a coupling capacitor $C_C$; this bears some resemblance to previously proposed AC-coupled on-chip interconnect [39], although previous techniques placed the coupling capacitor at the output of a routing driver in an effort to improve performance and reduce signal swing (and thus power). Transitions at $V_{IN}$ inject charge through $C_C$ into $V_A$; if these transitions are large enough, they will flip the state of the regenerative receiver, thus enabling data transmission from $V_{IN}$ through $V_A$ to $V_{INB}$. While many options exist for the implementation of $C_C$, we propose to use a metal-insulator-metal (MIM) capacitor due to its high capacitance density ($\sim$1-10 $fF/\mu m^2$), low series resistance, and the fact that additional active area is not generally required with this option. In a 65nm process, we believe a 5$fF$ MIM cap will be able to fit within the footprint of each routing multiplexer, thus will not requiring any additional area, and can be accommodated with low impact to overall layout and cost.

While a 5$fF$ capacitor is large enough to allow for sufficient signal swing when driving the parasitic capacitance at node $V_A$, we must ensure that the output impedance of the feedback path of the regenerative receiver is very high, otherwise, all of the charge injected into $V_A$ by $C_C$ will be immediately dissipated. This is achieved by applying bias voltages $V_{PB}$ and $V_{NB}$ to the gates of $M_1$ and $M_3$ in a bid to increase the drive resistance of the feedback path which drives $V_A$ (an alternative approach is to employ a combination of long channel devices and/or multiple series connected transistors in the place of $M_1$ and $M_3$). On the other hand, the regeneration time (and thus delay) of the receiver will vary in an inverse manner to the output impedance of the feedback path. To balance these two requirements, a regenerative receiver with hysteresis is employed; we optimized the transistor sizing of this topology to allow for reliable data transmission assuming $6\sigma$ variation in transistor parameters (specifically $V_T$) and $\pm 10\%$ variation in the capacitance of $C_C$.

Immediately following a rising (or falling) transition at $V_{INB}$, the output stage is activated with $M_8$ ($M_7$) turning on while $M_7$ ($M_8$) remains off, allowing for the output $V_{OUT}$ to fall (rise). Observe that immediately following a transition, the outputs of the (non-inverting) delay lines $DL_1$ and $DL_2$ hold the
previous value of $V_{INB}$. Once the rising (falling) transition has made its way through $DL_1$ and $DL_2$, $M_8$ ($M_7$) is shut off, which leaves $V_{OUT}$ tri-stated. As such, for the duration of time following an input transition when the output of $DL_1/ DL_2$ holds its previous value, the output stage is activated and the input transition is allowed to propagate to the output of the buffer. However, once the outputs of both delay-lines have transitioned to their new states, the output stage is tri-stated. This therefore ensures a low-impedance path to either rail when data is being transmitted by the buffer, but a high effective output impedance when the buffer is in a quiescent state. One subtle point to note is that $DL_1$ and $DL_2$ have asymmetric delays: $DL_1$ has a fast rise delay of $\tau_f$ (which in our design was around 25 ps), and a slow fall delay of $\tau_s$ (in our design, this was set to 300 ps), while $DL_2$ has a slow rise delay of $\tau_s$ and a fast fall delay of $\tau_f$. These two asymmetric delay lines are needed to ensure proper functionality in the event of a glitch at the input, as it can be shown that having a single symmetric delay line with equal rise and fall delays $\tau_D$ can lead to improper function if a glitch of pulse-width less that $\tau_D$ were to appear at the input to the buffer. While very specific values have been indicated here, the routing buffer presented in this work is able to tolerate significant variation in $\tau_s$ and $\tau_f$. All that is required is that $\tau_f$ is sufficiently smaller than the output transition time at $V_{OUT}$, which in our experiments was around 180 ps when the buffer is loaded by a length-4 wire, while $\tau_s$ must be sufficiently larger.

Transistors $M_9$-$M_{12}$ realize weak diode keepers which are needed to bound the DC wander at $V_{OUT}$ when the buffer is dynamically tri-stated, as excessive DC wander can lead to errors when transmitting data. This is because downstream receivers are capacitively coupled, and as such, the input transitions that are seen by receivers is proportional to the difference between $V_{DD}$ and the DC wander. Excessive DC wander may diminish the amplitude of signal transitions seen by downstream receivers to the point where signal transitions are no longer propagated. The diode keepers are sized to bound the maximum DC wander such that under $6\sigma$ parameter variation, the worst case signal transition seen at downstream receivers can still be reliably propagated. Moreover, the use of diode weak keepers ensures that leakage paths to either rail are still high impedance.

Finally, an SRAM cell (not shown) outputs signal $S$ and $\overline{S}$ to set the state of the routing buffer;
when $S$ is logic-1, the routing buffer operates in pulsed-signalling mode, allowing us to reduce leakage power, while when the output of $S$ is logic-0, the buffer operates in conventional mode. This SRAM cell is configured when the device is being programmed, and the mode of operation of each routing buffer is determined by an analysis of the worst case injected noise at the buffer output at compile time. We discuss the impact of noise in the following section.

All circuits were designed and simulated with commercial 65nm STMicroelectronics models for functionality verification and power and timing characterization. The dynamic and leakage power overheads arising from the circuitry needed for pulsed-signalling were negligible compared to the leakage and dynamic power dissipated for a length-4 routing wire with the routing multiplexer fan-out arising from a routing architecture consisting of fully populated, Wilton-style switch blocks [100]. On the other hand, there is a small delay penalty when a buffer is used in pulsed-signalling mode, since the inherent DC wander at the output of the buffer will degrade the signal swing seen by downstream routing buffers. Under worst-case conditions (a combination of worst case transistor variation resulting in degraded receiver sensitivity and worst-case DC wander), the delay penalty due to reduced signal swing is approximately 80ps. When the buffer is not used in pulsed-signalling mode, the delay penalty is negligible. As such, in addition to noise considerations discussed below, only conductors on paths with sufficient slack may be used in pulsed-signalling mode.

6.2.2 Noise Considerations

One potential source of concern with the proposed routing buffer is the fact that since the output is tri-stated, the buffer is sensitive to cross-talk. With $V_{OUT}$ tri-stated, capacitive coupling noise from adjacent routing conductors can lead to errors. If the noise exceeds a certain threshold, it may be interpreted as an actual signal transition by downstream routing buffers, since they too are capacitively coupled, and potentially have difficulty distinguishing between noise and actual data. Even if the noise injected into $V_{OUT}$ does not lead to an erroneous data transition on downstream routing buffers, the noise is somewhat residual, since by design there are no low impedance paths to either rail, as such the injected noise will decay at a slow rate. This is a problem as it can lead to a degraded amplitude of valid data transitions at $V_{OUT}$ seen at downstream receivers. In the worst case, the amplitude of such transitions is proportional to $V_{DD} - (|N_{INJ}^{(max)}| + |DCW^{(max)}|)$ where $N_{INJ}^{(max)}$ is the worst case noise injected onto $V_{OUT}$ and $DCW_{MAX}$ is the worst case DC wander. Therefore, for a given design, this power reduction technique cannot be applied to all routing conductors in the circuit in general, but rather only to those routing conductors whose injected noise can be bounded below some maximum threshold level, which in this chapter we call $N_S$ (which for the sake of simplicity is normalized to $V_{DD}$). Let $f_{MUX}(V)$ represent the voltage at the output of a routing multiplexer whose selected input is at $V$ volts. Let $V_{RT}$ represent the minimum voltage amplitude that can be reliably detected by the proposed hysteretic regenerative receiver in the presence of $6\sigma$ parameter variation of its constituent transistors, and let $V_{RR}$ represent the maximum voltage amplitude that can be reliably rejected by the receiver, again subject to transistor variation. We therefore require $f_{MUX}(N_S \cdot V_{DD}) < V_{RR}$ and $f_{MUX}(V_{DD} - N_S \cdot V_{DD} - DCW^{(max)}) \geq V_{RT}$. Through transistor sizing optimization, we found that $N_S \approx 1/4$, given that the routing multiplexers use a boosted gate voltage, which is a common feature in commercial FPGAs [26]. For 65nm, the maximum reliable gate voltage is 1.2 V.

Let $C(i,j)$ represent the coupling capacitance between conductors $i$ and $j$, $C(i)$ the total (plate and coupling) capacitance of conductor $i$, and $F(i)$ the set conductors within the same metal layer and
adjacent to \( i \). Also, let \( \text{occ}(i) \) be equal to 1 if \( i \) is a used conductor or 0 if \( i \) is unused. Therefore, for conductor \( i \) to operate in pulsed-signalling mode, the following condition must hold:

\[
\sum_{j \in F(i)} \frac{C(i, j)}{C(i)} \text{occ}(j) \leq N_S
\]  

(6.1)

In the following sections, we discuss how a conventional FPGA CAD flow may be modified to optimize leakage power by maximizing the number of used conductors which are to operate in pulsed-signalling mode, subject to the constraints described in this section.

### 6.3 CAD Support

As with the power reduction techniques described in Chapters 4 and 5, the leakage reduction technique proposed in this chapter leverages the fact that FPGAs often have many unused routing conductors, and by ensuring specific routing conductors are unused, the worst case noise acting on a used conductor may be reduced, thus allowing the conductor to operate in pulsed-signalling mode. For the remainder of this chapter we define \( F(n) \) to be a set of neighbour conductors of \( n \), i.e. the neighbour conductors which lie on the same metal layer and are adjacent to the \( n \). Thus, to minimize leakage power through our proposed technique, we seek to ensure that a sufficient number of conductors \( j \in F(n) \) are unused, and thus can be used as shielding conductors to allow \( n \) to safely operate in pulsed-signalling mode, as described in Section 6.2.2.

Special effort must be made during the physical implementation of a circuit to ensure that while a signal is routed between source and sink pins, specific routing conductors along the path are unused in order to minimize power. Our proposed CAD flow builds on VTR (a conventional FPGA CAD flow) \[84\], where we have made modifications to the router and added a routing conductor mode selection phase which takes place after routing and timing analysis. These modifications are discussed in the following sections.

#### 6.3.1 Power-Aware Router

The cost function incorporated in VPR’s router was previously described in Section 4.6.1. Our modification of the cost function of the VPR router is as follows:

\[
\text{Cost}_n = (1 - \text{Crit}_i) \cdot [\text{cong}_n \cdot \text{cost}_n + PF \cdot \text{power}_n + AF \cdot \text{power}_n \cdot \text{infringe}_n] + \text{Crit}_i \cdot \text{delay}_n
\]  

(6.2)

where \( PF \) and \( AF \) are empirically determined scalar weighting terms, \( \text{power}_n \) is a term to guide nets to use routing conductors which are likely to operate in a low-power mode, and \( \text{power}_n \cdot \text{infringe}_n \) is used to guide nets away from the “friend” conductors of a potentially used routing conductor which is likely to operate in lower power mode. The term \( \text{power}_n \) is given by:

\[
\text{power}_n = \begin{cases} 
1 & : \sum_{j \in F(n)} \frac{C(n,j)}{C(n)} \text{occ}(j) - N_S > 0 \\
0 & : \text{otherwise}
\end{cases}
\]
where $C(n, j)$, $C(n)$, $occ(j)$, and $N_S$ are as defined in Section 6.2.2. The term $power\_cost\_infringe_n$ is equal to:

$$
\sum_{j \in F(n)} \frac{C(n, j)}{\sum_{k \in F(j)} C(j, k) \cdot occ(k)} \cdot occ(j) \cdot power\_cost_j \cdot (1 - max\_crit_j)
$$

(6.3)

where $max\_crit_j$ is the worst case criticality over all nets currently vying for $j$.

The motivation for the modification to the cost function is as follows: while routing a circuit, we wish to maximize the number of active conductors which can be put into pulsed-signalling state in order to reduce leakage power. In general, when routing connection $i$, we need to consider two cases: 1) connection $i$ has sufficient timing margin, or 2) connection $i$ is timing critical. For the former case, we strive to optimize the routing for two different situations. First, we attempt to guide nets to use segments with a sufficient number of unoccupied neighbours; the term $power\_cost_n$ attempts to optimize for this goal, since it estimates whether or not the noise on conductor $n$ exceeds $N_S$. Any routing conductor $n$ whose $power\_cost_n$ is equal to 0 may be used in pulsed-signalling mode. Conversely, if $power\_cost_n$ is greater than 0, it is unlikely to operate in pulsed-signalling mode, therefore we attempt to avoid using it. The second goal is to avoid injecting excessive noise onto tracks which may otherwise potentially be used in pulsed-signalling mode. This optimization goal is dealt with by the term $power\_cost\_infringe_n$: in evaluating conductor $n$, we examine all neighbours $j$ of $n$ which are used, and assess the likelihood of $j$ operating in pulsed-signalling mode. If $j$ is likely to be used by a low criticality net (given by the $1 - max\_crit_j$ term) and yet has $power\_cost_j > 0$ (i.e. there was an opportunity for $j$ to operate in pulsed-signalling mode and yet it cannot due to excessive noise), we should assign some cost to using $n$. This cost is scaled by $C(n, j) / \sum_{k \in F(j)} C(j, k) \cdot occ(k)$, since the cost of $j$ not operating in pulsed-signalling mode should be distributed over its neighbours and weighted according to the noise-injected by each of $j$’s neighbours. Finally, if the current net has high criticality, then it is unlikely that $i$ can be used in pulsed-signalling mode due to its inherent delay penalty, and as such we opt to focus on optimizing $i$’s delay.

6.3.2 Routing Wire Mode Selection

After routing and timing analysis have completed, we are in a position to determine the operating modes of the routing conductors in the chip. In this case, we seek to optimize leakage power, subject to timing and maximum injected noise constraints. Algorithm[4] provides pseudocode for the greedy approach we used in this study to solve this problem. With the core objective of minimizing leakage power, the goals of the algorithm are to determine which of the unused routing conductors will operate in pulsed-signalling mode, which of the unused routing conductors will act as shields to help isolate used conductors operating in pulsed-signalling mode from adjacent noise sources[4] and which of the unused conductors can be set in low-leakage, tri-stated mode. In the pseudocode shown $noise(i)$ is the worst case total injected noise on conductor $i$, and is equal to $\sum_{j \in F(i)} occ(j) \cdot C(i, j) / C(i)$, $selects(i)$ is the number of MUXes in the fanout of $i$ where $i$ connects to a selected input pin, $slack(G, i)$ returns the timing slack available for the connection associated with routing conductor $i$ given timing graph $G$, $update\_timing\_graph(G, i, t)$ is a routine (not shown for the sake of brevity) which incrementally updates the timing graph $G$ when the delay of the connection associated with routing conductor $i$ is increased by $t$ seconds (i.e. the routine

---

[4]: An unused routing conductor, when driven to a fixed voltage can block the propagation of capacitive noise from switching events. We therefore refer to such conductors as shields as they protect active conductors from such noise.
**Input:** a set $W$ of routing conductors, a timing graph $G(V, E)$

**Output:** a data structure $states(i)$ mapping routing conductor $i$ to its state

1. begin
2. $states(i)$ $\leftarrow$ unknown for all $i \in W$
3. /* Build the set of used wires */
4. $WU \leftarrow \{x \mid x \in W \land \text{occ}(x) = 1\}$
5. /* Build the set of unused wires */
6. $WN \leftarrow \{x \mid x \in W \land \text{occ}(x) = 0\}$
7. /* First determine the state of used conductors */
8. foreach $i \in WU$ in descending order of $\text{selects}(i)$ do
9. if $\text{noise}(i) < N_S$ or $\text{slack}(G, i) < t_D$ then
10. $states(i)$ $\leftarrow$ ungated
11. else
12. $states(i)$ $\leftarrow$ gated
13. /* Update G to reflect state of $i$ */
14. update_timing_graph($G, i, t_D$)
15. /* Compute noise margin */
16. $N_m \leftarrow \text{noise}(i) - N_S$
17. foreach $j \in F(i) \cap WN$ in descending order of $\text{selects}(j)$ do
18. if $C(i,j)/C(i) < N_m$ and $states(j) == \text{unknown}$ then
19. /* conductor $j$ can potentially be gated, don't set its state yet */
20. $N_m \leftarrow N_m - C(i,j)/C(i)$
21. else
22. /* use $j$ as a shield */
23. $states(j)$ $\leftarrow$ ungated
24. end
25. end
26. /* Now finalize the states of unused conductors */
27. foreach $i \in WN$ do
28. if $states(i) == \text{unknown}$ then
29. $states(i)$ $\leftarrow$ gated
30. end
31. end
32. end

**Algorithm 1:** Routing wire mode selection

only updates edges that are affected, and avoids a complete timing analysis). Recall that $t_D$ is the delay penalty of operating in pulsed-signalling mode (as mentioned previously, this is 80ps), and the other terms are as described previously. It should be noted that since it can be shown that typically the input pins of a routing multiplexer which have the greatest leakage current are the selected input pins (the pass-transistors connected to these pins are “on”, thus these pins have low impedance), the worst case leakage power of a routing conductor is approximately proportional to $\text{selects}(i)$. As such, we use this quantity as an estimate for leakage power where appropriate in the algorithm.

The algorithm takes as input the timing graph for the circuit being optimized, and a set $W$ for all the used and unused routing wires in the circuit. The output of the algorithm is the state for each routing conductor $i$, and is equal to either gated or ungated. A used routing conductor will operate in pulsed-signalling mode if its state is gated, and will operate in conventional mode if its state is ungated. An unused routing conductor will be tri-stated if its state is gated, and will act as a shield if its state is ungated. While we wish to maximize the number of used conductors that are operating in pulsed
signalling mode, we also wish to maximize the number of unused conductors in tri-state mode, as they lead to reduced leakage.

The first part of the algorithm determines the state of all the used routing conductors in the circuit. We visit these conductors in descending order of their estimated leakage power (using selects\((i)\) as an estimate). For each routing conductor \(i\), we assess if either the injected noise onto \(i\) is greater than \(N_S\) or if sufficient timing slack is unavailable, in which case \(i\) cannot be used in pulsed-signalling mode, and therefore its state is set to ungated. Otherwise, \(i\) can be used in pulsed-signalling mode and will have its state set to gated, whereafter \(G\) is updated by increasing the delay of the connection associated with conductor \(i\) by \(t_D\) seconds. Following this, we compute the noise margin for conductor \(i\), denoted \(N_m\) in the pseudocode, which is the difference between the total noise injected onto \(i\) and \(N_S\). If this is non-zero, then some of the unused neighbouring conductors of \(i\) may be tri-stated, leading to reduced leakage. We determine which of the unused conductors \(j \in F(i)\) may be potentially tri-stated by visiting them in descending order of their estimate leakage power (again using selects\((j)\) as an estimate), and then assessing if the injected noise onto \(i\) from \(j\) is less than the current value of \(N_m\). If this quantity is less than \(N_m\), then \(j\) may potentially be tri-stated, however we leave its state as ungated, and revisit the conductor after all used conductors have been visited. If instead the noise injected from \(j\) onto \(i\) is greater than \(N_m\) we set the state of \(j\) to ungated, and thus \(j\) will act as a shield.

After the states of all used conductors are determined, we visit each unused conductor \(i\) whose state is still set to unknown. These conductors are either not adjacent to any used conductors, or the determination of their state was deferred until this phase of the algorithm. This would only occur when an unused conductor is adjacent to ungated used conductors and/or gated used conductors with sufficient noise margin. As such, all of these conductors can be tri-stated, meaning \(\text{states}(i) = \text{gated}\).

While the above approach is greedy, as will be discussed in the next section, our results appear to indicate that it works well. Nonetheless a less greedy approach will likely improve the quality of results further, but investigation of such approaches is left for future work.

### 6.4 Experimental Study

In order to assess the merits of the proposed technique, we used the set of benchmark circuits packaged with VTR \([84]\); statistics for these benchmarks were provided in Table 4.2. Our baseline architecture contains unidirectional wire segments which span 4 CLB tiles, uses the Wilton switch block \([100]\), and has logic blocks with ten 6-LUTs/FFs per CLB. Each benchmark circuit \(c\) was routed on the baseline architecture to determine the minimum number of tracks per channel needed to route each circuit successfully \((W_{\text{min}}(c))\); all routing buffers in the baseline architecture are conventional. We subsequently conducted experiments considering two different scenarios; in the first, we set the channel width for each circuit \(W(c)\) to \(1.3 \times W_{\text{min}}(c)\). In the second scenario, we set \(W(c)\) to \(W_{\text{max}}\), where \(W_{\text{max}}\) is the 10% larger than the largest value of \(W_{\text{min}}(c)\) observed across all circuits. This mimics the methodology employed in Chapter 4.

In our experiments, we primarily assess the interconnect static power reduction and area overheads resulting from our proposed technique. The critical path degradation was negligible for this power reduction technique; thus for the remainder of this section, we focus solely on power reduction and area overhead results. The results of our experiments are shown in Table 6.1.
6.4.1 Power Reduction

Our experiments revealed that the proposed pulsed-signalling technique allows for an interconnect static power reduction of $\sim 40\%$ on average under the $W(c) = W_{\text{min}}(c)$ channel width assumption and $\sim 37\%$ under the $W(c) = W_{\text{max}}$ assumption. Interestingly, for most of the benchmarks the average static power reduction for the case when $W(c) = 1.3 \times W_{\text{min}}(c)$ is greater than when $W(c) = W_{\text{max}}$; recall that a similar trend was observed in our study of static power reduction via the methods discussed in Chapter 4. As with those results, the justification for this trend is as follows: for most circuits in our benchmark set, $1.3 \times W_{\text{min}}(c)$ is significantly less than $W_{\text{max}}$, consequently the proportion of routing resources which are unused increases significantly when $W(c)$ is varied between $W_{\text{min}}(c)$ and $W_{\text{max}}$. As the number of unused resources is increased, the impact of routing multiplexer leakage on total interconnect leakage becomes less pronounced. This is why although the probability of operating a routing conductor in pulsed signalling mode increases as channel width increases, the net impact from pulsed-signalling on overall leakage is reduced.

6.4.2 Area Overhead

Table 6.1 provides a circuit-by-circuit breakdown of the routing area overhead incurred by our power reduction technique; results show that on average, we anticipate a $\sim 20-23\%$ increase in routing area, depending on channel width assumptions. Given that for commercial architectures routing area is approximately 50% of total FPGA core area [66], this means that total core FPGA area grows by $\sim 10-12\%$.

It is important to consider the impact which these area increases will have on routing power. Assuming a square tile layout, a $\sim 10-12\%$ increase in tile area will result in tile dimensions growing by $\sim \sqrt{1.12} \approx 5.8\%$. As described in previous chapters, while this analysis may be conservative, we may nonetheless estimate a $\sim 5.8\%$ increase in interconnect dynamic power. However, given the significant power savings from the pulse-signalling technique, a $\sim 5.8\%$ increase in routing dynamic power for a $\sim 40\%$ decrease in routing leakage power is a favourable trade-off. For instance, if we assume that static power is $\sim 30\%$ of total power [40], these numbers indicate that total routing power can still be reduced by $\sim 10\%$. For many designs that run at lower clock rates, routing power will be dominated by leakage, meaning that overall routing energy savings will increase.

6.4.3 Discussion

In this chapter, we presented a technique wherein both used and unused routing buffers would be gated in a bid to reduce routing leakage power, as an extension of the idea presented in Chapter 4. Used routing buffers would be temporarily ungated when in the process of transmitting signal transitions to downstream circuits, but would otherwise be in a gated, low-leakage state. Our experimental study revealed that routing leakage could be reduced by $\sim 40\%$ at an total area cost of $\sim 10\%$. While these results would typically be encouraging given the significant leakage reduction, in light of the results presented in Chapter 4, we must conclude that this approach is sub-optimal given that greater leakage reductions were afforded by the various circuit-level and routing techniques presented in Chapter 4 at a significantly smaller area cost. The Achilles heel of the pulsed-signalling technique lies in the fact that the routing conductor may only safely operate in low-leakage mode if neighbouring routing conductors are unused and act as shields to guard the dynamically-gated routing conductor from noise. While we employed router cost-function modifications to route signals in a manner which would foster such
arrangements, our results showed that in general it was difficult to ensure that a sufficient number of used routing conductors met the necessary conditions such that meaningful power reductions could be realized. Moreover the routing conductors which are used as shields, while unused, cannot be gated (since then they would not be able to shield a vulnerable routing conductor from the noise injected by other neighbouring conductors) and thus are not in a low-leakage state. This therefore reduces the effective leakage reduction of the pulsed-signalling technique relative to the techniques presented in Chapter 4 where all unused routing conductors may unconditionally operate in a low-leakage state. This combination of a greater difficulty in ensuring used routing conductors may operate in a low-leakage mode and a degradation in the amount of leakage that may be reduced in unused resources handicaps the amount of effectiveness of this technique. The complex nature of pulsed-signalling also makes this technique less attractive; pulsed-signalling attempts to minimize power by dynamically shutting off routing buffers after a signal transition has propagated successfully through a switch whereas the techniques in Chapter 4 make use of the inherent flexibility of an FPGA’s routing architecture and simple, relatively low cost transistor-level optimizations. Thus, while this study has shown that pulsed-signalling may not be the most effective means of reducing leakage power in FPGA interconnect, it has highlighted the efficacy of the techniques we presented in Chapter 4.

6.5 Summary

This chapter presented a novel leakage reduction technique for FPGA interconnect where routing buffers were dynamically gated when they are not actively transmitting signal transitions; we called this technique pulsed-signalling. We presented circuits, described the conditions required to enable pulsed-signalling, and presented a full CAD flow to support the proposed technique. An experimental study was also conducted and described, the results of which showed that the proposed technique could reduced interconnect static power by $\sim 37-40\%$ at an area cost of $\sim 10\%$. We then contrasted these results to the results presented in Chapter 4 and concluded that the set of techniques described in Chapter 4 represent a more effective means of reducing static power in the interconnect of FPGAs.
<table>
<thead>
<tr>
<th>Circuit</th>
<th>( W = 1.3 \times W_{\text{min}(c)} )</th>
<th>( W = W_{\text{max}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Leakage Reduction [%]</td>
<td>Routing Area Overhead [%]</td>
</tr>
<tr>
<td>alu4</td>
<td>41.4</td>
<td>19.5</td>
</tr>
<tr>
<td>apex2</td>
<td>40.4</td>
<td>20.7</td>
</tr>
<tr>
<td>apex4</td>
<td>39.8</td>
<td>20.6</td>
</tr>
<tr>
<td>bgm</td>
<td>40.4</td>
<td>21.9</td>
</tr>
<tr>
<td>bigkey</td>
<td>41.7</td>
<td>20.6</td>
</tr>
<tr>
<td>blob_merge</td>
<td>42.0</td>
<td>20.9</td>
</tr>
<tr>
<td>boundtop</td>
<td>41.8</td>
<td>20.7</td>
</tr>
<tr>
<td>ch_intrinsics</td>
<td>41.3</td>
<td>20.0</td>
</tr>
<tr>
<td>clma</td>
<td>43.3</td>
<td>22.0</td>
</tr>
<tr>
<td>des</td>
<td>42.0</td>
<td>20.1</td>
</tr>
<tr>
<td>diffeq</td>
<td>41.5</td>
<td>20.3</td>
</tr>
<tr>
<td>diffeq1</td>
<td>42.3</td>
<td>20.6</td>
</tr>
<tr>
<td>diffeq2</td>
<td>38.0</td>
<td>21.2</td>
</tr>
<tr>
<td>dsip</td>
<td>43.0</td>
<td>19.8</td>
</tr>
<tr>
<td>elliptic</td>
<td>42.0</td>
<td>21.2</td>
</tr>
<tr>
<td>ex1010</td>
<td>41.1</td>
<td>21.2</td>
</tr>
<tr>
<td>ex5p</td>
<td>39.1</td>
<td>20.7</td>
</tr>
<tr>
<td>frisc</td>
<td>41.7</td>
<td>21.1</td>
</tr>
<tr>
<td>LU32PEEng</td>
<td>38.0</td>
<td>22.8</td>
</tr>
<tr>
<td>LU8PEEng</td>
<td>41.0</td>
<td>22.1</td>
</tr>
<tr>
<td>mcml</td>
<td>42.0</td>
<td>21.8</td>
</tr>
<tr>
<td>misex3</td>
<td>40.8</td>
<td>20.3</td>
</tr>
<tr>
<td>mkDelayWorker32B</td>
<td>40.0</td>
<td>21.5</td>
</tr>
<tr>
<td>mkPktMerge</td>
<td>41.8</td>
<td>20.3</td>
</tr>
<tr>
<td>mkSMAdapter4B</td>
<td>39.2</td>
<td>20.7</td>
</tr>
<tr>
<td>or1200</td>
<td>40.8</td>
<td>21.2</td>
</tr>
<tr>
<td>pdc</td>
<td>41.4</td>
<td>22.0</td>
</tr>
<tr>
<td>raygentop</td>
<td>39.2</td>
<td>20.9</td>
</tr>
<tr>
<td>s298</td>
<td>41.0</td>
<td>20.1</td>
</tr>
<tr>
<td>s38417</td>
<td>42.7</td>
<td>20.3</td>
</tr>
<tr>
<td>s38584.1</td>
<td>41.0</td>
<td>20.5</td>
</tr>
<tr>
<td>seq</td>
<td>41.0</td>
<td>20.7</td>
</tr>
<tr>
<td>sha</td>
<td>41.4</td>
<td>20.3</td>
</tr>
<tr>
<td>spla</td>
<td>41.7</td>
<td>20.9</td>
</tr>
<tr>
<td>stereovision0</td>
<td>40.0</td>
<td>20.7</td>
</tr>
<tr>
<td>stereovision1</td>
<td>34.0</td>
<td>22.0</td>
</tr>
<tr>
<td>stereovision2</td>
<td>41.1</td>
<td>22.7</td>
</tr>
<tr>
<td>stereovision3</td>
<td>42.0</td>
<td>21.0</td>
</tr>
<tr>
<td>tseng</td>
<td>42.2</td>
<td>20.3</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td><strong>40.9</strong></td>
<td><strong>20.9</strong></td>
</tr>
</tbody>
</table>
Chapter 7

Conclusion

7.1 Summary and Contributions

Power reduction has become an increasingly important thrust in the development of modern digital systems. The growing demand for energy efficient computing has been driven by recent trends in the use and proliferation of mobile electronic devices, as well as advances in burgeoning fields such as machine learning and augmented/virtual reality which have spurred a wide range of revolutionary, and yet power-hungry applications. As such, FPGAs must also improve in energy efficiency if they are to participate in these new avenues for growth of the industry. This issue is compounded by the fact that continued process scaling inevitably leads to greater challenges in achieving power efficiency while maintaining adequate levels of performance. One example is the emergence of direct-source-to-drain tunneling, which is expected to be a pivotal issue in sub 7-nm processes. These factors have motivated the need to devise a wide range of approaches to reduce power consumption.

To that end, this dissertation presented several techniques to improve the energy efficiency of FPGAs at the circuit and architectural level, and also presented the associated CAD tools/flows needed to optimize designs for the proposed power reduction techniques. The contributions of this thesis are:

- In Chapter 2 we discussed power dissipation in FPGAs and provided an overview of a wide range of different techniques that have been proposed to reduce FPGA power. This chapter is derived, in part, from a chapter we authored in the book *Reconfigurable Logic: Architecture, Tools, and Applications* published by CRC press [41].

- In Chapter 3 we proposed a technique to minimize the power dissipation which arises because of glitches. We showed that previous approaches to glitch power reduction in FPGAs which attempted to eliminate glitches by equalizing path delays was fundamentally flawed because of PVT variations. We proposed a novel circuit which can be calibrated to filter out glitches which have a wide range of different pulse-widths, and proposed to augment conventional BLE’s with the proposed circuitry. The optimization of glitch power using our proposed circuitry in non-trivial, as such we developed a custom CAD flow to maximize glitch power reduction under timing constraints. Our experiments showed that glitch power could be reduced from \(\sim 60-70\%\), depending on circuit parameters. This work has been published in [46].

- In Chapter 4 we showed that conventional FPGA architectures have an abundance of unused
routing conductors for typical designs. These unused routing conductors, when tri-stated, allow for a reduction in both dynamic and static power in an FPGA’s interconnect network. We therefore propose two low-cost/low-power tri-state routing buffers to implement our proposed routing power optimization techniques. We also propose enhancements to a conventional FPGA router to maximize the amount of power that may be reduced through our proposed techniques, while ensuring a minimal impact to circuit performance. This work has been published in [43] and is currently under review in [44].

- Chapter 5 presented a second technique to leverage the unused routing conductors in FPGA’s. We proposed a charge recycling technique to minimize dynamic power dissipated in the interconnect network of FPGAs; the unused routing conductors in this case serve as charge reservoirs to facilitate charge recycling. Again, in addition to the necessary circuitry required to enable the proposed technique, we also present a CAD flow consisting of a modified FPGA router in addition to a post-routing mode selection pass which maximizes the power reduction through charge recycling while meeting timing constraints. This work has been published in [42].

- Chapter 6 describes another static power reduction technique which extends the concepts presented in Chapter 4. Here, we proposed to dynamically tri-state used buffers whenever they were not actively transmitting data. The challenge with this particular technique was the fact that used buffers which are dynamically tri-stated are susceptible to noise-induced bit errors; as such, this prevents us from applying the technique to every used routing conductor, but rather, only to those conductors whose worst-case coupling induced noise was less than a specific threshold. We therefore developed a CAD flow consisting of a modified FPGA router and a post-routing mode selection pass which attempt to maximize the number of used routing conductors which may be safely operated in a dynamically tri-stateable mode. While a considerable amount of leakage power could be reduced through this technique, we showed that overall the amount of power that could be saved using the techniques discussed in Chapter 4 was far greater in comparison. This work was published in [45].

7.2 Future Work

The proposed power reduction techniques presented in this thesis serve as a foundation upon which we anticipate future research will be able to build on. In the following sections, we propose enhancements and areas of future study for each of these techniques. We also discuss how the techniques may be combined to yield increased power reduction in aggregate.

7.2.1 Glitch Reduction

As indicated in Chapter 3, the fanin delay push-out balancing scheme used in the work is a conservative approach to glitch power optimization, and indeed compromises to some degree our ability to remove glitches. Future work will investigate alternative approaches and/or computationally efficient methods to detect situations in which fanin delay balancing can be neglected, even for what we call consequential nodes. Development of compact and accurate models which allow us to predict glitch statistics given the relative arrival times and switching statistics at input signals would serve this purpose well, and so this would be a promising avenue for future research. Along the same lines, our proposed CAD flow formulated the glitch optimization problem as an MILP. While an MILP is able to produce high quality
results for combinatorial optimization problems, they are associated with a significant run-time penalty. As such, another direction of future work will be to develop run-time efficient heuristics which may solve the optimization problem without appreciably sacrificing the quality of results. We also discussed two techniques to extend the practicality of a MILP based glitch optimization algorithm – namely early solver termination and circuit partitioning – which can be experimented with as well.

Since our proposed glitch filtering technique allows for glitches to be suppressed under varying PVT, yet another direction for future work will be to explore the problem of multi-corner glitch power optimization, thus yielding a truly comprehensive glitch power reduction technique. This primarily involves an enhanced CAD flow which must first characterize glitch power distributions across while taking PVT variation into account, and then to find the optimal glitch filter settings to minimize glitch power across a wide range of variation in PVT.

7.2.2 Energy Optimization of Unused Routing Resources

While Chapter 4 investigated power reduction of FPGA interconnect with the use of tri-state routing buffers, there is also the possibility that a more flexible (and thus routeable) interconnect network may be built using tri-state buffers. In fact, tri-state routing buffers were commonly used in FPGA interconnect [19] but were replaced by direct-drive routing due to their superior overall area-delay product [63]. Nonetheless, modern FPGAs such as Altera’s Arria 10 have reintroduced tri-state routing buffers in parts of their interconnect network to improve utilization of their long-lines which has a positive impact on routeability [96]. The power reductions which we have shown are possible with the use of tri-state routing buffers provide a compelling argument for their re-introduction into the interconnect networks of modern FPGA architectures in their own-right; the potential improvements in routeability strengthen that argument.

One such avenue for future work which can build on the contributions of this chapter would be to explore how an FPGA’s routing architecture may be enhanced from a routeability point of view with tri-state routing buffers. The enhanced flexibility may also lead to further power optimization for two reasons:

1. A more flexible interconnect network will likely lead to a reduction in the total wire-length required to route a given design. A reduction in routed wire-length will lead to reduced capacitance which needs to be switched, thus benefiting dynamic power consumption. This also means the number of used routing buffers will be reduced, which has a positive impact on static power consumption, since in our proposed technique unused routing buffers may be put in a low-leakage state (i.e. tri-stated).

2. Greater interconnect flexibility may also make it easier for us to guarantee that routing conductors used by high-activity nets are adjacent to unused routing conductors. Recall that we modified VPR’s router such that nets which had high-activity would attempt to use routing conductors which had unused neighbouring conductors. Similarly, the router would attempt to steer nets with low activity away from conductors that neighboured conductors used by high-activity nets. A more flexible routing architecture will likely ease the burden the router faces in trying to optimize for power in this manner. This also applies to the router modifications which sought to minimize static power dissipated in routing multiplexers.

We therefore anticipate that this will be an interesting and fruitful direction for future research.
7.2.3 Charge Recycling

In Section 5.6, we presented in great detail future research directions which may serve to improve the interconnect dynamic power reduction possible through charge recycling, and also presented in detail how charge recycling may be extended to other parts of an FPGA – namely the clock network and block RAM – to yield further power reductions. We are encouraged by the preliminary analysis presented in that section of how much power may be saved, and expect that a thorough evaluation of these ideas will yield favourable results. One point which was not mentioned in Chapter 5 is the fact that the MILP based mode selection of routing buffers suffers from very long run-times. As with our recommended future work to extend the glitch reduction technique, we believe future research which seeks to develop heuristics which provide high quality solutions to the mode selection problem in a run-time efficient manner will be beneficial.

7.2.4 Pulsed Signalling

Our experiments revealed that the power reduction afforded by the pulsed-signalling technique was inherently inferior to that achieved through the techniques presented in Chapter 4. However, one aspect which merits further investigation is incorporating the router cost function modification in Chapter 4 which seeks to route signals away from the leaky pins of used routing multiplexers. We believe this will help improve the static power reductions of this technique. That being said, the greatest shortcoming of this technique is the fact that a buffer which is tri-stated is vulnerable to noise, and the necessary safeguards which need to be put in place to ensure bit-errors are prevented is a significant obstacle which degrades the amount of leakage power which may be reduced. As such, circuits and/or routing architectures which can ensure such noise-induced data hazards can be deterministically prevented would help to greatly enhance the power reduction possible through this technique. We believe this is an essential future research direction to improve the viability of this technique.

7.2.5 Combining Power Reduction Techniques

The power reduction techniques proposed in this thesis are generally complementary to existing power reduction techniques such as power gating, clock gating, dynamic voltage and frequency scaling (DVFS), multi-\(V_{DD}\) approaches, and a host of other techniques as well. An analysis of how the proposed techniques may be combined with other techniques would be an interesting subject for future study. However, it may also be useful to consider how each of the proposed techniques may be combined with one another in a bid to further improve on power reduction results. Consider, for example, the techniques presented in Chapter 4; these power reduction techniques are largely independent to both the glitch reduction technique and the charge recycling technique. Consequently, optimization of interconnect dynamic and static power with tri-state routing buffers may be combined with either the proposed glitch reduction technique or the charge recycling technique\(^1\) to yield substantial dynamic power savings.\(^2\) Conversely, the glitch reduction technique and the charge recycling technique are not independent power reduction approaches since they both trade-off power savings with the available slack; these two techniques will therefore compete with one another. As such, the manner in which these two power reduction techniques should be combined to truly optimize energy efficiency is non-trivial and would require further study.

\(^1\)In fact, the charge recycling technique already incorporates circuitry which allows the routing buffers to be tri-stateable. Consequently, the charge recycling technique may be augmented with the techniques described in Chapter 4 with virtually zero additional cost.

\(^2\)We anticipate that the resulting power savings will simply be the sum of the individual power savings of each technique.
Nonetheless, we believe considerable improvements to energy efficiency may be achieved by combining any number of the proposed techniques.

7.3 Conclusion

In this thesis, we strove to develop low-cost techniques to attack various sources of power consumption in FPGAs. We are encouraged overall by the increases in energy efficiency achieved by each of our techniques and are also optimistic about the prospects for further power savings by the enhancements to each technique which we have left for future work. FPGA power reduction remains an open and challenging problem. While a wide range of measures to reduce power in FPGAs have been proposed over the last 15 years, only a small fraction of these techniques have appeared in commercial products. This is largely because of the non-negligible area and performance cost associated with the vast majority of power reduction techniques. While improving power efficiency will enable FPGAs to enter new markets, vendors have traditionally been very careful in ensuring that their competitive position in existing markets is not compromised by any effort to enter a new one; this has resulted in an aversion to implementing aggressive power reduction techniques in their products. However, as has often been stated in this thesis, new and very lucrative markets which have formed in recent times may encourage vendors to rethink this strategy. We are therefore hopeful that the work presented in this thesis will help vendors take the steps needed to pursue energy reduction in their products.
References


147


