Circuit Solutions and Tool Flow of Ultra-Low-Power FPGAs

A Dissertation

Presented to

the faculty of the School of Engineering and

Applied Science

University of Virginia

in partial fulfillment
of the requirements for the degree

Doctor of Philosophy

by

He Qi

December 2017

APPROVAL SHEET

This Dissertation
is submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Author Signature: [illegible]

This Dissertation has been read and approved by the examining committee:

Advisor: Benton H. Calhoun

Committee Member: Ronald Williams

Committee Member: Gabriel Robins

Committee Member: Steven Bowers

Committee Member: Kamin Whitehouse

Committee Member: Benton H. Calhoun

Accepted for the School of Engineering and Applied Science:

[illegible]

Craig H. Benson, School of Engineering and Applied Science

December 2017

Abstract

Low-power miniature systems for ubiquitous computing such as wireless sensor networks have been developing rapidly in the past years. The growing demand on collecting and analyzing information from surrounding environment drives researchers and engineers to develop Internet-of-Things (IoT). This trend requires future integrated circuits for IoT devices to be ultra-low-power (ULP), flexible, and low-cost. Existing circuit solutions of IoT devices are either too costly such as sub-threshold ASICs, or too power-consuming such as sub-threshold microprocessors. ULP FPGAs operating in near/sub-threshold region, flexible and much lower-power than sub-threshold microprocessors, become a promising hardware solution for IoT applications. In this dissertation, circuit/architecture and tool flow of a custom ULP FPGA are explored and developed.

1) Energy Efficient FPGA Interconnect

The global interconnect is the major power consumer of the core fabric of FPGAs. Studies have shown that over 65% of power is dissipated in the interconnection fabric. The same conclusion applies to delay and area. The strict requirements on both speed and energy of IoT applications make energy reduction and energy-efficiency improvement of FPGA routing fabrics a driving challenge. In this dissertation, an energy-efficient low-swing interconnect is modeled, optimized, and evaluated in near/sub-threshold region. When implementing Microelectronics Center of North Carolina (MCNC) benchmarks, the proposed interconnect leads to 68.4% delay reduction and 47.5% energy reduction compared to prior works.

2) Per-Path Voltage Scaling and Power-Gating

Per-path voltage scaling is a technique to reduce FPGA energy to just the minimum while maintaining the overall FPGA speed by reducing the supply voltage on non-critical paths. However, no existing work applied it to FPGA interconnect due to large area overhead. In this dissertation, this problem is solved by using the low-swing interconnect. When using this technique along with power-gating, a 22.3% - 56.5% energy reduction is observed. A custom low-power FPGA is fabricated and measured. When implementing a 4-bit adder, it consumes 277x lower power and 3.4x lower energy than Microsemi IGLOO, which is the most low-power FPGA in the market today.

3) Low-Power FPGA Evaluation Platform

To evaluate custom FPGAs, full tool flow and benchmark support are needed. However, existing commonly used FPGA benchmarks are either too large for ULP FPGAs or too simple to fully utilize ULP FPGA resources. Also, the existing benchmark synthesis tools either only support commercial FPGA architectures, or have crucial limitations on the syntax of input Verilog. In addition, the existing power estimation tools do not accurate for low-power FPGAs and the embedded accelerators/IPs. Solutions to those problems are addressed in this dissertation. The functionality of the custom flows has been verified using a custom low-power application suite. Compared to Microsemi IGLOO, the custom FPGA on average consumes 315x less power and 75x less energy when implementing the custom low-power FPGA application suite.

Acknowledgement

I firstly want to thank my parents, who give me their utmost support and trust. When I make decisions that I think will surprise them, they always choose to support me with smiles and jokes. Research life is both enjoyable and hard. They completely understand me when I struggled with my research progress. I can't remember how many time they said "we don't expect you to be excellent, but just wish you could happy". I did fell warm and got back to my fighting mode. Thank you! Now we video chat on the phone to share each other's life, but I really miss them.

During my second year of PhD, I met Xiaoyu. She is beautiful, smart, kind, and she is my wife now. These years, I spent a big portion of my spare time on research and learning new skills. I very much appreciate her support and understanding on this. As a PhD of electrical engineering, Xiaoyu supports me not only in the way of family, but also technically. When I have troubles with my research, I am always inspired by talking to her. Xiaoyu and I raised a cute golden retriever Candy. We both amused by her when we disappointed about unexpected failures in research, and finally get pumped again.

I want to thank professor Benton Calhoun, who is my research adviser at UVa. Ben is always generous at giving chance to students who interested in his research. I joined UVa as a teaching assistant. Back then, I was a bad English speaker and not good at expressing my thoughts. Ben encouraged me and agreed to let me join Bengroup. I was so glad to have this opportunity to prove myself and become a research assistant in Bengroup to work on the interesting projects. I learnt a lot from Ben, especially research methodology and project management. I was focused on detailed research questions, but Ben trained me to think broader on the high level. This exercise is so valuable that I can benefit from it for my entire life. I very much appreciate for everything.

When I applied to UVa PhD program, I was attracted by Bengroup website, which made me think this is a professional, friendly, and energetic group. It turned out I was wrong. The fact is Bengroup is even better! The team is very like to share and help each other. I especially want to thank Dr. Oluseyi Ayorinde. Seyi and I worked on the same project, but he joined the group one year before me, which means he knows everything I don't know and have to know. Seyi never hesitates to help and always provide more than my expectation. Even years later, Seyi still always come to my cubic to check if I need any support. It's my pleasure to work with you! Other current and former students (Yousef, Yanqing, Aatmesh, Kyle, Peter, Jim, Alicia, Yu, Patricia, Manula, Farah, Chris, Dilip, Kevin, Arijit, Divya, Abhishek, Harsh, Ningxi, Shuo, Daniel, Jacob, Henry, Rishika, Sumanth, etc.) are also very kind and friendly. I want to thank all of you for the help and being around these years. I am so glad to know you. I also want to thank other faculty, Staff, and my friends who helped me and shared time with me at UVa.

List of Figures

Contents

1 Introduction

1.1 Motivation of Low-Power FPGAs

Low-power miniature systems for ubiquitous computing such as wireless sensor networks have been developing rapidly in the past years. The growing demand on collecting and analyzing information from surrounding environment drives researchers and engineers to develop internet of things (IoT). This trend leads to new requirements on integrated circuits. Firstly, due to the potential large number of IoT devices in the future, frequently changing battery for all of them becomes unfeasible. Furthermore, many sensors such as the ones equipped on human or deployed in the environment desire light weight, high portability, and wireless communication ability. All these demands require hardware used in IoT applications to be ultra-low-power (ULP). Secondly, due to the rapidly changing demands and technologies today, these hardware needs computational flexibility for regularly algorithm upgrade or functionality change. Finally, since the cost is potentially high to design sensors using in tons of different applications, low-cost hardware is highly recommended.

Existing hardware solutions for ubiquitous computing include ULP ASICs and ULP microprocessors working in sub-threshold region. However, the development of sub-threshold ASICs for IoT applications is costly and time-consuming due to high design complexity. On the other hand, sub-threshold microprocessors are too power-consuming. FPGAs operating in near/sub-threshold region, flexible and much lower-power than sub-threshold microprocessors, become a highly desired hardware solution for IoT applications. However, comparing to functionality equivalent ASICs, FPGA implementations consume 7x - 14x more power, and 4x slower [4]. This gap makes it not easy for FPGAs to meet energy and performance requirements of IoT applications at the same time. To widely use FPGAs in IoT, energy reduction and energy-efficiency improvement are critical.

1.2 Thesis Statement

By scaling the supply voltage down to near/sub-threshold region, optimizing the circuit and architecture of the interconnect fabric, and applying per-path voltage scaling and power-gating to the interconnect, the ULP FPGAs become an energy-efficient hardware solution for IoT devices or other low-power applications that require well-balanced flexibility, cost, and energy-efficiency. The custom tool set and application suite enable fast and effective evaluation of ULP FPGAs.

1.3 Goals

The major research goals of this dissertation include:

- Optimize ULP FPGA interconnect circuit, supply voltage, and physical implementation towards maximum energy-efficiency
- Apply per-path voltage scaling and power-gating to the interconnect of ULP FPGAs
- Develop an ULP FPGA evaluation platform (including a low-power application suite, a Verilog synthesis flow, and a fast power estimation flow)
- Enable the tool support of evaluating the effects of using embedded hard IPs on ULP FPGAs

2 Background

2.1 General FPGA Architecture

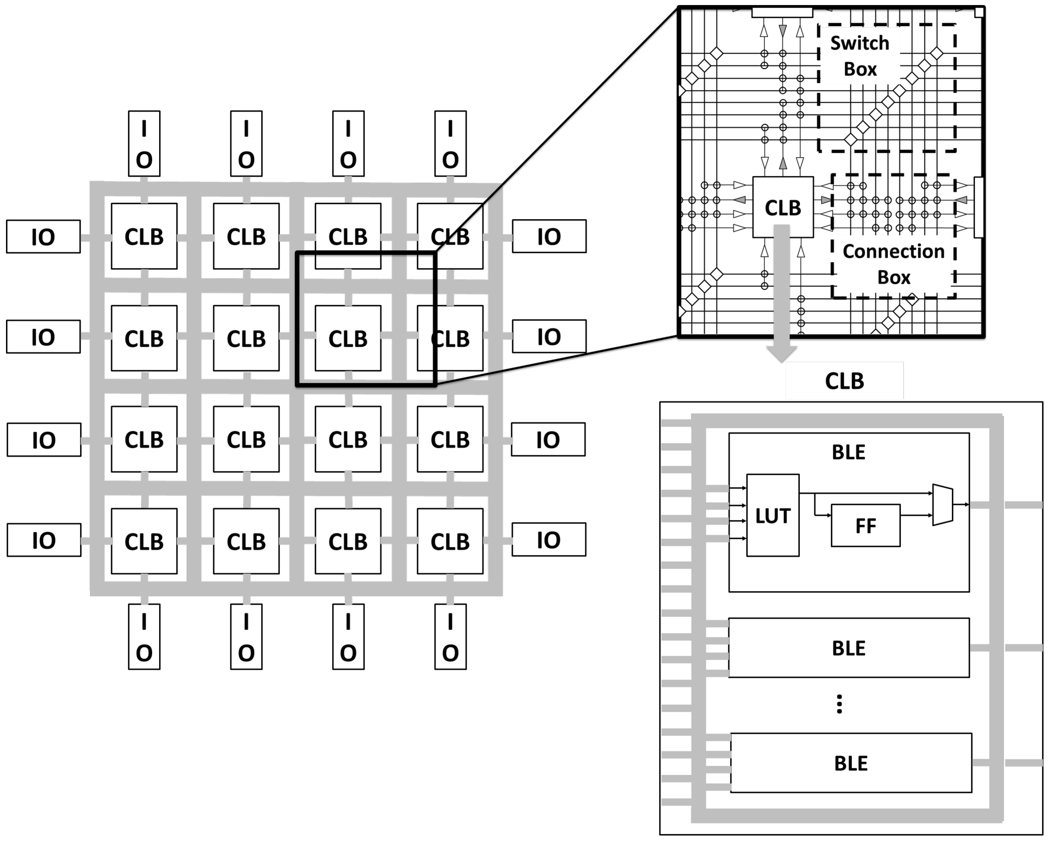


Figure 1: Basic architecture of the core fabric of FPGAs

FPGAs are integrated circuits that can be configured to different logic functions as needed after manufacturer. The basic structure of the core fabric of FPGAs includes Configurable Logic Blocks (CLB), global interconnect, and IO blocks. Each CLB consists of multiple Look-Up-Tables (LUT), Flip-Flops (FF), and multiplexers (MUX), while all CLBs and IO blocks are connected by the global interconnect. By changing the values in configuration bits that distributed all over the FPGA, users can determine the logic function of CLBs, the connectivity pattern between CLBs, and whether the IOs are inputs or outputs.

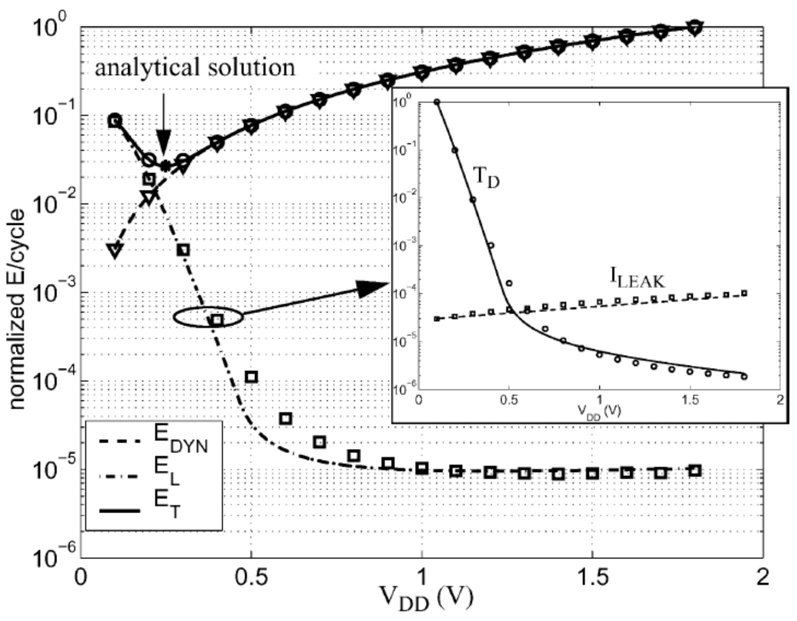


Figure 2: Active energy, leakage energy, and total energy of a FIR filter when sweeping its supply voltage V_{DD} [5]

The detailed architecture illustrated above is shown in figure 1. There are multiple Basic Logic Elements (BLE) in each CLB. Each BLE includes one LUT, one FF, and one MUX. The output of the LUT is connected to the input of the FF and one of the inputs of the MUX, while the MUX is used to determine whether the BLE is implementing combinational or sequential logic by selecting between the outputs of the LUT and the FF. The BLE outputs are connected to the CLB outputs through buffers. The CLB local interconnect plays a role to help communication between BLEs inside a CLB, by connecting the output of each BLE to the input of other BLEs and providing feed back loops to its own inputs. Also, there are large MUXes placed at the input of each BLE to select the BLE inputs from all the CLB inputs and BLE outputs. The FPGA global interconnect is used to make connections between CLBs and IOs. The multi-track interconnect includes Connection Boxes (CB) and Switch Boxes (SB), where CBs are used to connect CLB inputs and outputs from/to the global interconnect, and SBs are used to route the signals from the original CLB to the destination CLBs. Both CBs and SBs are made of buffers and switches, while configuration bits are connected to these switches. CBs are located at each input/output of the CLBs. By turning on specific switches, the signal can be detected or directed to specific tracks of the global interconnect. This multi-track architecture provides high flexibility of FPGAs. SBs are located at the intersections of vertical tracks and horizontal tracks. By turning on the corresponding switches, signals inside the global interconnect can be directed to the destinations.

2.2 Sub-threshold Operation

While many applications today pay more attention to high performance, the ubiquitous computing applications such as wearable sensors require extremely low energy consumption to support long battery life or even energy harvesting technology. Since active energy is quadratically proportional to the supply voltage V_{DD} at nominal voltage, the method of reducing V_{DD} is widely used in low-power-oriented designs [6]. Sub-threshold operation is a technique to reduce V_{DD} to lower than the threshold voltage of transistors. In sub-threshold, current is exponentially proportional to V_{DD} as shown in (1).

$$I_{subV_T} = I_0 e^{\frac{(V_{GS}-V_T)}{nV_{th}}} \tag{1}$$

Reducing V_{DD} down to sub-threshold makes it possible to decrease energy consumption to extremely low level. Although active energy reduces with V_{DD}, the leakage energy increases rapidly in sub-threshold due to the exponentially increased delay. At nominal voltage, leakage energy can be ignored, but becomes the major energy contributor in sub-threshold. Figure 2 shows the energy of a FIR filter when sweeping the supply voltage V_{DD} [5]. As shown, the lowest-energy-point is reached around 0.25V, where leakage energy and active energy are equal.

Obviously, near/sub-threshold operation is a very effective technique used to reduce energy. Although many works are done in this area, they mainly focus on ASIC and processor designs. FPGAs can also benefit from this technique when designed for ultra-low-power applications.

2.3 Existing Low-Power FPGAs

Researchers and engineers from both commercial and academic fields spend time on developing low-power FPGAs. As the largest FPGA vendors, however, Xilinx and Altera's expertise is more on the high-performance end, and currently have no public products for ultra-low-power applications. Their FPGAs consumes power in the range of 10s of Watts, which is too high for IoT applications. There are also commercial FPGAs designed towards low-power. The Lattice iCE40 [7] has a standby power of 71uW, while the Microsemi IGLOO [8] has a standby power as low as 10uW by using Flash-based configuration bits. However, the power of these products keeps in the range of uW only in sleep mode. When implementing applications, their power raises to several mW. In academic field, [9] presents a 4-CLB FPGA, which consumes as little as 40uW total power. However, this design is too small to implement meaningful low-power applications. In [10], a full FPGA fabric with over 2000 logic blocks, block RAMs, and DSPs is introduced. By using a hierarchical interconnect, this design reduces power by 4x-5x comparing to Xilinx Virtex products, while keeping competitive speed. However, since not targeting at ultra-low-power applications, this design consumes power in the range of several tens of mW, which is too high for IoT.

3 Energy Efficient FPGA Interconnect

3.1 Motivation

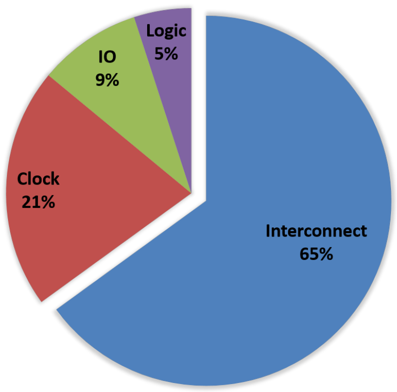


Figure 3: FPGA Core Fabric Energy Breakdown

Interconnect traditionally contributes a big portion of delay and power in circuits. As the size of logic blocks shrinks with technology node scaling, the interconnect cannot scale at the same rate due to physical limitations, such like electromigration and cross-talk. Thus, the interconnect becomes consuming even more delay and power in modern sub-micron electronic systems. In low-power system development, a research focus on reducing interconnect power is crucial.

Unlike interconnect on SoCs and CPUs, FPGA interconnect consists not only wires and buffers, but also tens even hundreds of switches in series along paths with similar amounts of branches. This leads to huge fanout of the drivers and buffers in the interconnect, as well as the capacitance load. In low-power FPGAs, as the supply voltage is low, the capacitance load on the interconnect becomes even more dominant. As shown in figure 3, the specialty of FPGA interconnect makes it contribute 65% or more power of FPGA core fabric, which consists of only CLBs and interconnect [1] [2] [3].

Modern mainstream FPGAs integrated processors, transceivers/receivers, and other high-power hard IPs on chip to support more high-performance applications, such as machine learning, 5G network, data-centers, etc. As a result, the core fabric itself no longer the major power consumer of modern FPGAs on the market, and the interconnect power becomes not as important as before. However, in low-power FPGAs that have no much power budget for hard IPs, reducing interconnect power is still the most effective way to improve the overall FPGA power efficiency.

3.2 Prior Art

Researchers reduce FPGA interconnect power in three ways: 1) technology scaling 2) circuit and architecture optimization 3) place and route (P&R) algorithm improvement

All computing platforms benefit from technology node scaling in terms of speed, power, and area. As the dimension of devices shrink to sub-14nm in 2017, the FPGA core fabric power is reduced by hundred even thousands time compared to ten years ago. FinFet and SOI devices also enables ultra-low-power applications by reducing leakage power effectively. However, the emerging low-power applications also bring more and more strict power budget on the hardware. For example, personal health care applications need ultra-long battery-life of the sensors to enable circuit-on-skin. In [11], a self-powered body sensor node SoC is proposed with a full chip power as low as 6.45uW. Other low-power applications have similar power budget on the hardware. FPGAs usually consumes 14x more power compared to ASICs due the design for flexibility [4]. Keeping FPGA power low enough to meet the requirements of health care applications is even harder. In this scenario, simply switching to newer technology node cannot solve the problem. As technology scaling is reaching the physical limit, circuit/architecture/algorithm level low-power techniques are needed to apply to FPGA interconnect at the same time.

Typical circuit and architecture level power reduction techniques include 1) using multi/high-V_T devices 2) using reversed body bias 3) reducing supply voltage 4) inserting power-gating headers. All of these techniques have been applied to FPGA interconnect in exsiting work. Researchers in [12] and [13] exploited multi-V_T scheme, which allowed mixed usage of low and high threshold transistors in routing switches in order to reduce leakage current. In [12], researchers reduced FPGA leakage power by 1.7x - 2.5x by using reversed body bias, a technique of reducing transistor body voltage to increase threshold voltage. In [14], a new FPGA routing switch design that is programmable to operate in three different modes is introduced. In low-power mode, the supply voltage is lowered to just maintain the minimum function requirement, and the leakage power is reduced by up to 52% and active power is reduced by up to 31% comparing to in high-speed mode. Similarly, in [15] and [16], researchers applied a multi-V_{DD} scheme in FPGA interconnect and saved up to 61% of power. However, although these works reduced interconnect power effectively, they are at a high cost of speed degradation. As a result, the power efficiency is not improved.

Another research direction of FPGA interconnect power reduction is P&R algorithms. A smart algorithm can pack logic blocks in a minimum number of CLBs and minimize the total length of global interconnect. The power of FPGAs with shorter interconnect routing wires is much more efficient than FPGAs with long wires. In addition, since better P&R algorithms use less routing resources to implement applications, FPGAs could be made smaller to achieve lower power while still meet the speed requirement. There are many works have been done in this area. In [17], a "path finder" algorithm reduces routing delay by 11x. In [18], the channel width of FPGA interconnect is reduced by 22%. Algorithm level improvement is beyond the scope of this dissertation, but it is important to point out its importance in power reduction.

3.3 Low-Swing Interconnect

3.3.1 Overview

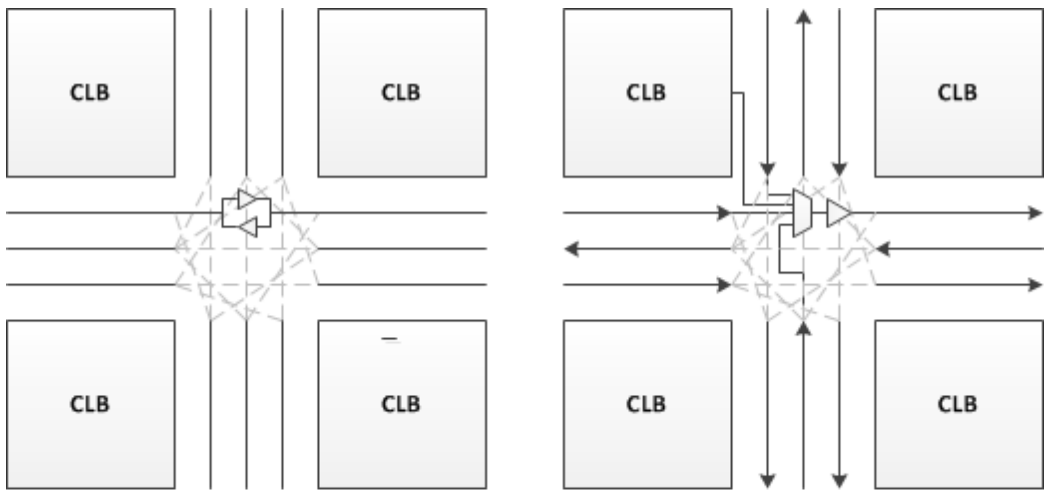


Figure 4: (a) Bi-directional switch box (b) uni-directional switch box

While power is the energy consumption of circuits in unit time, energy itself is a more important metric to evaluate hardware used in battery-life-oriented applications. As discussed in chapter 2, the total energy consists of dynamic energy and leakage energy. The dynamic energy of FPGA interconnect is calculated by

$$Dynamic \quad Energy = C * V_{DD} * VDD_{swing} \tag{2}$$

where C denotes the total lumped capacitance of wires/switches/buffers, V_{DD} is the main supply voltage on interconnect drivers/buffers, and VDD_{swing} is the swing of the signal transmitting through the interconnect channel. Typically, VDD_{swing} has a value equal to V_{DD}. However, in some special designed circuits, such as the low-swing interconnect that will be discussed in this chapter, their values could be different.

To minimize dynamic energy, all of the three terms in the equation need to be minimized as long as the FPGA speed is in the acceptable range given by low-power applications. Reducing supply voltage V_{DD} to near/sub-threshold region is the most effective method to reduce dynamic energy because of the near quadratic correlation. However, as indicated in figure 2, the leakage energy increases with V_{DD} decrease, and finally becomes the major portion of system energy. Thus, the value of V_{DD} needs to be carefully selected to achieve overall minimum FPGA interconnect energy.

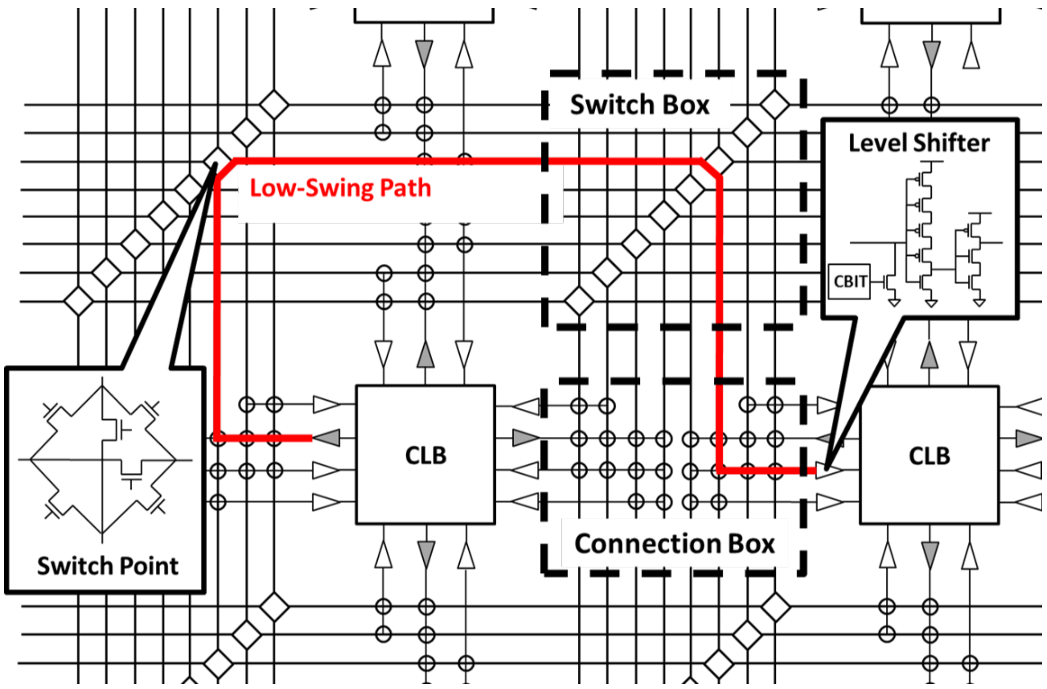


Figure 5: Basic structure of low-swing interconnect

The routing fabric in FPGAs is defined as the electrical connectivity hardware between CLBs. It is comprised of connection boxes that connect CLBs to the routing channel, switch boxes that form the connectivity of routing paths, and wire segments. The traditional bi-directional and uni-directional SBs are shown in figure 4 (a) and (b) respectively. Each bi-directional routing switch is comprised of 2 tri-state buffers, while each unidirectional switch is comprised of an N-input multiplexer followed by a buffer, where N represents the number of tracks that can connect to the track that this switch drives [19] [20] [21]. As shown in the figure, no matter uni-directional or bi-directional, traditional FPGA interconnect uses multiplexers and buffers to implement routing switches to achieve high speed, but it suffers from high energy cost. Reducing supply voltage for conventional interconnect to near/sub-threshold helps solving this problem. However, since driver and buffer current decreases exponentially in sub-threshold, delay is increased exponentially as well. Although speed is not the major metric for low-power applications, the FPGAs still need to be fast enough to accomplish tasks. Simply upsizing drivers and buffers does not help, since speed depends linearly on device size but exponentially on V_{DD} in sub-threshold. The low-swing interconnect replaces the multiplexers and buffers structure with pass-gates [22] [23]. Its basic structure is shown figure 5. This topology reduces "C" by removing buffers. Also, the "VDD_{swing}" is also reduced due to the natural voltage drop across pass-gates. Furthermore, the novel level shifter (developed by Dr. Joseph Ryan) that receives the reduced swing signals at the input to the CLBs reduces delay by detecting the signal earlier in its transition than traditional receivers or level shifters [22] [23].

3.3.2 Delay and Energy Trade-Off

Although the design goal of this FPGA is to meet the requirements of low-power applications, the speed and energy requirements may vary with different applications. The simplest way to obtain this flexibility is to adjust the supply voltage of the FPGA. This means increasing V_{DD} when higher speed is needed, or decreasing V_{DD} when a lower power budget is set. However, since speed and energy are very sensitive (quadratic relation) to V_{DD} change in near/sub-threshold, adjusting V_{DD} requires highly accurate voltage control, which is not realistic on a small low-power chip. An alternative solution is instead of adjusting the main V_{DD} of the FPGA, just change the gate voltage of the routing switches in the interconnect. Since the routing switches in low-swing interconnect are pass-gates, changing their gate voltage can adjust the delay and signal swing of the interconnect with the overhead of a separate voltage rail and very small gate leakage energy penalty. Since changing gate voltage does not affect V_{DD}, but only "VDD_{swing}", the requirement on voltage control accuracy is dramatically reduced. This idea will be further discussed later in this chapter.

3.3.3 Applicable Conditions of the Low-Swing Interconnect

Although the low-swing design reduces interconnect energy effectively, the benefit is achieved with a price on reduced robustness. Since the signal swing is too small, the PVT variations and voltage/ground noises can easily make a logic high signal to logic low, especially in near/sub-threshold where V_{DD} is small. Thus, the low-swing interconnect is suitable for high performance systems with a need of power saving or ultra-low-power systems with signal degradation solutions, while will be discussed later in this chapter.

3.4 Interconnect Modeling

3.4.1 Overview

To implement meaningful applications, a low-power FPGA should include a large number of CLBs and a wide interconnect channel built with over a hundred of tracks. Simulating such a large system for only one cycle can take from tens of hours to several days. To make things worse, sweeping circuit parameters towards minimum energy and lower variation requires hundred/thousand times of runs in the design phase. All the subsequent work such as chip tapeouts need to pulse for months to wait the simulation results to be done. This is unreasonable and unacceptable. Thus, a circuit model of the low-swing interconnect is needed for design purpose. This model should be accurate enough to represent the complicated environment of the interconnect, and also should be simple enough to finish simulations in short time.

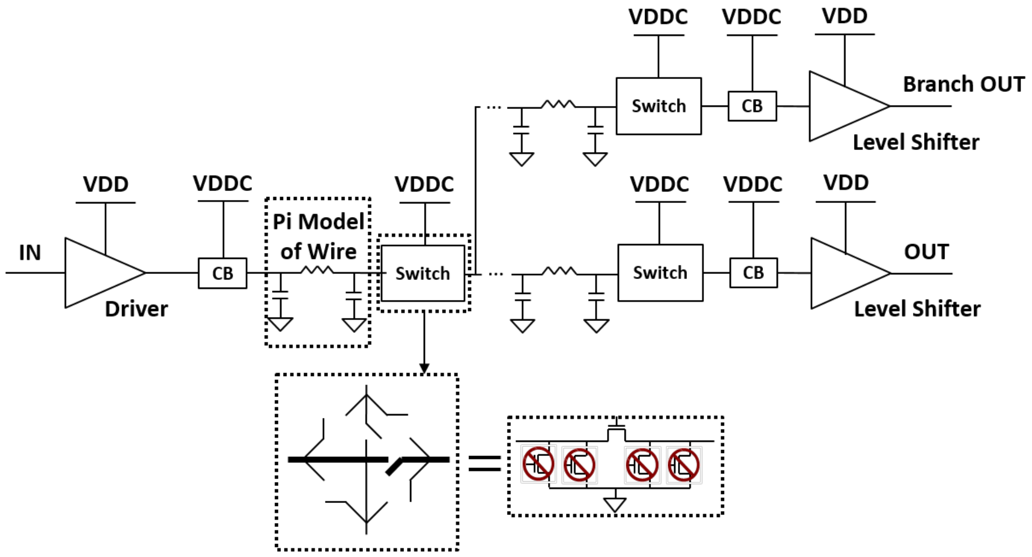
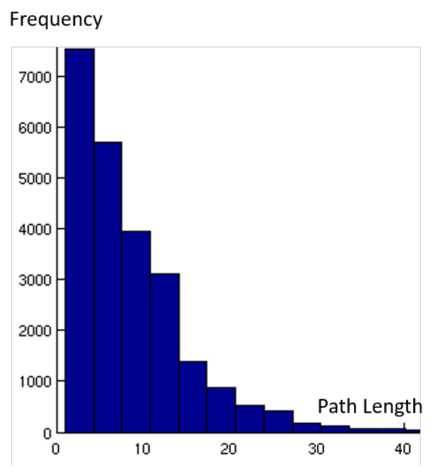


Figure 6: Diagram of the global interconnect path model

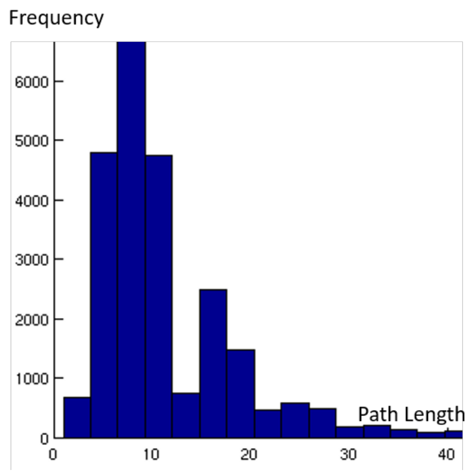
The architecture of low-swing interconnect has been shown in figure 5. As mentioned earlier in this chapter, a FPGA interconnect path is defined as the circuit starting from the driver at an output of a CLB, passing CBs and switches, then ending at a level shifter of the destination CLB. The function of level shifter is to regenerate the signal to full swing to avoid short-circuit current in the destination CLB as well as improving noise margin. Figure 6 shows the diagram of the abstraction of figure 5, which is the interconnect model used in this work. Each wire segment is modeled as a Pi structure to represent the highly capacitive long wires. Each routing switch is modeled as one turned-on switch and four turned-off switches connected to ground, representing the signal path and the leakage paths respectively. Each CB is modeled as a multiplexer. A separate V_{DDC} voltage is applied to routing switches and

CBs by high-V_T config bits to provide flexibility in delay and energy. Level shifters can be inserted either between two switches when regeneration is needed or at the end of a path. Branches are added based on the path distribution analysis. To optimize the circuit, parameters including the value of V_{DD}, V_{DDC}, the topology and size of CBs and switches, and the number of inserted level shifters will be varied and the corresponding influence on energy efficiency will be evaluated and discussed in the remaining part of this chapter.

3.4.2 Path Length and Branch Distribution



(a)



(b)

Figure 7: Path length distribution (a) segment length = 1 (b) segment length = 4

The length of an interconnect path in this dissertation is defined as the number of routing switch on the path from the origin CLB to the destination CLB. The length of paths varies from 1 to over 80 and is not equally distributed. To optimize the interconnect circuit, we ideally want to look at all interconnect paths when implementing all target applications. However, doing so is too time consuming and does not necessary. We can archive a near-optimal design by looking into a carefully selected collection of interconnect paths that can represent the majority scenarios. Since the optimal circuit of the interconnect highly depends on the length of the paths, we firstly need to set a range of path length to aim at.

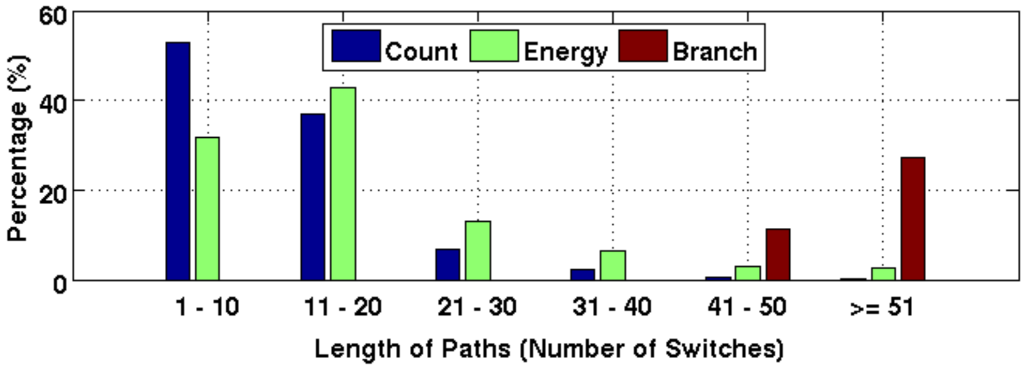
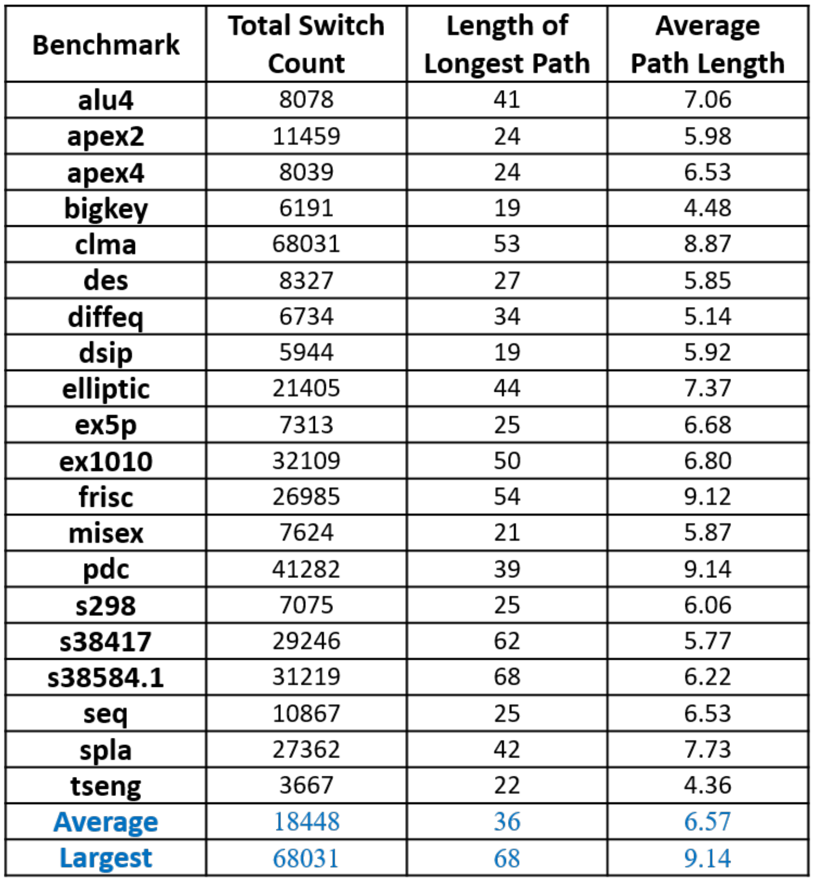


Figure 8: Abstracted Path and branch distribution

The path length distribution depends on the FPGA architecture, the P&R tool, and the benchmarks selected to map to the FPGA. In this chapter, an open source FPGA P&R tool called VTR [24] is used to run a general-accepted Microelectronics Center of North Carolina (MCNC) benchmark suite [25] to investigate the FPGA path distribution. A basic FPGA architecture, which has a logic cluster size of 8 and a look-up-table input number of 4, is selected as the target fabric to map the benchmarks. The path length distribution is shown in Fig. 7. For some express interconnect wires, the routing switch does not appear at every horizontal and vertical track intersections. The number of intersections they skipped affects the path length. To obtain a comprehensive path length distribution that considers all the cases, Fig. 7 shows two histograms present different segment lengths accordingly, which represent the number of CLBs each wire segment spans without routing switches in the middle. As a result, the histograms indicate no matter which architecture is used, the majority of paths are less than 40 routing switches. However, while most of the paths are less than 5 when segment length equal to 1, the length of most paths is around 10 when segment length is 4. This is because longer segment is harder to route to adjacent logic blocks when the applications are small and compact, especially for low-power applications.

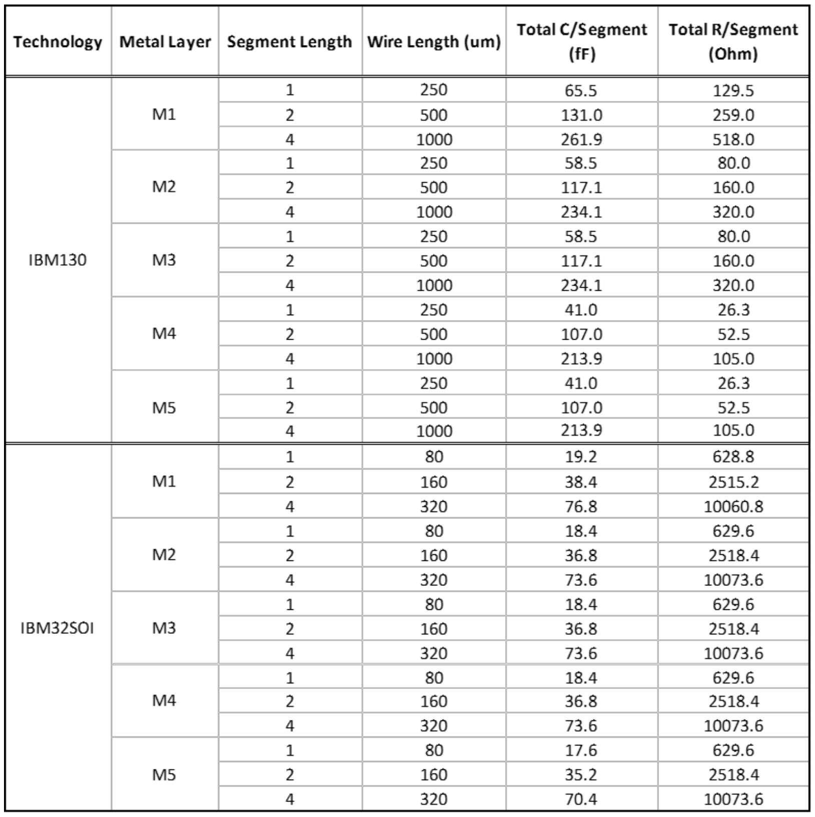


Benchmark	Total Switch Count	Length of Longest Path	Average Path Length
alu4	8078	41	7.06
apex2	11459	24	5.98
apex4	8039	24	6.53
bigkey	6191	19	4.48
clma	68031	53	8.87
des	8327	27	5.85
diffeq	6734	34	5.14
dsip	5944	19	5.92
elliptic	21405	44	7.37
ex5p	7313	25	6.68
ex1010	32109	50	6.80
frisc	26985	54	9.12
misex	7624	21	5.87
pdc	41282	39	9.14
s298	7075	25	6.06
s38417	29246	62	5.77
s38584.1	31219	68	6.22
seq	10867	25	6.53
spla	27362	42	7.73
tseng	3667	22	4.36
Average	18448	36	6.57
Largest	68031	68	9.14

Figure 9: Details of path distribution of MCNC benchmarks

The path length distribution cannot represent the energy distribution. Although most of the paths are short (less than 10), the longer paths consume much higher energy. As energy linearly depends on the total capacitance in the interconnect, the branch distribution is also worth to look at. In Fig. 8, the path length, energy and branch distributions are all shown in one plot for comparison. In the plot, paths are divided into 6 categories based on path length. The blue and green bars represent the path count distribution and the energy distribution. The red bar represents the average percentage of switches from the path that fall on branches rather than the main path. As indicated in Fig. 8, paths shorter than length 40 take about 98% of the total path count and consume about 94% of the total global interconnect energy. Although branches are very common in the FPGA interconnect network, there are few branches on paths shorter than 40. Such analysis indicates that in order to increase energy efficiency of FPGA interconnect, circuit level optimization should mainly focus on paths shorter than 40 without branches. However, the results highly depend on how the P&R tool operates. For that reason, paths less than 68 are all considered in this chapter. The details of path distribution of the largest 20 MCNC benchmarks are shown in Fig. 9.

3.4.3 Detailed Circuit Modeling



Technology	Metal Layer	Segment Length	Wire Length (um)	Total C/Segment (fF)	Total R/Segment (Ohm)
IBM130	M1	1	250	65.5	129.5
		2	500	131.0	259.0
		4	1000	261.9	518.0
	M2	1	250	58.5	80.0
		2	500	117.1	160.0
		4	1000	234.1	320.0
	M3	1	250	58.5	80.0
		2	500	117.1	160.0
		4	1000	234.1	320.0
	M4	1	250	41.0	26.3
		2	500	107.0	52.5
		4	1000	213.9	105.0
	M5	1	250	41.0	26.3
		2	500	107.0	52.5
		4	1000	213.9	105.0
IBM32SOI	M1	1	80	19.2	628.8
		2	160	38.4	2515.2
		4	320	76.8	10060.8
	M2	1	80	18.4	629.6
		2	160	36.8	2518.4
		4	320	73.6	10073.6
	M3	1	80	18.4	629.6
		2	160	36.8	2518.4
		4	320	73.6	10073.6
	M4	1	80	18.4	629.6
		2	160	36.8	2518.4
		4	320	73.6	10073.6
	M5	1	80	17.6	629.6
		2	160	35.2	2518.4
		4	320	70.4	10073.6

Figure 10: Table of the Equivalent Capacitance and Resistance of a Wire Segment

The interconnect model is built from a combination of five element sub-circuits: driver, level shifter, wire segment, routing switch, connection box.

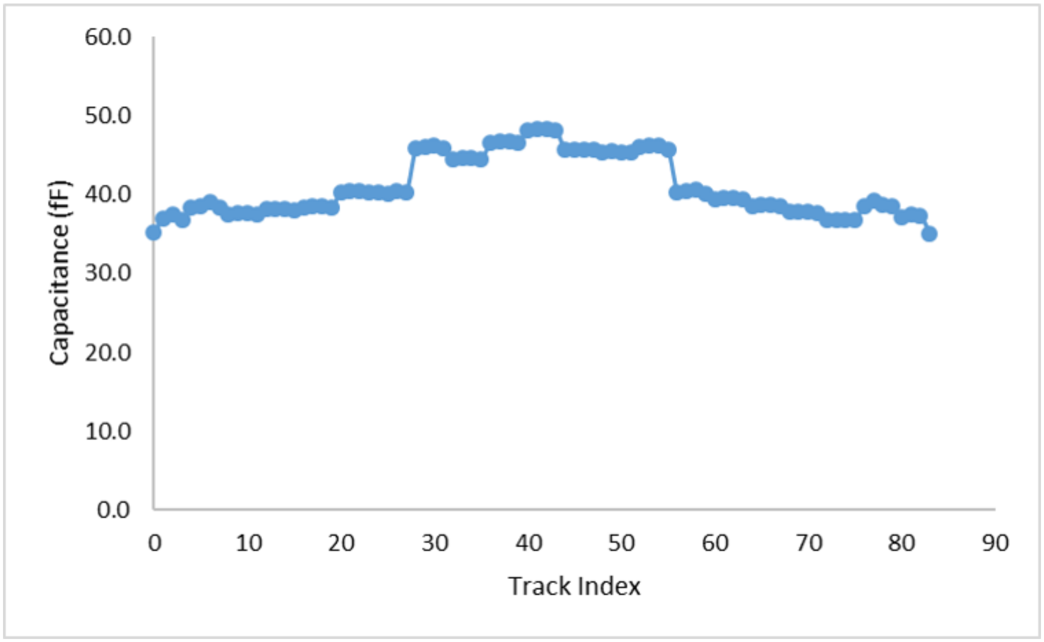


Figure 11: Total Capacitance of Wire Segments in a 84-Track Channel

Both the driver and level shifter are modeled as two cascade inverters. However, both inverters in the driver are standard structure with one PMOS and NMOS that have same driving strength, while the level shifter is built of one strong-NMOS interver followed by a strong-PMOS inverter taking advantage of stack effect. The purpose of this design is to detect small signal swings without speed overhead [22] [23]. In this dissertation, the same structure of this level shifter is used with modifications on sizing, and its schematic can be found in Fig. 5. Since the driver locates at the very beginning of interconnect paths, its input signal is always at full swing, so using a basic inverter-based design is fine and can save area. The size optimization of the driver will be discussed in the following contents of this chapter.

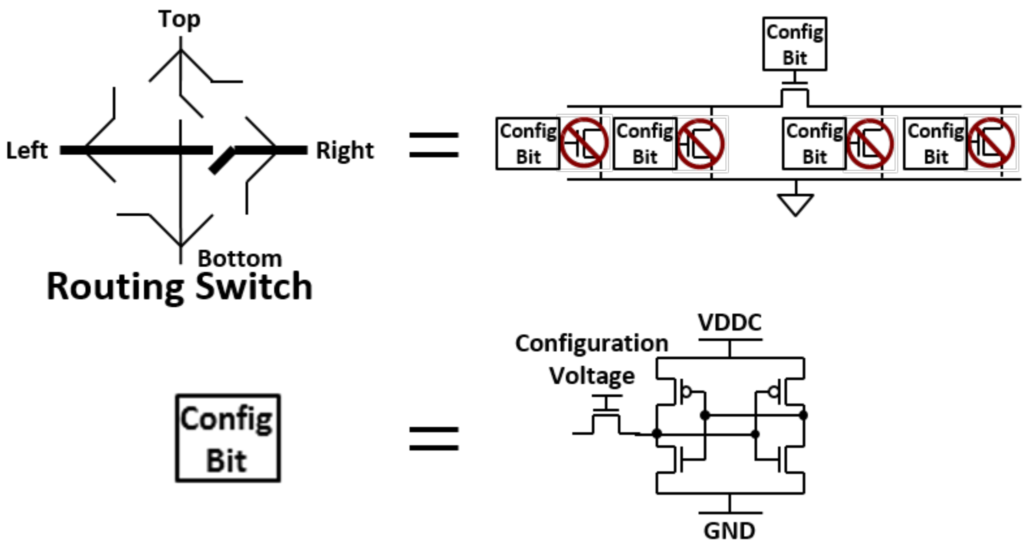


Figure 12: Routing Switch Modeling

To model the wire segments, either Pi model or T model can be used. Since both models have equivalent total capacitance/resistance values and no additional circuit parts between adjacent routing switches, either model works for this research. In this dissertation, the Pi model is selected. Three factors affect the value of R and C in the wire model: 1) the length of each wire segment 2) the metal layer used for routing in physical implementation 3) the coupling capacitance between wires in the same channel. As will be explained in 3.7, the length of each wire segment is 250um (segment length = 1)/500um (segment length =2)/1000um (segment length =4) in 130nm CMOS. Based on this, the equivalent capacitance and resistance values of wires implemented by each metal layer in both 130nm CMOS and 32nm SOI technologies are estimated and shown in Fig. 10. The RC values are calculated by reading from the PDK user manuals, then verified in the Cadence Virtuoso parasitic extraction environment. Since the interconnect channel includes a bundle of tens of tracks in parallel close to each other, the coupling capacitors between adjacent tracks can't be ignored. Fig. 11 shows the total capacitance of each track in an interconnect channel.

The tracks in the middle have much higher capacitance compared to those at the edge of the channel.

Both the routing switch and connection box are built with pure pass-gates and configuration bit-cells. While the connection box structure is one of the knobs in this research and will be discussed later, the circuit model of the routing switch is shown in Fig. 12. Each routing switch includes six pass-gates to provide potential connectivity of all four directions: Left-to-Right, Left-to-Top, Left-to-Bottom, Right-to-Top, Right-to-Bottom, and Top-to-Bottom. In the case of no branch in a path, one pass-gate will be turned on, four pass-gates will be turned off creating leakage paths, and the last pass-gate is not related to the example path. For example, if a signal is transmitted from left to right in Fig. 12, the Top-to-Bottom pass-gate is unrelated. In FPGA interconnect, if a switch is turned off, a floating net might be created and causes short-circuit current. To avoid this situation, all potential floating nets will be either pull up to V_{DD} or pull down to ground. In this research, ground is selected as the default. For the reasons described above, the routing switch model is a pass-gate with 4 parallel turned-off pass-gates to ground. The voltage to the pass-gates are not directly given by supplies, but through a configuration bit made of a 5T SRAM bit-cell. Since these memory cells are not in an array and are configured once without value changes thereafter, 6T or more complicated structures are unnecessary and minimum size high-VT transistors are used to minimize leakage.

3.5 Circuit and Supply Voltage Optimization

3.5.1 Overview

The goal of this research is to improve the energy efficiency of the interconnect while maintaining the robustness of it at a certain level. This goal results in many circuit design trade-offs. For example, the requirement of high energy efficiency indicated both low energy/operation and adequate speed. To reduce energy, we can 1) reduce transistor size of routing switches/CBs/drivers 2) decrease V_{DD}/V_{DDC} 3) using transmission-gates instead of pass-gates to implement routing switches. However, all of these methods decrease system speed, which is already very low in near/subthreshold. There is a sweet point to balance speed and energy for each design knob. The energy-delay-product (EDP) and the energy-delay (ED) curve of the interconnect are widely used metrics to evaluate energy efficiency of circuits, and are also used in this research. Another trade-off example is about circuit robustness. While reducing V_{DDC} or using smaller size of pass-gates decreases energy, the signal swing is also decreased and worsen robustness. All of these trade-offs are explored and explained in this chapter.

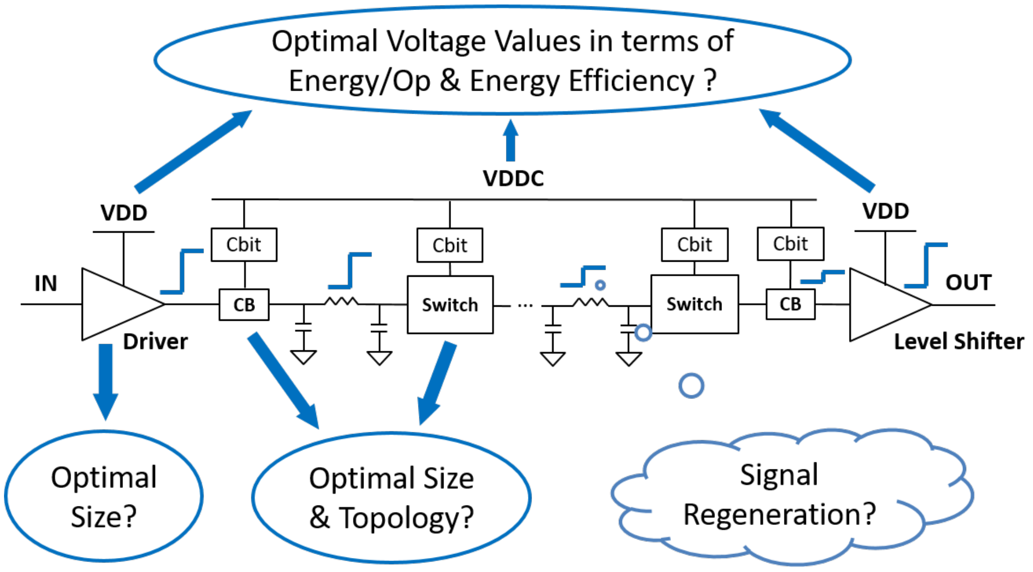


Figure 13: Knobs of Interconnect Optimization

3.5.2 Supply Voltage Optimization

Supply voltage V_{DD} is a dominant knob for EDP. There are three components contributing to EDP: delay, active energy, and leakage energy. V_{DD} affects all of the important parameters for energy efficient FPGAs. Path delay decreases exponentially in sub-threshold, while it only decreases quadratically in the above-threshold region. Energy is lower in the sub-threshold region and is dominated by leakage energy, while dynamic energy, which decreases quadratically with V_{DD}, dominates total energy for above-threshold operation [26]. In this research, V_{DD} is swept from 0.3V to 0.6V for paths with length of 5, 10, 20, and 40. V_{DDC} is swept from 0 to 0.8V above V_{DD}. The minimum EDP is obtained at $V_{DD} = 0.5$V. Increasing V_{DD} from 0.5V to higher cannot further decrease EDP, but increases energy. On the other hand, reducing V_{DD} to 0.4V is very beneficial when energy is more important than energy efficiency, because much smaller energy can be achieved with small EDP overhead. However, reducing V_{DD} to 0.3V results in rapidly increased EDP but relatively smaller energy reduction. This conclusion applies to path with a length of 5, 10, 20, and 40 indicating it is a general conclusion for the majority of paths involved in MCNC benchmark routing.

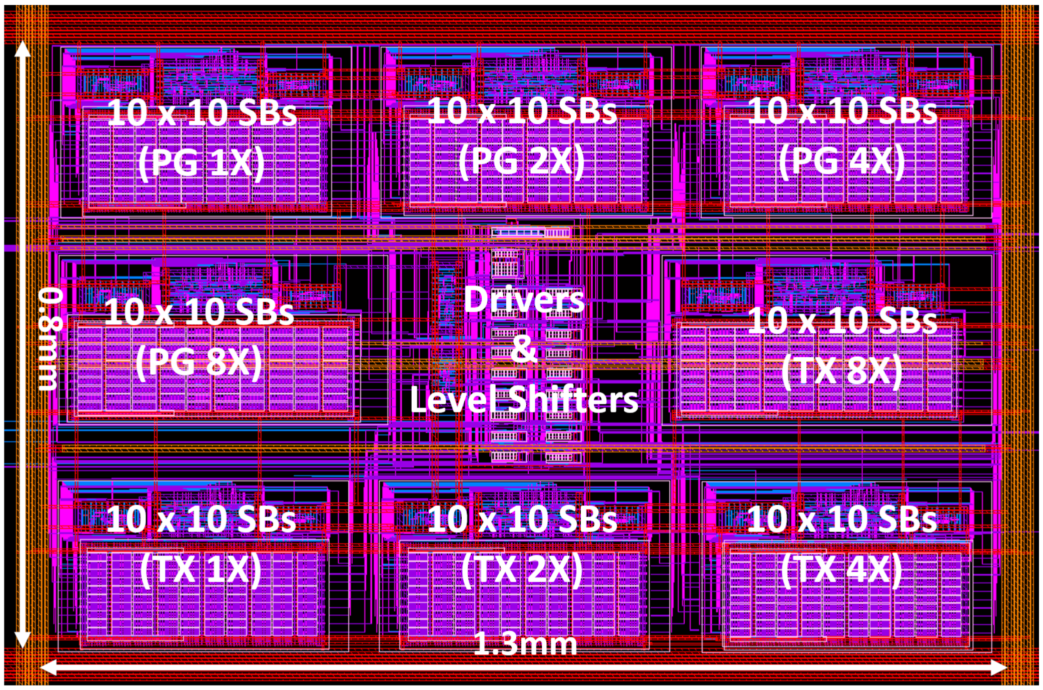
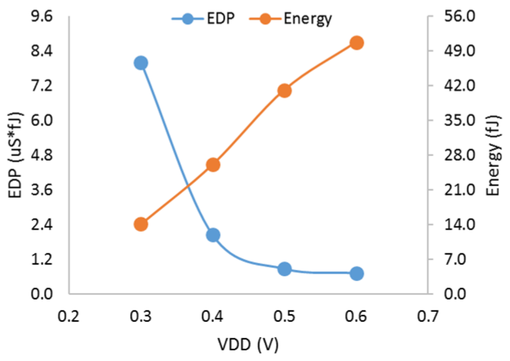
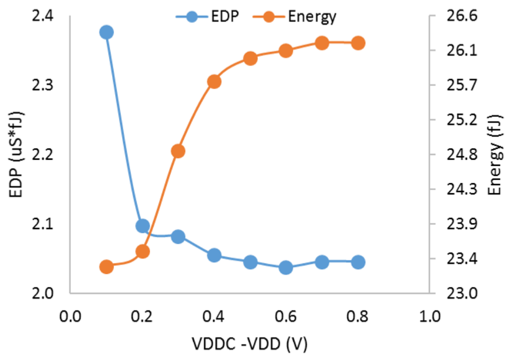


Figure 14: The layout photo of the characterization chip

Besides V_{DD}, energy and delay also depend on V_{DDC}. As explained earlier, the dynamic energy of a path equals to C*V_{DD}*VDD_{swing}. For smaller V_{DDC}, the equivalent resistance of switches is large due to sub-threshold operation. Larger resistance leads to increased voltage drop and decreased signal swing. Consequently, dynamic energy and speed are both low. Applying a higher V_{DDC}, on the other hand, results in higher dynamic energy but substantially reduced delay. As a result, the delay decreases sharply as V_{DDC} increases in the range of $V_{DD} \leq V_{DDC} \leq V_{DD} + 0.2V$. Keeping increasing V_{DDC} to above V_{DD} + 0.2V can no longer reduce delay as significantly as before. On the other hand, energy increases slowly as V_{DDC} increases when $V_{DD} \leq V_{DDC} \leq V_{DD} + 0.2V$, while it experiences a much faster increase when $V_{DDC} \geq V_{DD} + 0.2V$. Similar to delay, the EDP decreases sharply at low V_{DDC} and slowly at high V_{DDC}. This transition point is discovered at 0.2V higher than V_{DD}. However, this value varies with path length. It can reach 0.3V above V_{DD} for paths longer than 40 and 0.1V for paths shorter than 10.



(a)



(b)

Figure 15: Measurement results of the energy and EDP of a path when sweeping (a) V_{DD} (b) and V_{DDC} in 130nm CMOS (Path Length = 10)

To verify the above conclusions, a chip is designed to implement eight low-swing FPGA interconnect meshes with different circuit topology (pass-gates/transmission-gates) and transistor sizes (1x, 2x, 4x, and 8x) of routing switches in 130nm CMOS.

Each mesh is a 10-by-10 routing switch structure interconnected by wire segments, which are intentionally inserted between switches to imitate the RC of long wires in real FPGA fabrics. The meshes are driven by a driver block on the die. The driver block comprises drivers with different sizes followed by connection boxes that can be configured to be turned on or off. The annotated layout of the test chip is shown in Fig. 14. Each mesh on the chip is configured to signal paths with all branches and leakage paths. The measured normalized energy and EDP when sweeping V_{DD} and V_{DDC} are shown on the left and right plot respectively in Fig. 15. The chip measurement results indicate conclusions the same as simulation. This chip will be used again in circuit optimization later in this chapter.

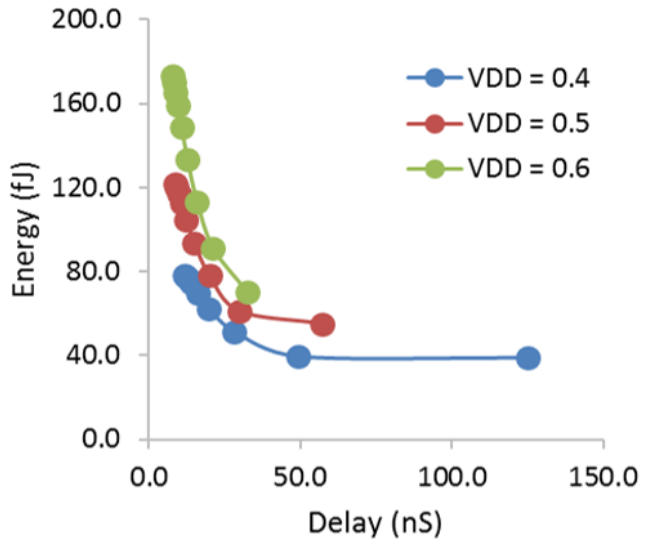


Figure 16: ED curves of interconnect paths at different V_{DD} in 32nm SOI (path length = 40)

In case the conclusions above vary from technology to technology, I also ran the same simulations in 32nm SOI, which is a more advanced technology in leakage reduction. Fig. 16 shows the ED curves of two paths (left: length 5; right: length 40) at different V_{DD} by sweeping V_{DDC} from the right end of each curve to its left end.

On the way of sweeping V_{DDC}, there is always a combination of delay and energy on the ED curve results in the lowest EDP. Higher V_{DD} always leads to higher energy, while usually leads to smaller delay. However, for paths longer than 40, path delay at different V_{DD} are almost same. Thus, using a lower V_{DD} for higher energy efficiency is recommended. In Fig. 17, the V_{DDC}-V_{DD} values lead to these minimum EDPs are also shown. The optimal V_{DDC} values decrease as the size of pass-gates increases for all paths. Furthermore, the optimal V_{DDC} values for longer paths are higher than shorter paths. To be more specific, when using 4x pass-gates as the routing switches, the optimal value of V_{DDC} is 0.1V higher than V_{DD} for paths with length of 5, and is 0.3 0.4V higher than V_{DD} for longer paths.

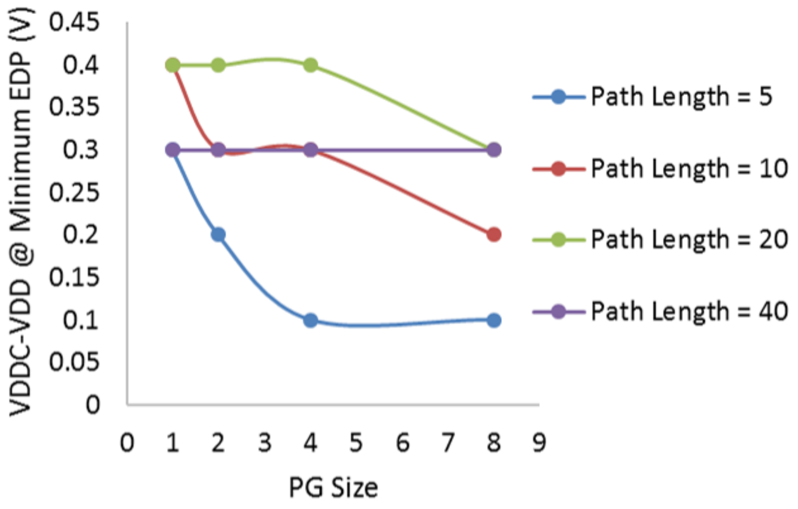
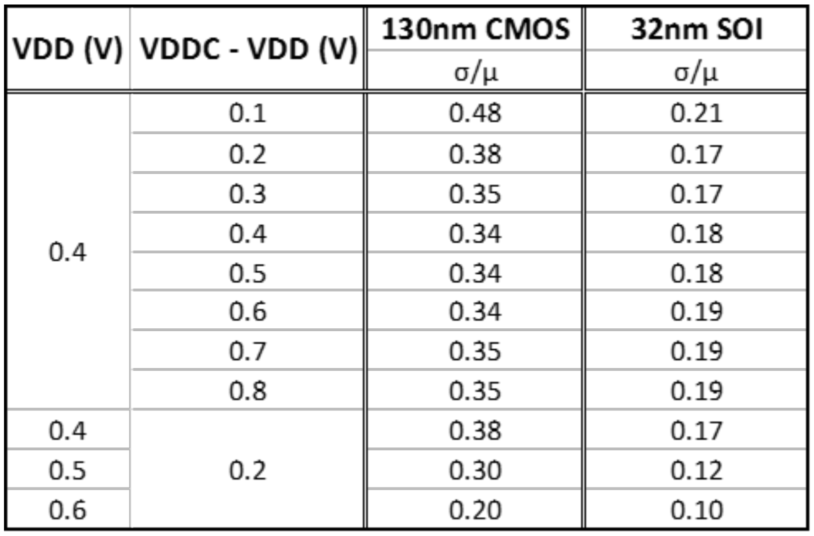


Figure 17: The optimal V_{DDC} value results in minimum EDP of interconnect paths with different lengths in 32nm SOI @ V_{DD}=0.4V

In near/sub-threshold operations, the process variation is significant. Fig. 18 shows the average delay and the standard deviation of a length 10 path at different voltages. The data was obtained by running Monte Carlo simulation in both 130nm CMOS and 32nm SOI technology nodes. The variation is estimated by σ/μ. As a result, the 32nm SOI is better in reducing variation in general. Variation in 32nm

SOI is only have of that in 130nm CMOS. For each technology node, increasing V_{DDC} effectively reduces variation when V_{DDC} is low. When 0.3V higher than V_{DD}, keep increasing V_{DDC} does not reducing variation further. On the other hand, increasing V_{DD} always reduces variation in near/sub-threshold.



VDD (V)	VDDC - VDD (V)	130nm CMOS	32nm SOI
		σ/μ	σ/μ
0.4	0.1	0.48	0.21
	0.2	0.38	0.17
	0.3	0.35	0.17
	0.4	0.34	0.18
	0.5	0.34	0.18
	0.6	0.34	0.19
	0.7	0.35	0.19
	0.8	0.35	0.19
0.4	0.2	0.38	0.17
0.5		0.30	0.12
0.6		0.20	0.10

Figure 18: The process variation of paths at different voltage

3.5.3 Circuit Optimization

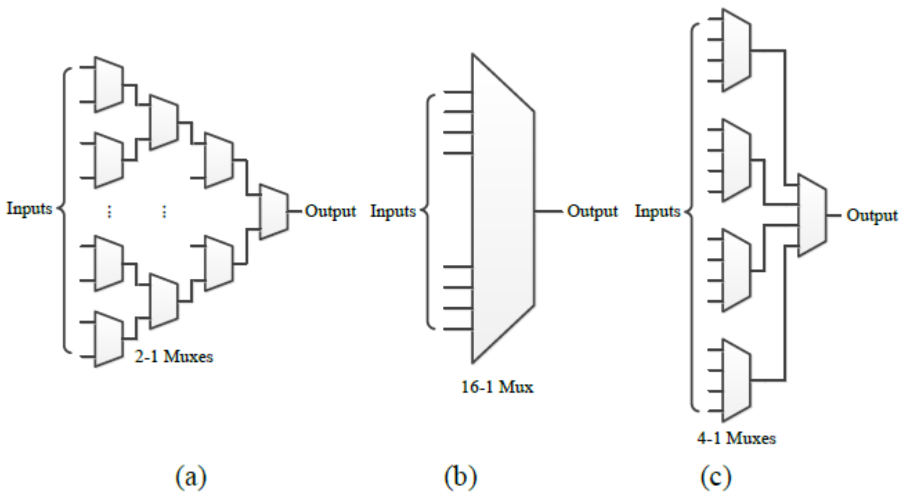
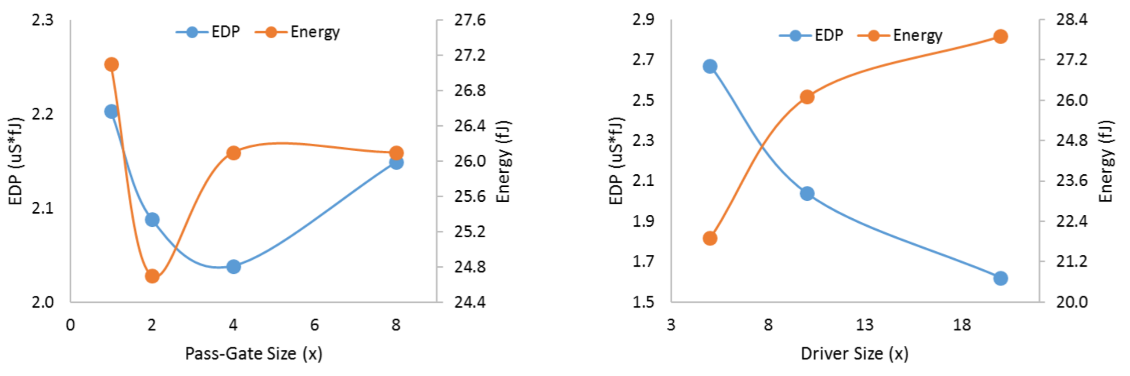


Figure 19: Schematic of different CB topologies: (a) full multiplexer (b) 1-stage multiplexer (c) 2-stage multiplexer

In this section, circuit optimization is performed on 1) connection box topology 2) routing switch implementation topology 3) routing switch transistor sizing 4) driver transistor sizing.

The connection boxes in FPGAs targeting at high performance operations are implemented by multiplexers with buffers to make connections between the routing fabric and the CLBs. In this research, the buffers are removed for low-power operations. According to SPICE simulation results, the connection boxes contribute 13.4% of total delay and 2.6% of total energy to a low-swing path. Only removing the buffers from the existing connection boxes without re-sizing and revisiting typologies results in un-minimized delay and energy. Fig. 19 shows three candidate connection box topologies for near/sub-threshold operations. The 1-stage design has the smallest delay because it adds only one transistor delay to the interconnect path. However, the capacitance load of this design is the sum of all drain/source capacitance of N transistors, where N represents the number of inputs of the multiplexer. In addition, the signal swing is also large. As a result, the 1-stage design suffers from high energy. In contrast, the full multiplexer benefits from both low active and leakage energy, but suffers from slow speed. Both of the two designs cannot guarantee the maximum energy efficiency in sub-threshold. The 2-stage multiplexer is a good alternative to balance energy and delay. The simulation result shows that the delay of the 2-stage multiplexer is 16% smaller than the full multiplexer, while the energy of the 2-stage multiplexer is 5% lower than the 1-stage design. In addition, the 2-stage design has the smallest variation among the 3 candidates. The overhead of using a 2- stage design is area (2.6x larger than a full multiplexer when N = 40). Considering energy efficiency and variation, the 2-stage design is optimal. This conclusion applies to paths with length of 5, 10, 20, and 40.

Since no buffers in the routing switches, drivers are the only consumer of the dynamic energy in low-swing interconnect. To achieve low energy, large drivers are not acceptable. However, simply reducing energy by decreasing driver size as much as possible is also not a good choice when delay is already large in the sub-threshold region. Under these circumstances, finding a driver size to balance energy and delay becomes a problem. As a result, increasing the size of drivers from 5x to 20x reduces delay by 55% with a 39% energy overhead. This result implies that a larger driver may result in a smaller EDP.



(a) (b)

Figure 20: Measured energy and EDP of paths with varying (a) switch size (b) driver size @ V_{DD}=0.4V in 130nm CMOS (Path Length = 10)

The transistor sizes of the routing switches/connection boxes also need to be optimized. Routing switches with a larger size introduce larger capacitance load into the interconnect fabric but result in larger signal swing and smaller delay. As mentioned in last section, a test chip is made to implement eight 10-by-10 routing switch meshes with different topology. As shown in Fig. 20 (a), the minimum EDP of the same path is obtained at a pass-gate size of 4x and is 15% lower than the EDP at a pass-gate size of 1x. In addition, the EDP of transmission-gates is always larger than pass-gates. In both simulation and measurement, the optimal switch size is sensitive to the RC value of wire segments. If ignoring wire RC, the optimal switch size is 1x. However, 2x switches are needed when wires are shorter than 45m, while 4x switches are needed for longer wires. Fig. 20 (b) shows that increasing the driver size from 5x to 10x reduces the EDP by 42% with a 2% energy overhead. Further increasing the driver size to 20x can decrease the EDP by 10% with a 10% energy overhead. Paths with length of 5, 10, 20 have the similar conclusions. The measurement results confirm the optimal choices of the topology and sizes of the circuit components, and the optimal value of supply voltages.

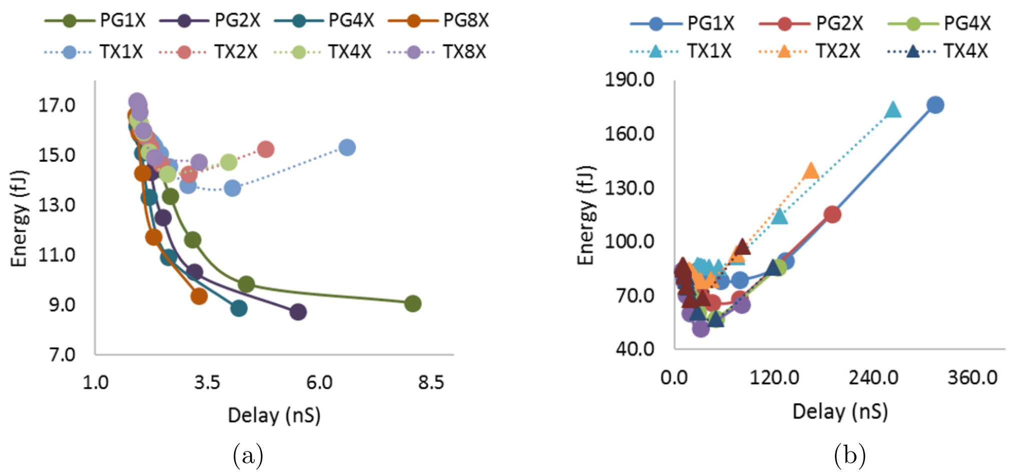


Figure 21: ED curves of interconnect paths with different switch box topologies and sizes @ V_{DD}=0.4V in 32nm SOI (a) path length = 5 (b) path length = 40

Similar to last section, I also run simulations with different circuit sizing and topology in 32nm SOI to explore how does the conclusions change with technology scaling. In Fig. 21, the ED curves of interconnect paths with varying switch box topology and size are shown. Each curve in the figure is obtained by sweeping V_{DDC}. From the bottom-right corner to the top-left corner on each curve, V_{DDC} is swept from low to high. As a result, for paths with length of 5, interconnect using pass-gates always consumes less energy than interconnect using transmission-gates. At low V_{DDC} (less than 0.3V higher than V_{DD}), transmission-gates provides faster speed than pass-gates. For paths with length of 40, the ED curves become closer to each other, but pass-gates benefit from 50% smaller area than transmission-gates. Furthermore, increasing pass-gate size from 1x to 4x dramatically reduces interconnect delay and energy. However, keeping increasing pass-gate size to 8x has no obvious effect on the ED curves.

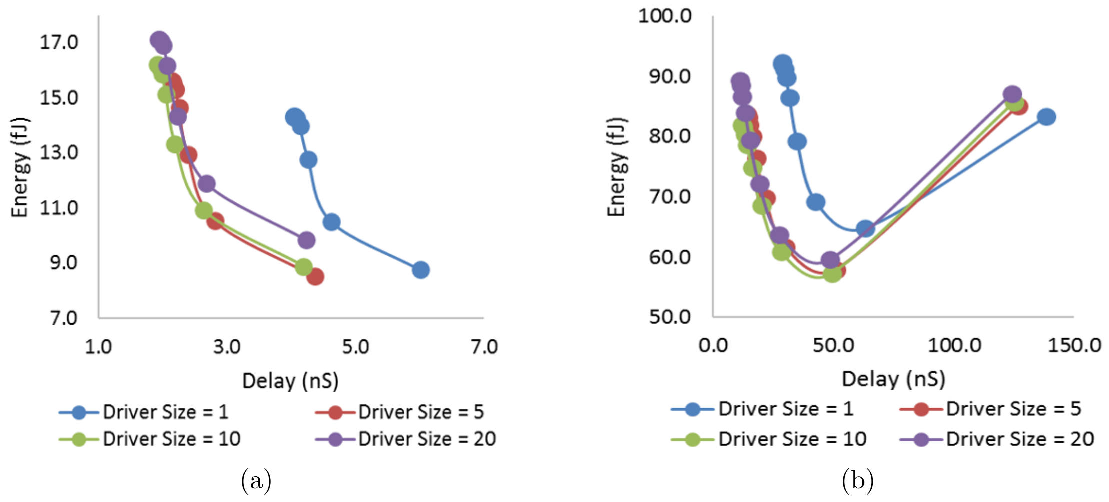


Figure 22: ED curves of interconnect paths with different driver sizes @ V_{DD}=0.4V in 32nm SOI (a) path length = 5 (b) path length = 40

VDD /VDDC(V)	Driver Size	Pass-Gate Size	130nm CMOS Path Length = 5 σ/μ	130nm CMOS Path Length = 20 σ/μ	32nm SOI Path Length = 5 σ/μ	32nm SOI Path Length = 20 σ/μ
0.4/0.6	1	1	0.33	0.29	0.22	0.16
		2	0.34	0.31	0.24	0.20
		4	0.34	0.31	0.25	0.24
		8	0.35	0.30	0.25	0.27
	5	1	0.33	0.33	0.17	0.12
		2	0.35	0.37	0.19	0.13
		4	0.35	0.38	0.21	0.15
		8	0.35	0.37	0.22	0.18
	10	1	0.33	0.31	0.17	0.12
		2	0.34	0.35	0.20	0.13
		4	0.34	0.36	0.22	0.14
		8	0.34	0.36	0.23	0.16
	20	1	0.32	0.29	0.18	0.12
		2	0.34	0.33	0.20	0.13
		4	0.34	0.34	0.22	0.14
		8	0.33	0.33	0.24	0.15

Figure 23: The process variation of paths with different circuit parameters

In Fig. 22, the driver size exploration in 32nm SOI is also shown. No matter how long the paths are, increasing driver size from 1x to 5x dramatically improved both delay and energy, and further increasing driver size doesn't help too much. For a path with length of 5, the EDP decreases when increasing the driver size from 1x to 10x. Keeping increasing the driver size to 20x has no much difference in EDP and the ED curve. For longer paths, driver size larger 5x results in similar ED curves.

The process variation is also considered in the circuit optimization of low-swing paths. In Fig. 23, the σ/μ of paths with different driver size and switch size are all shown. The data is obtained by Monte Carlo simulation in both 130nm CMOS and 32nm SOI. Similar to the conclusion indicates by Fig. 18, circuit in 32nm SOI is less affected by process variation. On the other hand, changing driver size and switch size does not affect the σ/μ of paths very much.

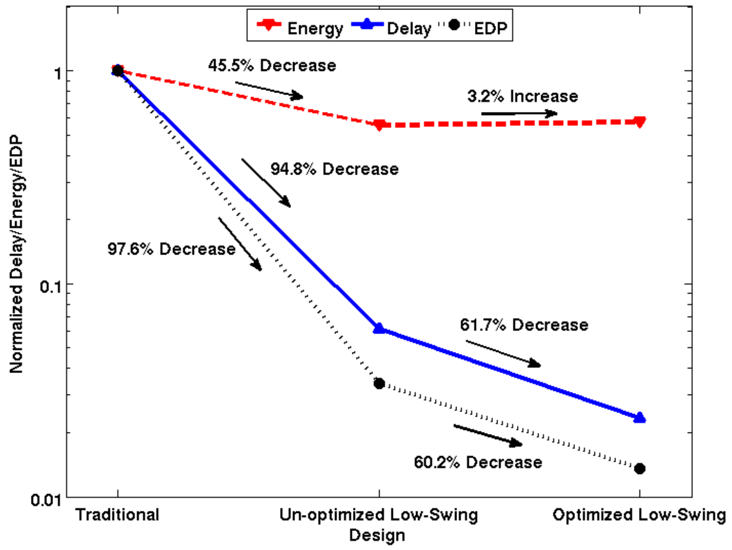
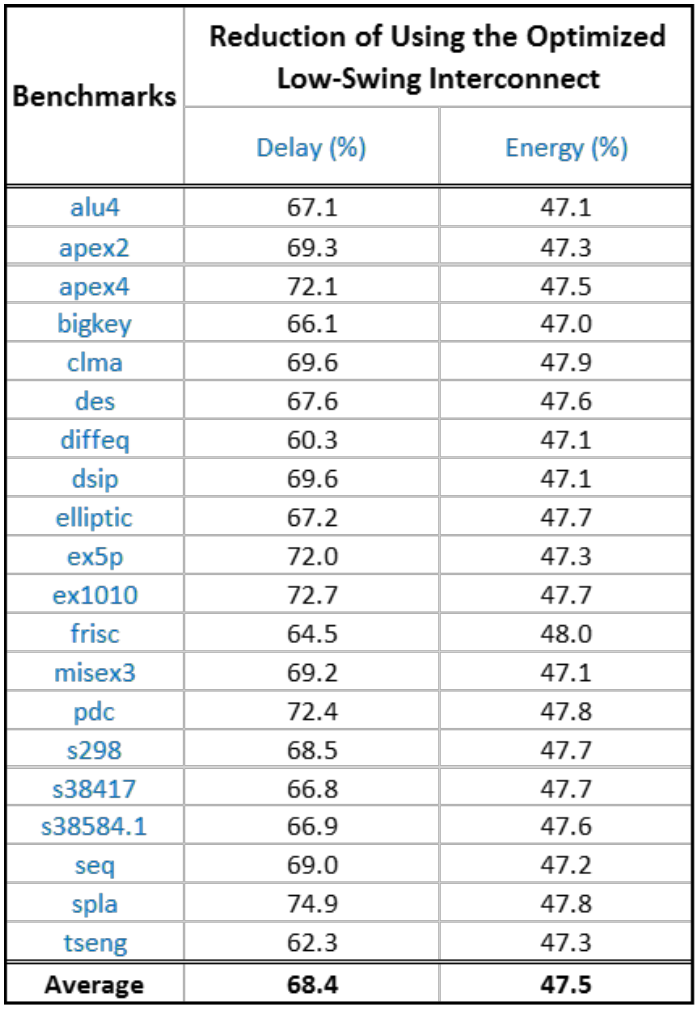


Figure 24: Comparison of the normalized delay, energy, and EDP of buffer-based, minimum-sized low-swing, and optimized low-swing interconnect @ V_{DD}=0.4V

The simulation results of the traditional buffer-based interconnect, un-optimized low-swing design, and optimized design are compared in Fig. 24. The optimized design has 61.7% smaller delay, 60.2% lower EDP, and 3.2% higher energy than the un-optimized design. The EDP is sharply reduced with very small energy overhead.

Comparing to the traditional uni-directional design, the optimized low-swing design has 97.7% smaller delay and 42.7% lower energy. The benefit of using the optimized interconnect is also evaluated using MCNC benchmarks. Fig. 25 shows the critical path delay and total energy of interconnect implementing 20 benchmarks. The numbers are calculated by running VTR for place and route info of each benchmark, simulating each path, then summing up. As a result, when using the optimized low-swing interconnect, the delay and energy of the interconnect reduce by 68.4% and 47.5% on average, respectively.



Benchmarks	Reduction of Using the Optimized Low-Swing Interconnect	
	Delay (%)	Energy (%)
alu4	67.1	47.1
apex2	69.3	47.3
apex4	72.1	47.5
bigkey	66.1	47.0
clma	69.6	47.9
des	67.6	47.6
diffeq	60.3	47.1
dsip	69.6	47.1
elliptic	67.2	47.7
ex5p	72.0	47.3
ex1010	72.7	47.7
frisc	64.5	48.0
misex3	69.2	47.1
pdc	72.4	47.8
s298	68.5	47.7
s38417	66.8	47.7
s38584.1	66.9	47.6
seq	69.0	47.2
spla	74.9	47.8
tseng	62.3	47.3
Average	**68.4**	**47.5**

Figure 25: The energy reduction of MCNC benchmarks when utilizing the optimized low-swing interconnect in 130nm CMOS

3.6 Level Shifter Insertion

3.6.1 Overview

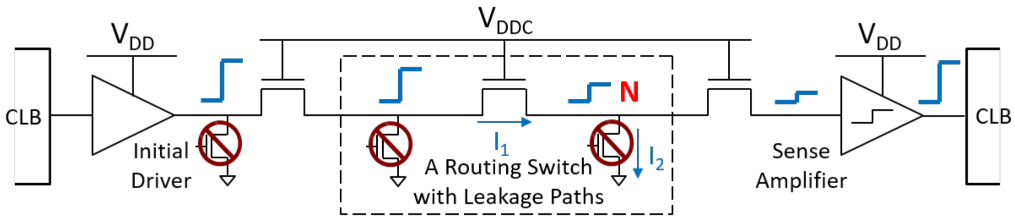


Figure 26: Signal Degradation along Low-Swing Interconnect Paths

Since the low-swing interconnect uses pass-gates without buffer to implement routing switches, it is not good at passing logic 1. In above-threshold operations, there is a voltage drop of V_T between drain and source of pass-gates, and the signal swing keeps at $V_{DD} - V_T$ after passing through the subsequent pass-gates. Using transmission-gates can avoid the voltage drop when needed. However, this is a different story in near/sub-threshold region, where V_{DD} is already about or lower than V_T. In this region, there is a voltage drop across both pass-gates and transmission-gates with a value depends on current. Fig. 26 shows a diagram of how does it work. In the routing switch inside the dotted line box, I_1 is charging node N, which is also discharged by the leakage current I_2 simultaneously. The signal degradation in near/sub-threshold operations applies to every routing switches along paths. As a result, the signals will keep degrading and ultimately become too small to be captured by the level shifters at the end of paths. Although the switching threshold of the level shifter used in this research [22] [23] is as low as 0.09V at $V_{DD} = 0.4$V, level shifters are still potentially needed to be inserted into long paths to regenerate signals.

3.6.2 Impact to Signal Swing

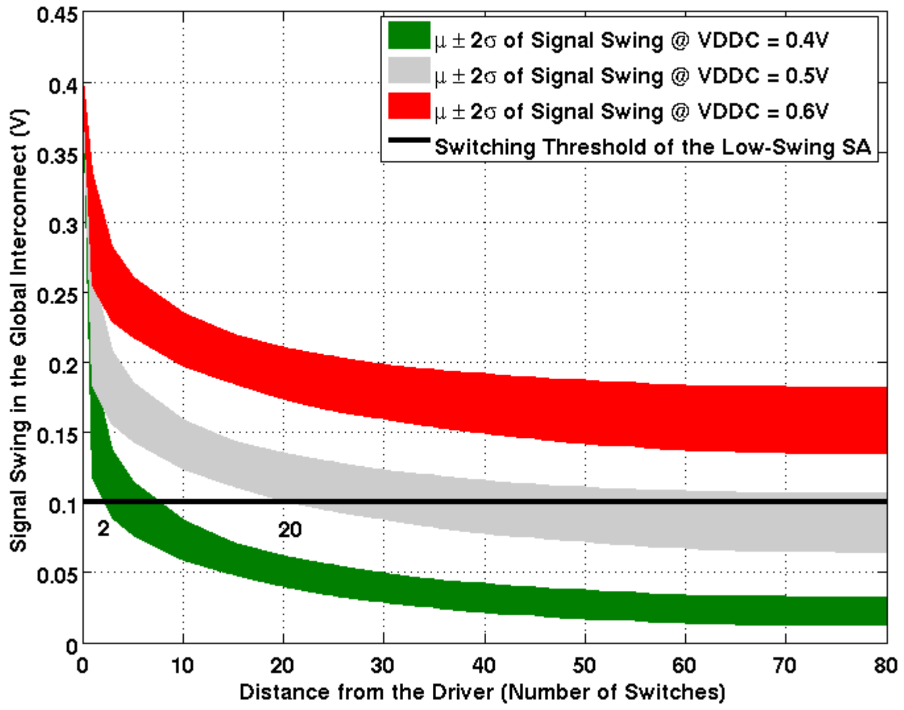
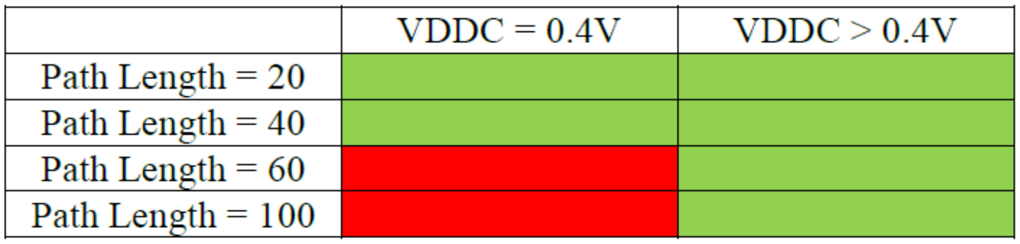


Figure 27: The signal swing of low-swing paths with varying length and the switching threshold of the level shifter @ $V_{DD} = 0.4$V in 130nm CMOS

In order to determine whether level shifters are needed, the first thing to look at is how does the signal swing changes along a path in simulation. Fig. 27 shows the signal swing change after passing through different numbers of routing switches at $V_{DD} = 0.4$V. In the figure, the x-axis represents the number of routing switches signals have passed through, while the y-axis represents the value of the signal swing at the end of the path. The areas in different colors represent the $\mu \pm 2\sigma$ range (from Monte Carlo simulations in SPICE) of the swing at different V_{DDC} values. The areas in red, grey, and green represent V_{DDC} of 0.6V, 0.5V, and 0.4V, respectively. The black horizontal line represents the mean value of the switching threshold of the level shifter. The x-value where the switching threshold of the level shifter and the signal swing intersect represents the maximum number of switches signals can pass through without requiring any level shifters in the middle of paths. If variation is ignored, a level shifter is needed after the signal passes through 5, 40, or over 80 switches when V_{DDC} equals to 0.4V, 0.5V, and 0.6V, respectively. If considering variation, the switch numbers just mentioned become 2, 20, and over 80. When $VDDC > 0.6V$, no level shifters are needed to maintain functionality of a path shorter than 80. Similar results are also observed in 32nm SOI node, where the signal swing never dropped below the switching threshold of the level shifters at $V_{DD} = V_{DDC} = 0.4V$ even under processing variation.



	VDDC = 0.4V	VDDC > 0.4V
Path Length = 20	🟩	🟩
Path Length = 40	🟩	🟩
Path Length = 60	🟥	🟩
Path Length = 100	🟥	🟩

Figure 28: Measured shmoo plot of signal degradation @ V_{DD} = 0.4V in 130nm CMOS

In addition to simulations, chip measurement results are desired to verify the conclusions on silicon. The test chip mentioned earlier in this chapter is also good for this research. The Shmoo plot in Fig. 28 shows the measured functionality of paths including signal degradation at V_{DD} = 0.4V. In the figure, green means the signal can be captured by the level shifter after passing through the corresponding number of switches at the corresponding V_{DDC}, and red means the signal swing is too small to be captured. As shown, the level shifter at the end of paths successfully captured the signals after passing through at least 100 switches when $VDDC \geq 0.5V$, but can only capture signals in paths shorter than 60 when V_{DDC} = 0.4V. If V_{DD} is higher than 0.4V, no level shifters are needed for paths shorter than 80, which is the largest path length observed in FPGAs implementing MCNC 20 benchmark suite. These measurement results are verified on multiple chips. From functionality point of view, no level shifters are needed except the one at the end of paths.

3.6.3 Impact to Energy Efficiency

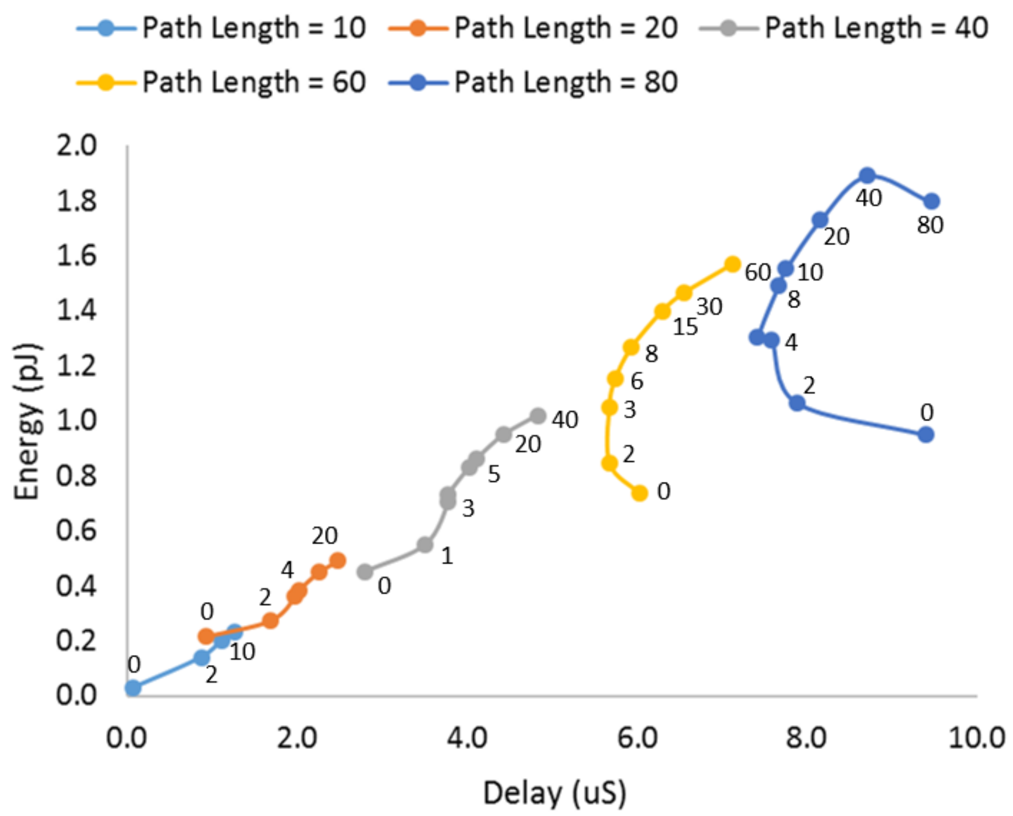


Figure 29: Measured ED curves of paths with different numbers of inserted level shifters @ $V_{DD} = 0.4$V in 130nm CMOS

Inserting level shifters affects not only functionality, but also delay and energy of paths. Inserting level shifters increases the total capacitance load in the interconnect, resulting in increased dynamic energy. However, the influence on delay after inserting repeaters is unclear. Thus, I configured the test chip to different length with different number of inserted level shifters to explore that. As shown in Fig.

29, the measurement results show that increasing the number of level shifters always increase both delay and energy for paths shorter than 80. The number beside each point represents the number of level shifters inserted. These results indicate no level shifters are needed in terms of path delay and energy. Fig. 30 shows the simulation results of the same thing, but in 32nm SOI. Inserting level shifters into paths shorter than 40 results in larger delay and energy. However, for longer paths, inserting 2 to 8 level shifters can reduce path delay effectively with energy overhead. Thus, the design decision in 32nm SOI depends on which metric is more important case by case.

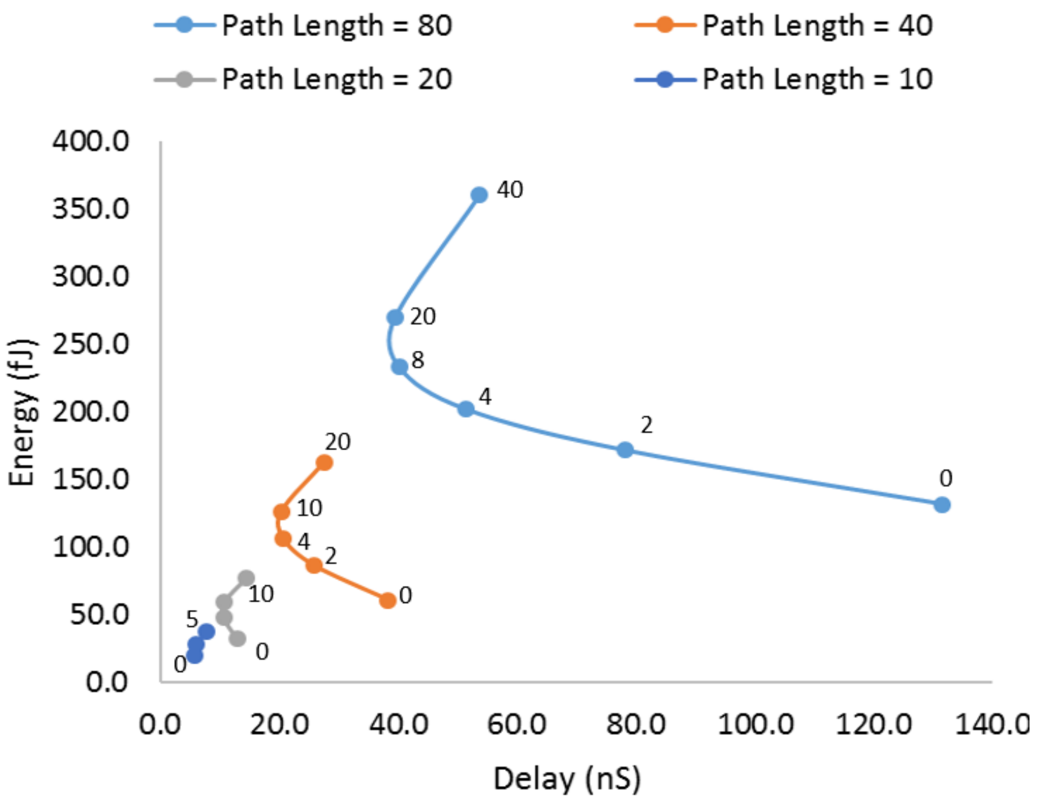


Figure 30: Simulated ED curves of paths with different numbers of inserted level shifters @ V_{DD} = 0.4V in 32nm SOI

3.7 Switch Box Layout

3.7.1 Overview

The value of parasitic resistance and capacitance of a wire segment between two routing switches is highly related to the total delay and energy of FPGA interconnect. The length of a wire segment equals to the dimension of a FPGA tile (as shown on the left of Fig. 31), which is the width of a CLB plus a switch box. Since RC value is linear proportional to the length of wire segments, minimizing the area of switch boxes (that also minimize the lengths of the intra-switch wires) could effectively reduce interconnect delay and energy.

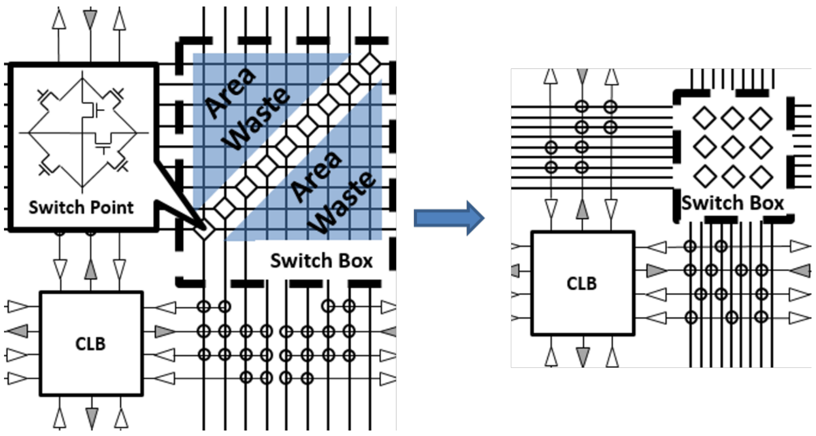


Figure 31: The diagram of traditional subset switch box (left) and the proposed folded-subset switch box (right)

3.7.2 Subset Switch Box Area Reduction

The switch box topology shown on the left side of Fig. 31 is a widely used one called Subset. It is built up with switch points that locate on each track of the interconnect channel. Traditionally, people make compact switch point layout separately, and then align the switch points on the diagonal of switch boxes. Doing so makes inter-switch routing simple in the layout. However, large triangle area in switch boxes is wasted and the minimum space between CLBs becomes determined by the dimensions of switch points instead of the minimum space between metal pieces defined by the technology specification, as shown in the figure. In this research, I designed a Folded-Subset switch box layout which breaks the alignment of the switch points and squeeze them into a minimum square area, as shown on the right side of Fig. 31.

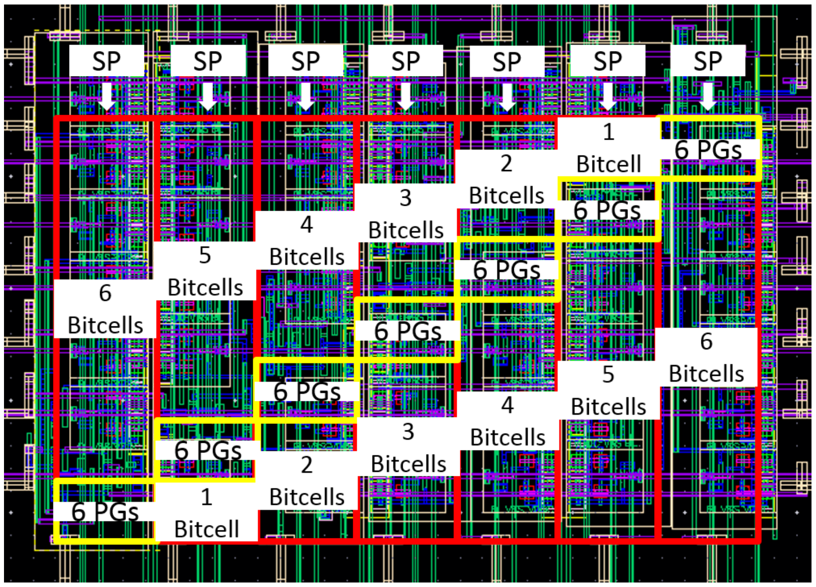


Figure 32: Layout of a 7-switch-points cluster

Minimizing area increases the local routing complexity of the switch boxes. To simplify routing and enable scalability, the layout is done hierarchically. The strategy is to combine several switch points as a cluster in layout, then creating the full switch box layout by tiling up several clusters. Each switch point is made of six pass-gates with a SRAM bit-cell attached to each pass-gate for enabling programmability. When making the cluster, the six pass-gates are firstly grouped, then lined it up with the six bit-cells to create the layout of a switch point, as shown in Fig. 32. However, simply tiling up several identical switch points as a cluster causes severe routing congesting problems. Fig. 33 shows the strategy to solve this problem. Seven different switch points are made by shifting the location of the pass-gates, then line them up. This cluster is the basic cell building up the 84-track switch box layout in the custom FPGA. As shown in Fig. 33, twelve of the clusters are used to make the Folded-Subset switch box. The proposed layout is scale-able to varying number of tracks without modifying the cluster.

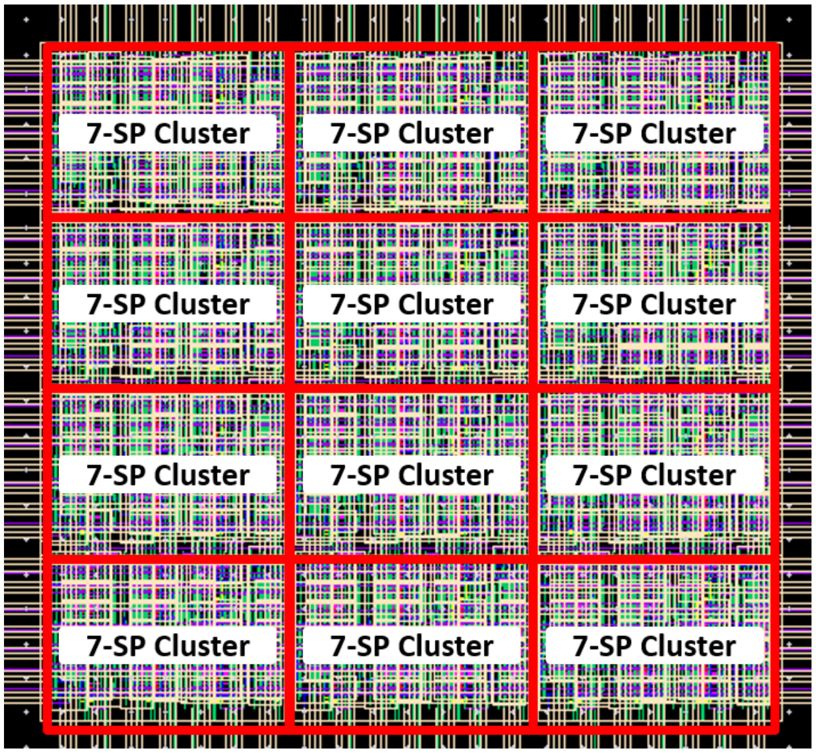


Figure 33: Layout of an 84-track switch box

In Fig. 34, the area comparison of the traditional Subset topology and the proposed Folded-Subset topology is shown. Due to the cluster-based hierarchical layout structure, the proposed switch box does not available for all channel widths. However, the channel width granularity is fine enough from FPGA design point of view. From observing the six cases shown in the table, higher channel width leads to higher area reduction by using the proposed layout. At channel width equals to 28, the area can be reduced by 8.7x, while at channel width equals to 280, this number increases to

86.6x. Although there is a channel width gap between 84 and 280 in the table, our switch box can be used for numbers in between, such as 112, 140, 168, 210, and 245.

Channel Width	Area of Subset SB (um * um)	Area of Folded-Subset SB (um * um)	Area Reduction
28	274.4 * 109.2	70.6 * 49.0	8.7X
42	411.6 * 163.8	70.6 * 73.5	13.0X
56	548.8 * 218.4	70.6 * 98.0	17.3X
63	617.4 * 245.7	105.9 * 73.5	19.5X
84	823.2 * 327.6	105.9 * 98.0	26.0X
280	2744.0 * 1092.0	176.5 * 196.0	86.6X

Figure 34: Area Comparisons of traditional Subset and the proposed Folded-Subset SBs in 130nm CMOS

3.7.3 Wilton Switch Box Area Reduction

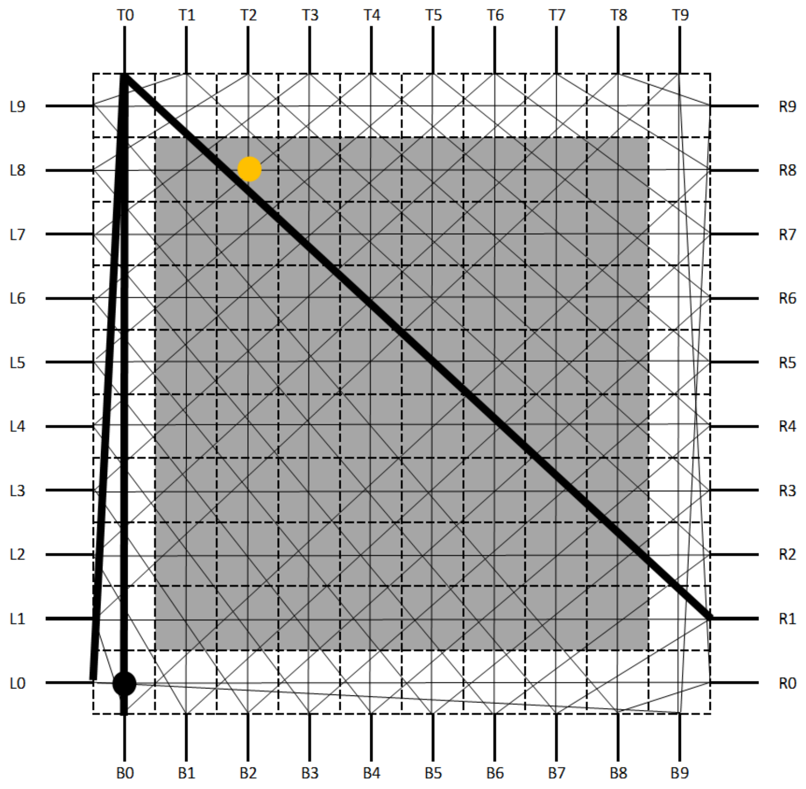
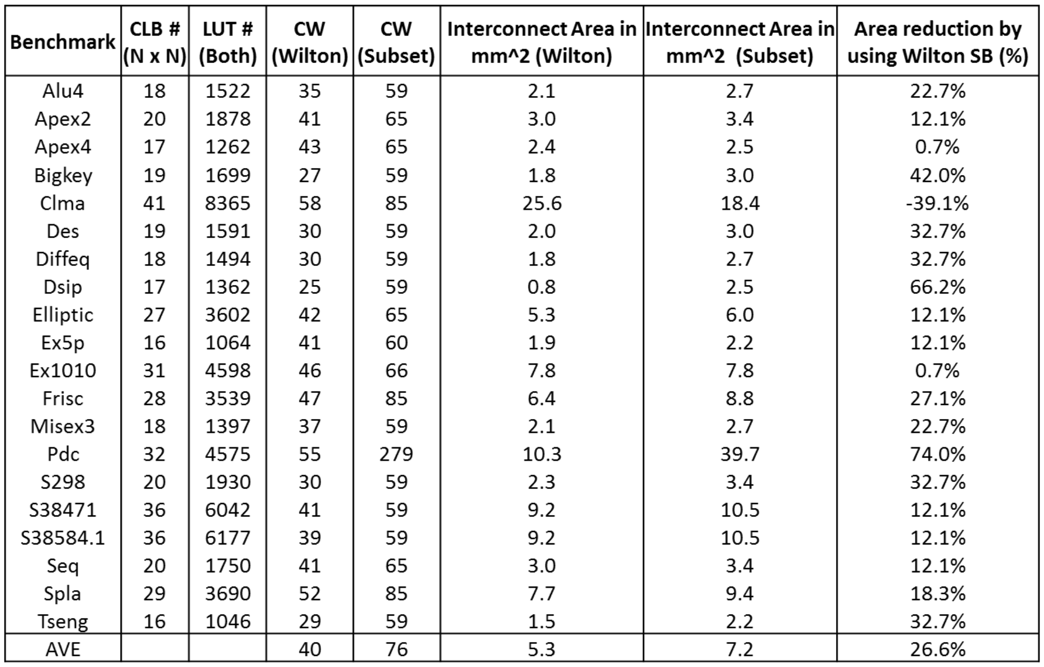


Figure 35: The diagram of a 10-track Wilton switch box

The Subset switch box is widely used because it is straightforward to implement and is easy to scale. For example, if designers want to expand a 10-track switch box to 12-track, they can just add two more switch points without redesigning the 10-track structure, as shown in Fig. 31. This is very beneficial when building several FPGAs with different number of tracks. However, Subset is impossible to route a signal from one track to another, even though the Folded Subset optimized the layout area. Wilton is another type of switch box topology as shown in Fig. 35. While track T0 connects to L0/R0/B0 in Subset, it connects to L0/R1/B0 in Wilton instead, providing an opportunity for routing between tracks. Thus, FPGAs using Wilton switch box always uses more tracks than FPGAs using Subset, as shown in Fig. 36.



Benchmark	CLB # (N x N)	LUT # (Both)	CW (Wilton)	CW (Subset)	Interconnect Area in mm^2 (Wilton)	Interconnect Area in mm^2 (Subset)	Area reduction by using Wilton SB (%)
Alu4	18	1522	35	59	2.1	2.7	22.7%
Apex2	20	1878	41	65	3.0	3.4	12.1%
Apex4	17	1262	43	65	2.4	2.5	0.7%
Bigkey	19	1699	27	59	1.8	3.0	42.0%
Clma	41	8365	58	85	25.6	18.4	-39.1%
Des	19	1591	30	59	2.0	3.0	32.7%
Diffeq	18	1494	30	59	1.8	2.7	32.7%
Dsip	17	1362	25	59	0.8	2.5	66.2%
Elliptic	27	3602	42	65	5.3	6.0	12.1%
Ex5p	16	1064	41	60	1.9	2.2	12.1%
Ex1010	31	4598	46	66	7.8	7.8	0.7%
Frisc	28	3539	47	85	6.4	8.8	27.1%
Misex3	18	1397	37	59	2.1	2.7	22.7%
Pdc	32	4575	55	279	10.3	39.7	74.0%
S298	20	1930	30	59	2.3	3.4	32.7%
S38471	36	6042	41	59	9.2	10.5	12.1%
S38584.1	36	6177	39	59	9.2	10.5	12.1%
Seq	20	1750	41	65	3.0	3.4	12.1%
Spla	29	3690	52	85	7.7	9.4	18.3%
Tseng	16	1046	29	59	1.5	2.2	32.7%
AVE			40	76	5.3	7.2	26.6%

Figure 36: The interconnect area of FPGAs implementing MCNC benchmarks

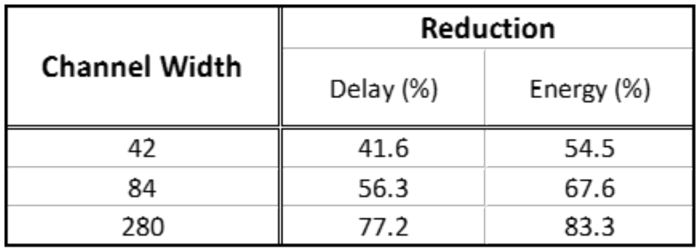
Since Wilton switch box provides intra-track connection, the nested tracks make its layout very complicated and hard to scale. Even adding one more track requires completely redesigning the switch box. The most straightforward way to layout a

Wilton switch box is adding a pass-gate for each potential connection from one track on one side (top/bottom/left/right) to any track on the other side. For example, T0 can connect to L0-L9/B0-B9/R0-R9. Doing so simplifies the process of making multiple switch boxes with different number of tracks. Designers just need to build a big mesh of pass-gates then make connections where needed. However, this method wastes a big portion of area on pass-gates prepared for non-exist connections. For example, the connection between T0 and R9 in Fig. 35 is clearly not exist. Another method is to add pass-gates only at intersections where physical connections exist. As an example in Fig. 35, since T0 connects to L0, a switch (represented as the black dot) is places. Each square area within dotted lines is a potential spot for "dropping" pass-gates. For a 10-track switch box, there are 100 such spots in total. However, Fig. 35 indicates only 60 connections exist. Thus, this method wastes the area of 40 "spots". To minimize area, I squeezed 60 pass-gates into the dark grey area in Fig. 35, which has 64 "spots" available. The methodology is to firstly "drop" all pass-gates at connections that originally exist in the grey area, like the yellow dot connecting T2 and L8. Then, "drop" other pass-gates like the black dot to grey spots left.

Fig. 36 shows the number of tracks needed and the switch box area of FPGAs implementing MCNC benchmarks. Using Wilton switch boxes, FPGAs needs 47.3% less tracks compared to using (Folded-)Subset. However, this number drops to 26.6% when comparing FPGA area, since Wilton switch box requires a big portion of area on its complicated internal routing.

3.7.4 Energy Efficiency Improvement

The proposed folded-subset switch box not only reduces FPGA area, but also increases FPGA interconnect energy efficiency. The simulated results of paths using subset switch box and folded-subset switch box are compared in Fig. 37. Since the area of switch boxes highly depend on number of tracks in the channel, the channel width needs to be considered in evaluating the energy efficiency improvements. To implementing MCNC benchmarks and the majority of applications in chapter 5, 84 tracks are needed. In this case, using the proposed switch box layout results in 56.3% delay decrease and 67.6% energy reduction of the global interconnect. Channel widths of 42 and 280 are also considered for different FPGA sizes.



Channel Width	Reduction	
	Delay (%)	Energy (%)
42	41.6	54.5
84	56.3	67.6
280	77.2	83.3

Figure 37: The energy efficiency improvement by using folded-subset switch box

3.7.5 Cross Talk

As the switch box area in physical layout is reduced, the space between two interconnect tracks is also reduced and becomes close to the minimum space defined by design rules of technology. The parasitic capacitance between tracks (42fF in 130nm CMOS) and between a track and substrate (10fF in 130nm CMOS) then make the custom interconnect suffering from cross-talk problem. In the interconnect model used for running simulations in this chapter, the cross-talk effect is considered by adding average parasitic capacitance between the adjacent tracks in to the Pi model of wire segments. As a result, including cross-talk increases the critical path delay of alu4 (a benchmark in MCNC suite) by up to 64.3%

3.8 Future Research

Todays data centers consume a vast amount of energy. According to the Department of Energy [27], data centers in the United States consumed about 70 billion kilowatt-hours of electricity in 2014, representing 2% of the countrys total energy consumption, up from 0.8% in 2000 [28]. In particular, companies spend roughly twice as much on powering and cooling data centers than the servers and components themselves [29]. There is an urgent need to explore energy efficiency improvement strategies for the hardware of data centers to reduce financial and environmental costs associated with high volume data centers. In addition to energy efficiency, data center computing platforms also require high speed and flexibility for rapidly-changing applications. Therefore, a growing number of companies have begun using FPGAs as the primary choice for computing platforms in several applications. However, existing FPGAs are not yet optimalthe energy efficiency of FPGAs could be further improved via circuit-level solutions. The low-swing design is already proved can reduce FPGA interconnect energy significantly in near/sub-threshold. It is also worth to explore if the same design can reduce energy of high-performance FPGAs used in data centers.

There are challenges in applying the custom interconnect to high-performance FPGAs. Firstly, modern FPGA interconnect architecture is different from the model used in this chapter. Compared to basic interconnect topology, modern interconnect 1) uses smaller CLBs for better flexibility, 2) breaks up the switch point structure and instead uses combination of wires with different spans or long express wires across the FPGA on both directions, 3) inserts registers everywhere to enable deep pipelining, and 4) has a variety of hard IPs. While each CLB used in the custom low-power FPGA contains 8 LUTs, modern CLB has only 2 LUTs. This makes the logic cluster smaller and transfers the burden of local interconnect into the global interconnect of the FPGA. On the other hand, the multi-span interconnect wires, the everywhere-registers, and the hard IPs dramatically increase the complexity of interconnect modeling, the value of equivalent capacitance, and the characteristics of leakage paths. To make matters worse, since the everywhere-registers are supposed to connect to the main V_{DD} while the signal swing is smaller, additional level converters may be needed to avoid large static current. These issues make the actual energy reduction we can achieve by applying the low-swing design to the custom slice unpredictable. The second challenge is that the speed and energy requirements on high-performance and low-power applications are quite different, prompting different design strategies for circuits. First, while the voltage drop across the NMOS pass-gates is the threshold voltage at nominal voltage, it is more complicated in near/sub-threshold, where the voltage drop depends on the balance between on-current and the leakage current everywhere in the interconnect. Thus, the signal swing might be quite different at varying supply voltages and a simple level shifter design may not avoid large static current in both cases. Secondly, since variation is a significant concern in near/sub-threshold, robust circuits need to be developed for low-power applications. However, such a robust design is not needed in high-performance applications, but the additional design degrades speed and increases energy. For example, to improve robustness, the transistor size needs to be increased in near/sub-threshold, but this change increases energy at nominal voltage. The metrics and criteria of energy-robustness balancing is a design challenge. Finally, leakage energy dominates in sub-threshold, indicating a focus on leakage energy reduction, while this is not an issue at nominal voltage.

4 Per-Path Voltage Scaling and Power-Gating

4.1 Motivation

It is desired that low-power FPGAs can be as energy efficient as possible due to the strict power budget. Although FPGA interconnect has been optimized in Chapter 3, architectural level optimization is also needed to further reducing energy. Dynamic voltage scaling (DVS) is a widely used technique where the supply voltage of computing devices can be increased or decreased dynamically depending on the load or environment changes during usage. Properly using DVS guarantees the system to consume energy at just the minimum to meet the speed requirement at all time. In FPGAs, the supply voltage can be adjusted either spatially or temporally. The spatial adjustment means applying different voltages to different paths on FPGAs. In ASICs and processors, this technique is usually called multi-V_{DD}. Since dynamic energy is proportional to V_{DD}^2, reducing the supply voltage on non-critical paths dramatically reduces FPGA energy while still maintaining the overall speed. Since FPGAs are flexible hardware, the critical path keeps changing when FPGAs switch between different applications. Thus, voltage values need to be reassigned to each path on FPGAs before using. To distinguish this, I call the spatial DVS used in this research "per-path voltage scaling" instead of multi-V_{DD}. In addition to "per-path voltage scaling", the supply voltage of FPGAs can also be adjusted at run-time to minimize energy waste due to process/voltage/temperature (PVT) variations and dynamically changing speed requirements. For example, in the absence of scene changes in MPEG video processing, reducing the processing speed by a reasonable amount will not affect the overall system performance due to the fixed fetching rate of frames [30]. When processing speed is no longer critical, power and energy can be saved by scaling down the supply voltage to or near the minimum energy point of the system. This temporal DVS is simply called DVS in this document. Power-gating is another low-power technique can be applied to FPGAs. The principle of power-gating is to cut off the connections of the idle circuit components from supply voltages to reduce leakage energy without speed overhead. Although DVS and power-gating are already used in FPGAs, additional energy reduction is expected by combining these techniques with low-swing interconnect introduced in Chapter 3.

4.2 Prior Art

4.2.1 Per-Path Voltage Scaling & DVS

Multi-voltage technique has been widely used in computing devices. For example, SoCs usually have one always-on voltage domain for control logic, and several voltage domains for processors, memory blocks, and IOs for different speed and energy requirements. The latest commercial FPGAs also have different voltage domains for embedded processors and IPs. Since FPGA core fabric consumes only a small portion of energy, assigning one voltage domain to it is good enough. The energy reduction of applying multi-voltage technique on the core can't outweigh the overhead of control circuit and additional routing. However, since the core fabric is the major energy consumer of low-power FPGAs, existing works have tried multi-voltage on the core. In [16], a FPGA architecture that applies dual-V_{DD} on CLBs and interconnect is introduced. The authors reduced the dynamic energy by applying nominal voltage to the critical path, and applying a lower supply voltage to all the non-critical paths. While [16] only explored designs on algorithm and architecture levels, [15] uses the similar technique with circuit level optimization. However, although some work implemented dual-V_{DD} on CLBs in chip designs, no existing work uses per-path voltage scaling on FPGA interconnect due to the high area overhead of adding headers to the buffers in switch boxes. Also, the designs in the existing works mainly focus on the traditional buffer-based interconnects at the nominal voltage, while wireless IoT applications require FPGAs optimized at near/sub-threshold. The low-swing design introduced in Chapter 3 provides a good opportunity. It is proven to be an energy-efficient design in near/sub-threshold. Also, since there are no buffers inside the switch boxes in the low-swing interconnect, a big portion of the potential area overhead no longer exists.

Dynamic voltage scaling has been used as a low-power technique on processors for years. However, most of the existing works focus on theoretical estimations and simulations of applying DVS on processors. For example, researchers developed an algorithm of real-time DVS in [31]. Although they applied it to the operating system and get an energy reduction of 20% to 40% on hardware platforms, neither circuit implementation nor verification is discussed. In [32], researchers dig into this topic more by measuring power of a processor running H.263 video benchmark. They build testing station on board and observed a 20x power reduction at a cost of 5x speed degradation. However, this work adjusts voltage from outside with additional equipment, which is not a solid on-chip solution for low-power applications. Recent years, more circuit level solutions for DVS are published as higher requirement on system power. In [33], a 77% efficiency DC/DC converter is proposed, along with other similar works enabling high efficiency on-chip DVS. In [34], a processor using DVFS achieves a 16% power reduction. In [30], an ultra-dynamic-voltage-scaling circuit solution is proposed. It allows circuit to operate as low as 0.33V with an energy reduction of 9x, which is very beneficial for low-power ASIC and processors. While more and more processors use DVS, this technique has not been widely used on FPGAs. Only few work tried to apply DVS to FPGAs using off-chip solutions. For example, in [35], researchers applied DVS to a commercial FPGA to solve PVT variation problem using a computer.

4.2.2 Power-Gating

Power-gating is a leakage reduction technique widely used in all kinds of chips today. In [36], researchers build an 8-core media processor in 65nm CMOS. By using power-gating, they reduced leakage power by up to 98%. In [37], fine-grained power-gating along with other low-power techniques helped designers reducing the leakage energy of a 64Mb SRAM by 37%. In ultra-low-power applications, where supply voltage is usually reduced to near/sub-threshold, using power-gating is even important because of the increased portion of leakage power. For example, [11] uses power-gating to minimize leakage of a self-powered body sensor SoC that has extremely low power budget. As FPGAs already consume higher power than ASICs, using techniques like power-gating is essential. In [38], researchers achieved a 99% leakage reduction by applying power-gating to look-up-tables. In [39], designers also applied power-gating to FPGA interconnect, reducing leakage power by 70% 84%. However, no existing work examined how much FPGA interconnect leakage energy could be saved in near/sub-threshold operations. Especially, nobody tried to apply power-gating to the configuration bit-cells, which is a big source of leakage energy in near/sub-threshold.

4.3 Per-Path Dynamic Voltage Scaling

4.3.1 Architecture Overview

Low-power FPGAs are expected to be as energy efficient as possible. Reducing supply voltage on non-critical paths can effectively minimize energy while maintaining the overall FPGA speed [6]. Per-path voltage scaling is a low-power technique based on this concept. As shown in Fig. 38, the circuits in dark-gray represent the critical path that are attached to V_{DDH} through headers. The circuits in mid-gray represent non-critical paths that are attached to V_{DDL}. The circuits in light-gray are in idle mode and are power-gated. The energy reduction is completed by turning on/off a pair of header transistors. In this research, two voltage rails are used to implement per-path voltage scaling. Circuits on the critical path are attached to a higher supply voltage V_{DDH}, while the rest of the circuits are attached to a lower voltage V_{DDL}. When both transistors are turned off, the circuit component is power-gated in order to reduce leakage [16] [15].

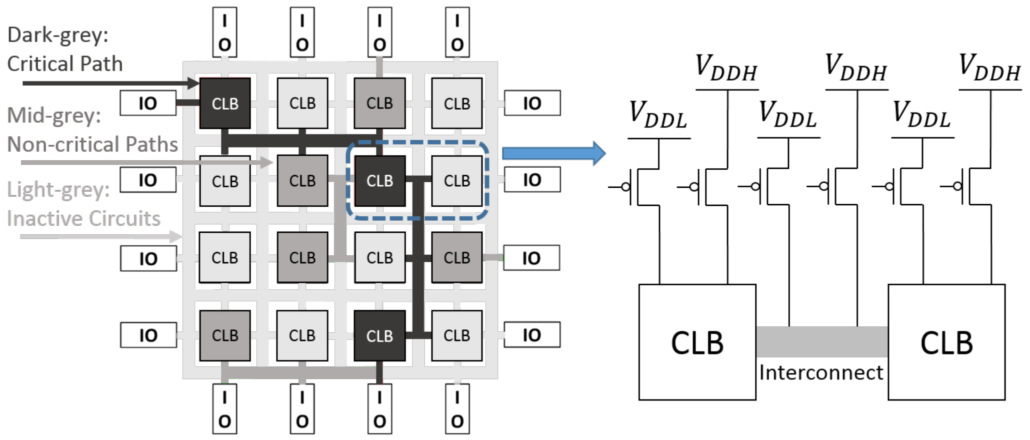


Figure 38: The concept diagram of applying per-path voltage scaling and power-gating techniques to FPGA interconnect

Multi-voltage has been proven to be capable of minimizing energy in ASICs, processors and FPGA CLBs. However, this technique was not easy to be applied to FPGA interconnects due to the large area overhead. Fig. 39 shows the interconnect model discussed in Chapter 3, but with additional headers enabling per-path voltage scaling. In each switch box, a configuration bit-cell is used to control the switch. Because the traditional interconnect fabric has buffers in each switch box, implementing per-path voltage scaling on the interconnect requires adding headers and configuration bit-cells to each switch box. This will substantially bloat the already large area of the switch boxes. The size of headers usually larger than transistors in buffers to avoid voltage drop. Fig. 40 shows the estimated area overhead of using per-path voltage scaling. When header size is 10x, the overhead could as large as 43%. For this reason, although many papers evaluated the potential energy reduction by using per-path voltage scaling, no design use this technique on chips. The low-swing interconnect, which removes buffers from switch boxes, discussed in Chapter 3 solved this problem. As shown in Fig. 39, the headers used for the voltage assignment are not needed anymore and the area overhead is reduced to near zero. In this chapter, the interconnect model in Fig. 39 will be used for evaluating energy reduction of using per-path voltage scaling.

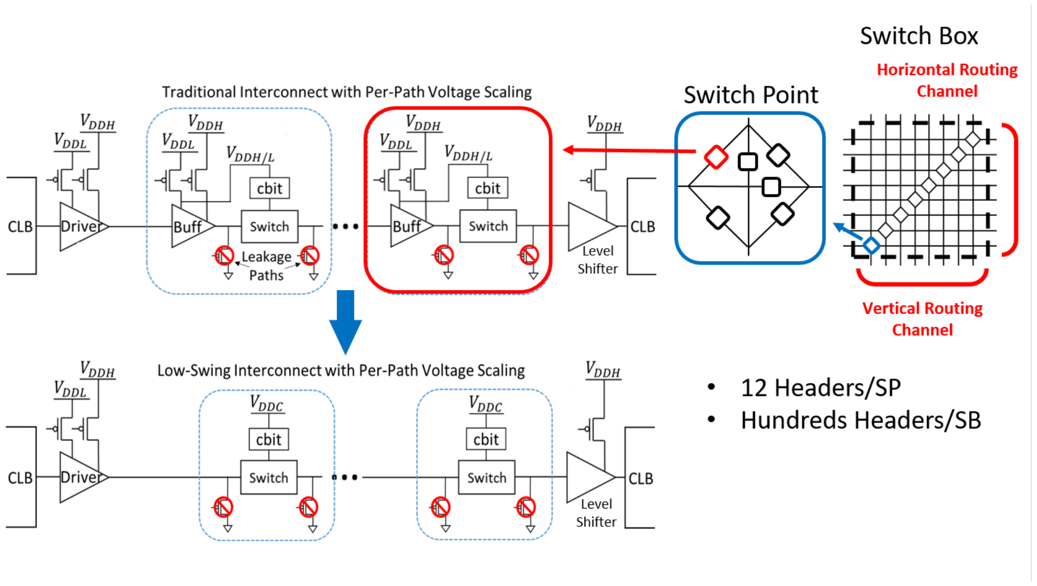
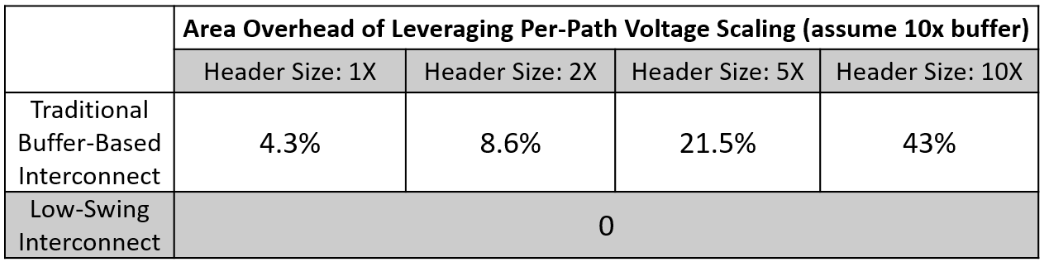


Figure 39: The comparison of applying per-path voltage scaling on the traditional interconnect and the low-swing interconnect

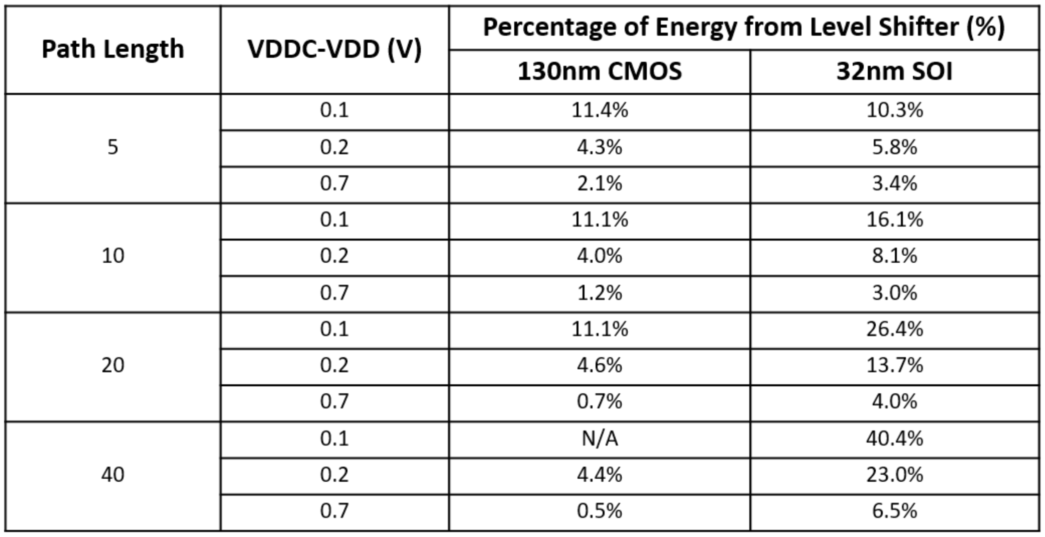
The bit-cells in switch boxes are attached to an additional boosted supply V_{DDC}, which is used to adjust path delay and energy dynamically. This method is more energy-efficient and less sensitive than adjusting the main supply voltage V_{DD}. Since the energy of V_{DDC} is purely leakage in the bit-cells, which is orders of magnitude smaller than the energy of V_{DD} applied to drivers and level shifters, the energy waste of voltage regulators used for adjusting V_{DDC} is much smaller than adjusting V_{DD}.



	Area Overhead of Leveraging Per-Path Voltage Scaling (assume 10x buffer)			
	Header Size: 1X	Header Size: 2X	Header Size: 5X	Header Size: 10X
Traditional Buffer-Based Interconnect	4.3%	8.6%	21.5%	43%
Low-Swing Interconnect	0			

Figure 40: The area overhead of headers introduced when using per-path voltage scaling

4.3.2 Level Conversion



Path Length	VDDC-VDD (V)	Percentage of Energy from Level Shifter (%)	
		130nm CMOS	32nm SOI
5	0.1	11.4%	10.3%
	0.2	4.3%	5.8%
	0.7	2.1%	3.4%
10	0.1	11.1%	16.1%
	0.2	4.0%	8.1%
	0.7	1.2%	3.0%
20	0.1	11.1%	26.4%
	0.2	4.6%	13.7%
	0.7	0.7%	4.0%
40	0.1	N/A	40.4%
	0.2	4.4%	23.0%
	0.7	0.5%	6.5%

Figure 41: The percentage of interconnect energy from level shifters under different conditions

Circuit using the multi-V_{DD} technique requires the designers to carefully manage cross-voltage-domain issues. That is, isolation and level shifting circuit is needed at the interface of two voltage domains to avoid short circuit current. The per-path voltage scaling technique brings the cross-voltage-domain issue to FPGA interconnect. Since the critical path varies from application to application, each interconnect path in FPGAs has potential cross-voltage-domain problem. Thus, over thousands of level shifters needed to be inserted to traditional interconnect. However, the low-swing interconnect already has level shifters to regenerate low-swing signals. They can also be used to minimize the short circuit current of converting V_{DDL} back to V_{DDH}, so no additional circuit is needed. Cross-voltage-domain operations introduce additional energy in level shifters. Fig. 41 shows the simulated percentage of energy from level shifters under different conditions. At lower V_{DDC}, the signal swing is smaller and results in higher short circuit current. When V_{DDC} is high enough, the percentage of level shifters energy is lower than 5% for most paths.

4.3.3 Voltage Regulation

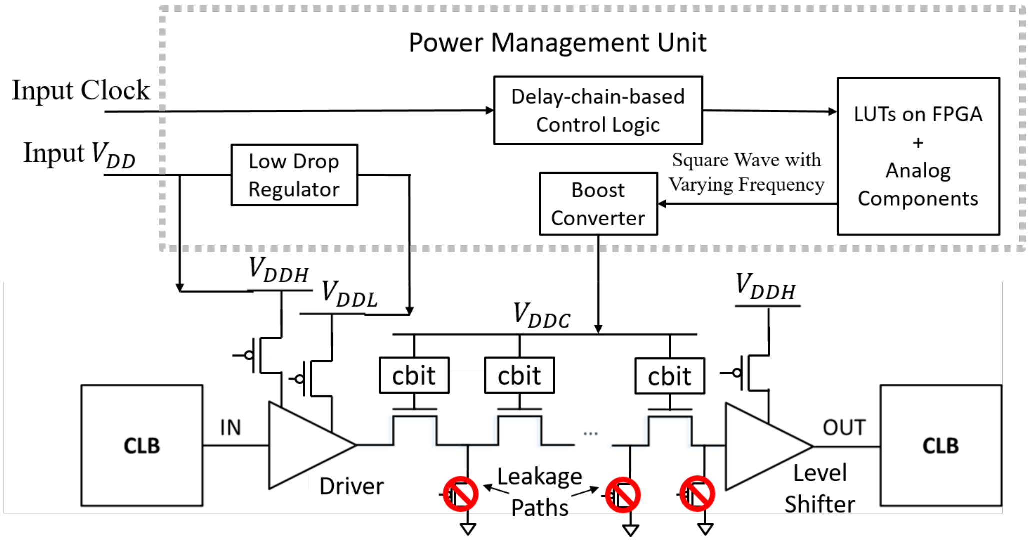


Figure 42: The concept diagram of the proposed architecture and the power management unit

There are three voltage sources mentioned so for implementing per-path voltage scaling and DVS: V_{DDH}, V_{DDL}, and V_{DDC}. However, providing all three supplies from off-chip is not realistic. For example, low-power applications like personal health sensors require very small form factor. Having multiple off-chip supplies usually indicates multiple batteries, which are large, heavy, and increase the chance of replacing. One solution is using only one battery and generating the other two voltage values on board. However, on board regulation does not energy-efficient or reliable due to long wires. The best strategy is performing voltage regulation on chip through a power management unit (PMU). How to design a PMU is beyond the scope of this dissertation. However, as the energy overhead of generating additional supplies on-chip is large, this part of circuit cannot be ignored. Thus, the voltage regulating strategy and hardware components are chosen and theoretically estimated.

When estimating the overall energy reduction of using per-path voltage scaling and DVS, the energy overhead of generating additional supplies needs to be taken into account. Fig. 42 shows one possible solution of the on-chip power management unit. The type of voltage regulator used to generate each supply is indicated. First of all, V_{DDH} directly comes from a chip pin. As what will be explained later in this chapter, the difference between V_{DDH} and V_{DDL} leading to the largest energy reduction is small (0.05V - 0.15V). In this case, low-drop voltage regulator (LDO) is chosen to generate V_{DDL} for its near-90% efficiency and small area.

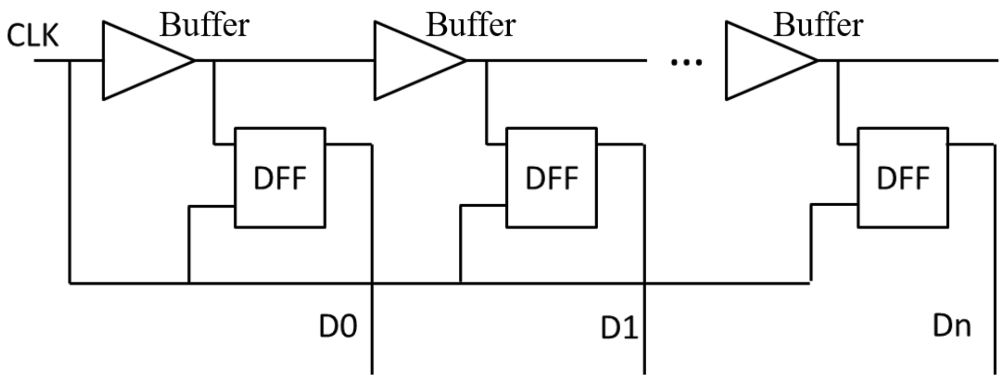


Figure 43: The concept diagram of the delay chain circuit and the proposed delay detector and voltage controller architecture

Due to the varying speed requirement of applications, we sometimes need to dynamically adjust V_{DDC}. To generate V_{DDC}, a boost converter is needed, as its value is higher than V_{DDH}. The frequency of the input square wave of the boost converter depends on, and is converted from, the varying target system speed of applications. In the existing works, people use switch-capacitor-based voltage regulators to control voltages according to the detected chip delay found by the commercial IBM Critical Path Monitor [40] or the Intel Droop Detector [41]. However, those designs are large, complicated, and time-consuming in the design process. In this research, a simpler voltage controller based on the idea of the Logic Delay Measurement Circuit (LDMC) discussed in [35] for low-power designs. As shown in Fig. 43, the delay-chain-based control logic takes the system clock as the only input. The frequency of this clock determines the number of buffers a high-to-low transition can propagate. When the falling edge of the clock arrives, the clock signal has half cycle to propagate though the delay chain before the rising edge arrives and triggers the flip-flops. Thus, only a portion of the flip-flops close to the input have outputs of 0, while the rest stay at 1. The on-chip LUTs along with analog components can be used to generate the input square waveform of the boost converter, taking the flip-flop outputs as control signals. In other word, by changing the frequency of the system clock, the V_{DDC} value will be adjusted to meet the new speed requirement at run-time. The method of adjusting V_{DDC} described above is just a reference. Better circuit solutions may occur, but beyond the scope of this dissertation. Also, the delay-chain-based control logic can also be implemented by LUTs if varying number of buffer stages are needed.

4.3.4 CAD Flow for VDD Assignment

For the remainder of Chapter 4, we define net to mean any signal path from an output of a CLB to an input of another CLB. Each net includes one or more switch boxes. We define path to mean any signal path from a FPGA input pad to an output pad. Each path involves multiple switch boxes and CLBs. By assigning the nets on the non-critical paths to V_{DDL}, the overall energy of the FPGA can be reduced. There are two important knobs for V_{DDH} & V_{DDL} assignment: the portions of the nets assigned to and the values of V_{DDH} & V_{DDL}. We need to find the best combination of V_{DDH} & V_{DDL} values that can minimize the interconnect energy without increase the critical path delay. In this research, a custom timing analysis tool is created to automatically do this optimization.

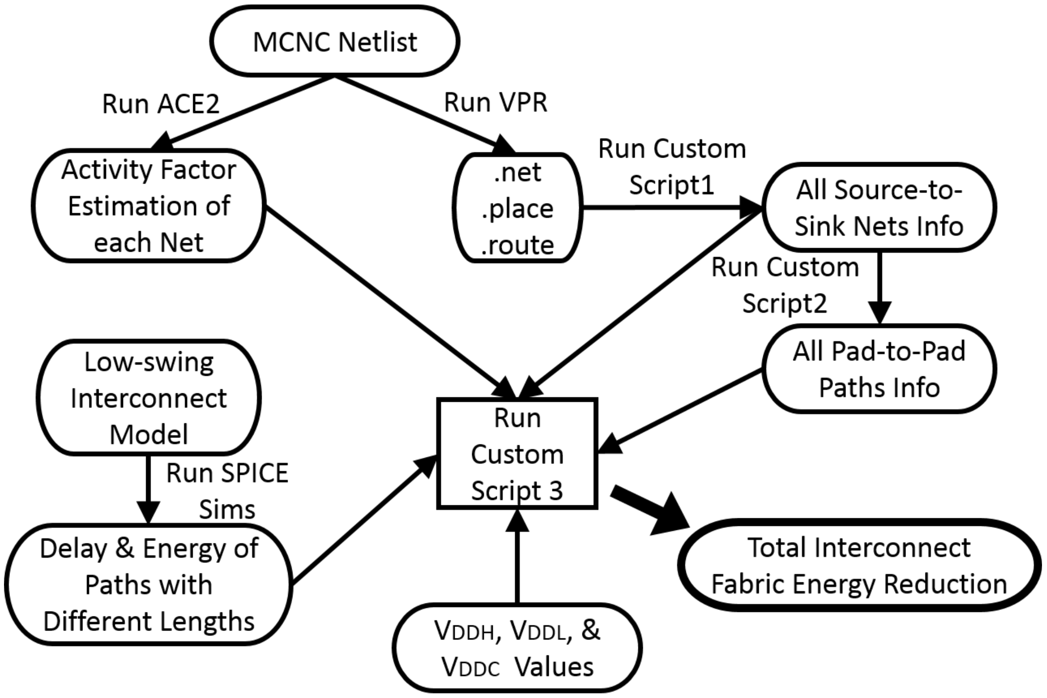

MCNC Netlist
Run ACE2
Run VPR
Activity Factor Estimation of each Net
.net .place .route
Run Custom Script1
All Source-to-Sink Nets Info
Run Custom Script2
All Pad-to-Pad Paths Info
Low-swing Interconnect Model
Run SPICE Sims
Delay & Energy of Paths with Different Lengths
Run Custom Script 3
Total Interconnect Fabric Energy Reduction
VDDH, VDDL, & VDDC Values

Figure 44: The flow chart of the custom multi-V_{DD} assignment tool

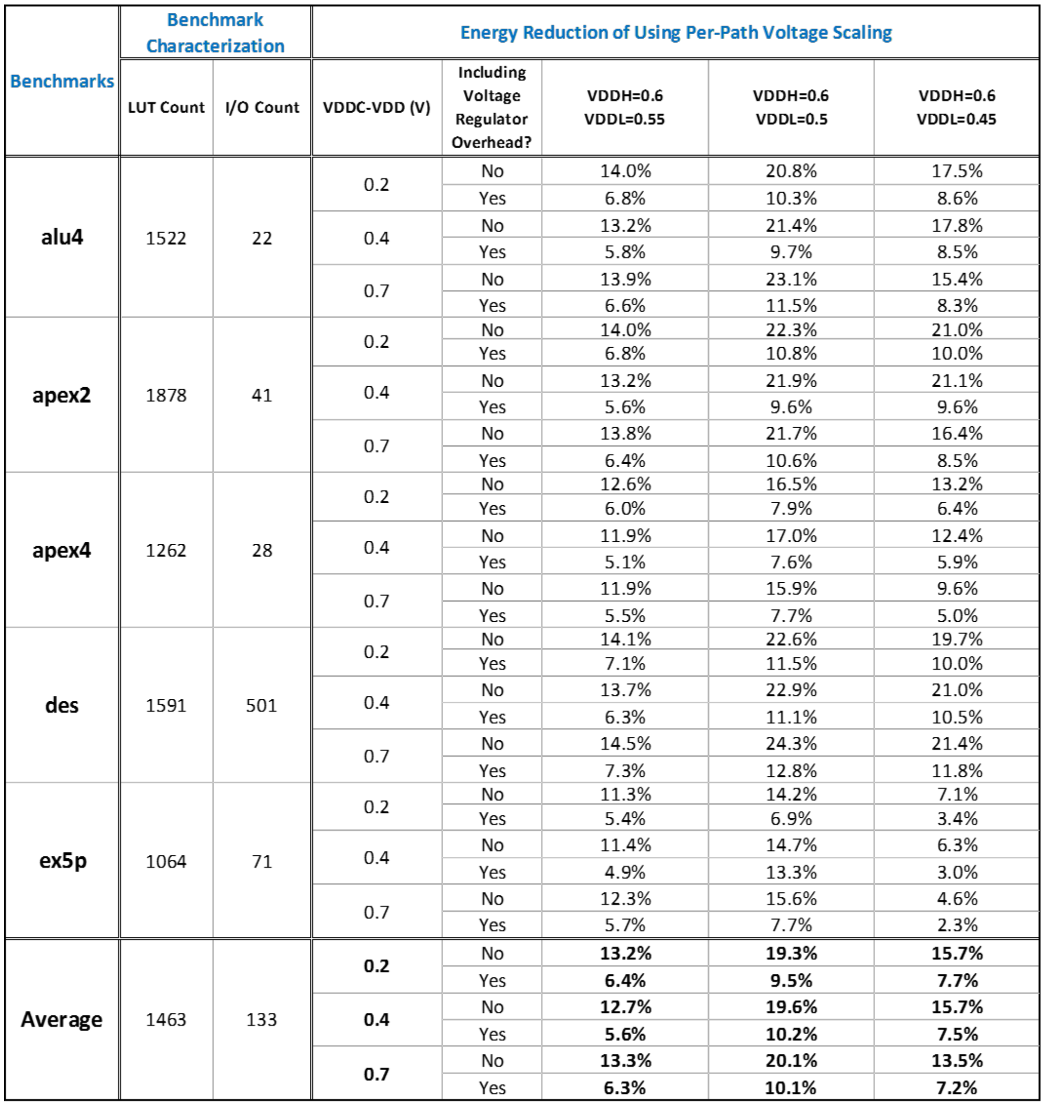
The custom tool is based on VTR [24], which can do complete timing analysis for many FPGA architectures. VTR estimates interconnect timing based on constant delay values of circuit components (switches, buffers, and wires) described in its architecture files. However, it assumes the circuit components of the same type always have same delay. This is not true for an architecture using per-path voltage scaling. For example, VTR assumes a switch in net # 1 and another switch in net # 2 have same delay. However, when applying V_{DDH} to net # 1 and V_{DDL} to net # 2, the delay of the two switches aren't the same. In this case, VTR timing analysis no longer accurate. To solve this problem, we can keep using VTR by creating a model for each net. This makes the architecture files extremely complicated. Also, since the nets assigned to V_{DDH} & V_{DDL} vary among applications, this method requires us to make a specific architecture file for each benchmark. For this reason, instead of using the entire VTR flow, I extracted the routing info (the length of each net, the start point and end point of each net, and how the nets build up paths) from VTR output files, then calculated the delay of each path by adding up the delay of each net on the path. Since the low-swing interconnect circuit and the operating voltage applied to it are very different from the assumptions of VTR, the delay of nets is obtained by running SPICE simulation. Compared to the timing analysis results of VTR, the custom tool finds the same critical paths. The only difference is the absolute delay values. Since the custom tool allows users to recalculate the delay of every path in the multi-V_{DD} assignment process, we can keep trying to assign different voltages to each net until the FPGA archives the lowest energy point without changing the critical path. This is what VTR cannot do.

The details of the custom tool are shown in Fig. 44. In order to do multi-V_{DD} assignment as while as timing analysis for a benchmark, we need the routing info from running VTR, the simulated delay and energy of each net from running SPICE, the activity factor of each net by running ACE 2.0 [42], and a script (the custom script 3 in Fig. 44) to do multi-V_{DD} assignment, critical path delay calculation, and energy saving calculation. In this script, a simple brute-force algorithm is used to initially assign all the nets to V_{DDH} and try to reassign V_{DDL} to each net. If the critical path does not change after the assignment, that net is kept on V_{DDL}, otherwise assign it back to V_{DDH}. The script exports the info of the portions of the nets assigned to V_{DDH} & V_{DDL}, the energy before and after using the multi-V_{DD} scheme, and the energy distribution of the FPGA interconnect at every V_{DDH} & V_{DDL} value combinations. Then the energy overhead of using the voltage regulator is estimated based on the exported info. To use the routing info from VTR, two additional scripts are also created to parse the text-based .net, .place, and .route files generated by VTR. The custom script 1 in Fig. 44 creates a dictionary to store the detailed info of each net, while the script 2 creates that of each path. For the activity factors, I use 0.2 for all FPGA inputs. The activity factors of internal nets are then automatically generated by running ACE 2.0.

4.3.5 Energy Reduction Results

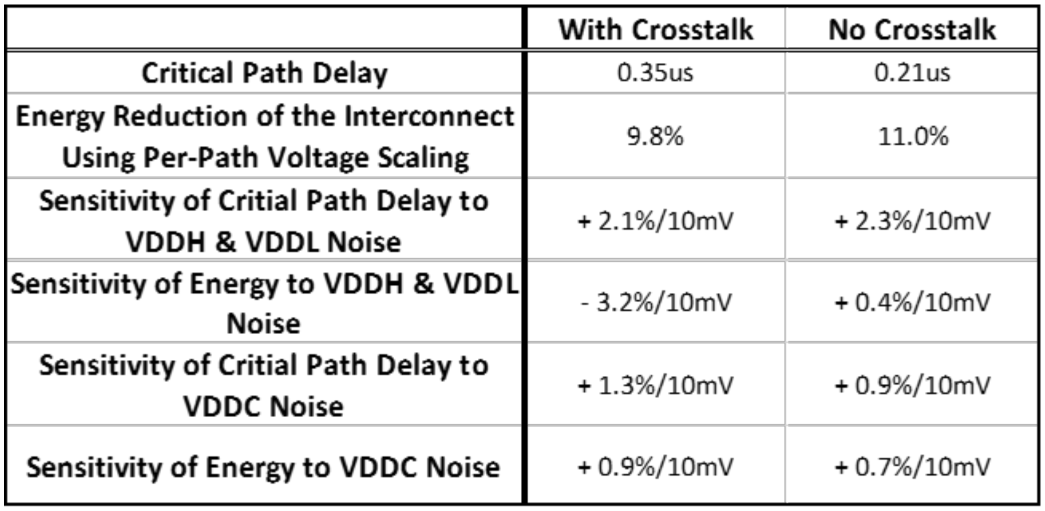
In this research, I swept V_{DDH} from 0.45V to 0.6V and V_{DDL} from 0.15V lower than V_{DDH} to V_{DDH}, to find the supply voltage values leading to the lowest energy of FPGAs implementing MCNC benchmarks. The result shows the higher V_{DDH}, the more energy savings we can achieve from using per-path voltage scaling. Fig. 45 shows the overall energy reduction of the interconnect when implementing five largest MCNC benchmarks at different voltage values. The details of the benchmarks are also shown. The FPGA architecture chosen has 8-LUT CLBs with 4-inputs of each LUT and switch boxes at each intersection of horizontal and vertical channels.

As shown in Fig. 45, when V_{DDH} is set to 0.6V, the maximum energy saving of the interconnect is achieved at V_{DDL} equals to 0.5V, which is 0.1V lower than V_{DDH}. Theoretically, keep reducing V_{DDL} always results in higher energy reduction. However, lowering V_{DDL} leads to higher portion of leakage energy. Also, too low V_{DDL} makes some non-critical paths become the new critical path, and increase the overall FPGA delay and energy. For these reasons, an optimal V_{DDL} value is observed in terms of energy reduction. In Fig. 45, an average energy saving of 20.1% is archived when the LDO overhead does not included. As discussed earlier, a LDO is used to estimate the voltage regulator overhead of generating V_{DDL}. The energy efficiency of the LDO approximately equals to the output voltage divided by the input voltage, which is $\frac{V_{DDL}}{V_{DDH}}$ in this case. If included LDO overhead, the average energy saving drops to 10.1%.



Benchmarks	Benchmark Characterization		Energy Reduction of Using Per-Path Voltage Scaling				
	LUT Count	I/O Count	VDDC-VDD (V)	Including Voltage Regulator Overhead?	VDDH=0.6 VDDL=0.55	VDDH=0.6 VDDL=0.5	VDDH=0.6 VDDL=0.45
alu4	1522	22	0.2	No	14.0%	20.8%	17.5%
				Yes	6.8%	10.3%	8.6%
			0.4	No	13.2%	21.4%	17.8%
				Yes	5.8%	9.7%	8.5%
			0.7	No	13.9%	23.1%	15.4%
				Yes	6.6%	11.5%	8.3%
apex2	1878	41	0.2	No	14.0%	22.3%	21.0%
				Yes	6.8%	10.8%	10.0%
			0.4	No	13.2%	21.9%	21.1%
				Yes	5.6%	9.6%	9.6%
			0.7	No	13.8%	21.7%	16.4%
				Yes	6.4%	10.6%	8.5%
apex4	1262	28	0.2	No	12.6%	16.5%	13.2%
				Yes	6.0%	7.9%	6.4%
			0.4	No	11.9%	17.0%	12.4%
				Yes	5.1%	7.6%	5.9%
			0.7	No	11.9%	15.9%	9.6%
				Yes	5.5%	7.7%	5.0%
des	1591	501	0.2	No	14.1%	22.6%	19.7%
				Yes	7.1%	11.5%	10.0%
			0.4	No	13.7%	22.9%	21.0%
				Yes	6.3%	11.1%	10.5%
			0.7	No	14.5%	24.3%	21.4%
				Yes	7.3%	12.8%	11.8%
ex5p	1064	71	0.2	No	11.3%	14.2%	7.1%
				Yes	5.4%	6.9%	3.4%
			0.4	No	11.4%	14.7%	6.3%
				Yes	4.9%	13.3%	3.0%
			0.7	No	12.3%	15.6%	4.6%
				Yes	5.7%	7.7%	2.3%
Average	1463	133	**0.2**	No	**13.2%**	**19.3%**	**15.7%**
				Yes	**6.4%**	**9.5%**	**7.7%**
			0.4	No	**12.7%**	**19.6%**	**15.7%**
				Yes	**5.6%**	**10.2%**	**7.5%**
			0.7	No	**13.3%**	**20.1%**	**13.5%**
				Yes	**6.3%**	**10.1%**	**7.2%**

Figure 45: The energy reductions of the low-swing interconnect implementing MCNC benchmarks using per-path voltage scaling @ 0.6V in 130nm CMOS



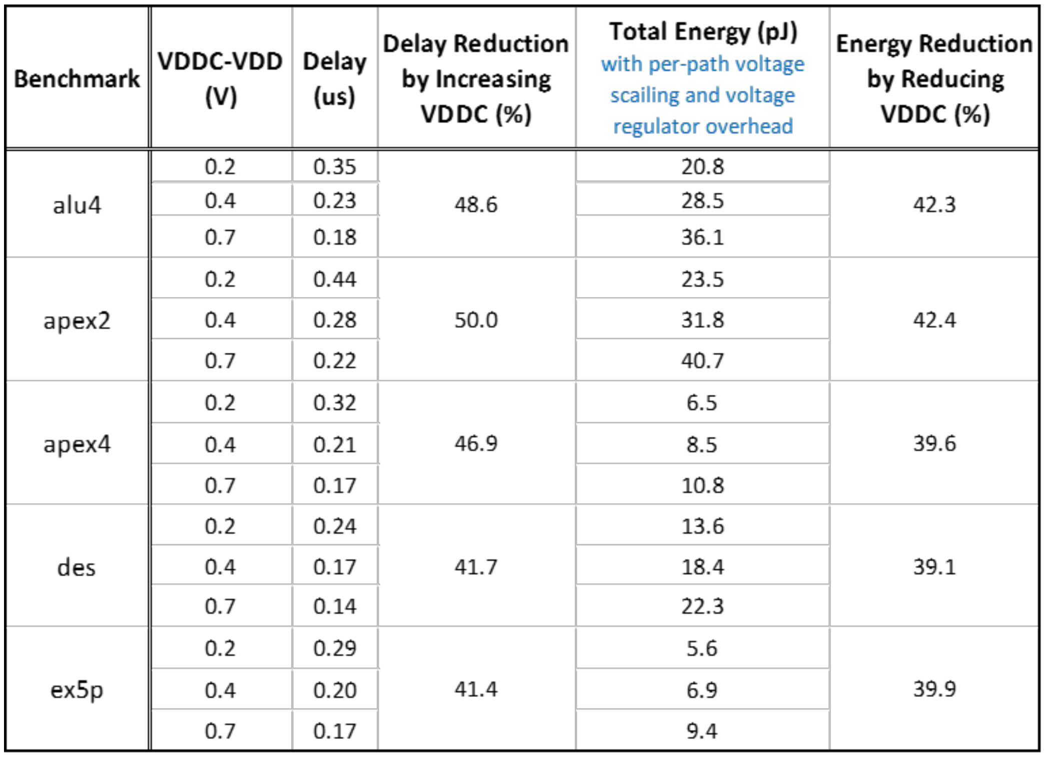
	With Crosstalk	No Crosstalk
Critical Path Delay	0.35us	0.21us
Energy Reduction of the Interconnect Using Per-Path Voltage Scaling	9.8%	11.0%
Sensitivity of Critial Path Delay to VDDH & VDDL Noise	+ 2.1%/10mV	+ 2.3%/10mV
Sensitivity of Energy to VDDH & VDDL Noise	- 3.2%/10mV	+ 0.4%/10mV
Sensitivity of Critial Path Delay to VDDC Noise	+ 1.3%/10mV	+ 0.9%/10mV
Sensitivity of Energy to VDDC Noise	+ 0.9%/10mV	+ 0.7%/10mV

Figure 46: The affects of cross-talk and power noise on the energy reduction of interconnect implementing alu4 @ 0.6V in 130nm CMOS

So far, all the energy reduction numbers are estimated based on an activity factor of 0.2 of all the inputs. However, this assumption cannot cover all the cases. Thus, I ran simulations to show how does the results change while sweeping the activity factor from 0.1 to 1.0. Although increasing the toggle rates increases the FPGA energy dramatically, the percentage of energy reduction doesn't change much if the V_{DDC} value is a constant. In Fig. 46, the influences from cross-talk and voltage noise are also examined. As a result, although the critical path delay and the interconnect energy are vulnerable to cross-talk, they are not very sensitive to voltage noise. All the data shown in this figure is simulated at V_{DDH} = 0.6V and V_{DDL} = 0.5V.

Finally, the simulated maximum and minimum delay and energy of the interconnect can be adjusted to by using DVS are shown in Fig. 47. The V_{DDC} value is swept from V_{DDH} to 0.7V higher than V_{DDH} for each benchmark. However, when V_{DDC} is less than 0.2V higher than V_{DDH}, the swing of the signals in the interconnect will reduce to a level that cannot be detected by the level shifters in the simulation. This leads to functionality failures of the FPGAs. DVS allows approximately 45%

delay reduction or 40% energy reduction when needed in addition to per-path voltage scaling. Take alu4 as an example, the critical path delay can be adjusted from 0.35μs to 0.18μs and energy per operation from 36.1pJ to 20.8pJ at 0.6V.



Benchmark	VDDC-VDD (V)	Delay (us)	Delay Reduction by Increasing VDDC (%)	Total Energy (pJ) with per-path voltage scailing and voltage regulator overhead	Energy Reduction by Reducing VDDC (%)
alu4	0.2	0.35	48.6	20.8	42.3
	0.4	0.23		28.5	
	0.7	0.18		36.1	
apex2	0.2	0.44	50.0	23.5	42.4
	0.4	0.28		31.8	
	0.7	0.22		40.7	
apex4	0.2	0.32	46.9	6.5	39.6
	0.4	0.21		8.5	
	0.7	0.17		10.8	
des	0.2	0.24	41.7	13.6	39.1
	0.4	0.17		18.4	
	0.7	0.14		22.3	
ex5p	0.2	0.29	41.4	5.6	39.9
	0.4	0.20		6.9	
	0.7	0.17		9.4	

Figure 47: The adjustable range of delay and energy of the interconnect implementing MCNC benchmarks when using DVS @ 0.6V in 130nm CMOS

4.4 Power-Gating

4.4.1 Architecture Overview

Power-Gating is an effective technique to reduce leakage energy in idle circuits. FPGA researchers traditionally use this technique to reduce the leakage energy of the buffers in switch boxes [16] [15]. However, this part of energy is naturally zero for low-swing interconnect since all buffers have already been removed. As leakage energy becomes the dominant part when the supply voltage is scaled down to near/sub-threshold, the leakage in the configuration bit-cells can no longer be ignored [6]. The interconnect energy breakdown of the low-swing interconnect implementing MCNC benchmarks is shown in Fig. 48. About 50% of the total FPGA energy at 0.6V is contributed by the leakage energy in idle circuits, where the majority is contributed by configuration bit-cells. The numbers shown in the figure is based on the assumption of an activity factor of 0.2, which is an empirical number. Increasing the activity factor increases the portion of dynamic energy. In the extreme case, where the activity factor equals to 1, the percentage of leakage energy drops to 26.7%. However, this is not realistic in real applications.

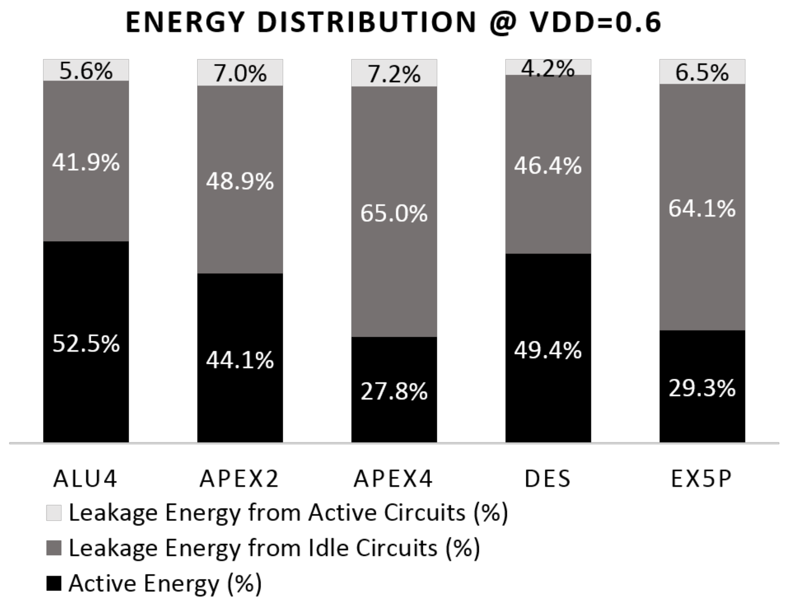


Figure 48: Energy breakdown of the low-swing FPGA interconnect implementing MCNC benchmarks @ 0.6V in 130nm CMOS

In this research, power-gating is applied to the bit-cells in switch boxes. The simplest bit-cells are 5T SRAM cells used to turn-on or turn-off the switches in the interconnect or to store LUT values in CLBs. To use power-gating, PMOS headers need to be inserted between the voltage supply and bit-cells. When the corresponding switch boxes are in idle mode, turning-off the PMOS headers reduces leakage current in the bit-cells due to stack effect. By using high-V_T transistors, the leakage could be even smaller. However, the size of the headers need to be properly selected. Too large headers result in higher leakage, while smaller headers lead to degraded speed and robustness due to IR drop. As shown in Fig. 49, there are different architectures of power-gating. The coarse-grain power-gating only need one header in each switch box. All bit-cells in a switch box can only be turned on or off at once. The fine-grain architecture, on the other hand, has one header for each bit-cell to control them individually. Although the fine-grain power-gating can be applied to more switch boxes, it has higher routing and area overhead than the coarse-grain design. This trade-off will be discussed later. The control signal of a header is provided by another bit-cell instead of scan-chains, because of the large total number of headers. Thus, applying power-gating to each bit-cell is not realistic, since the same amount of bit-cells is required to performing power-gating.

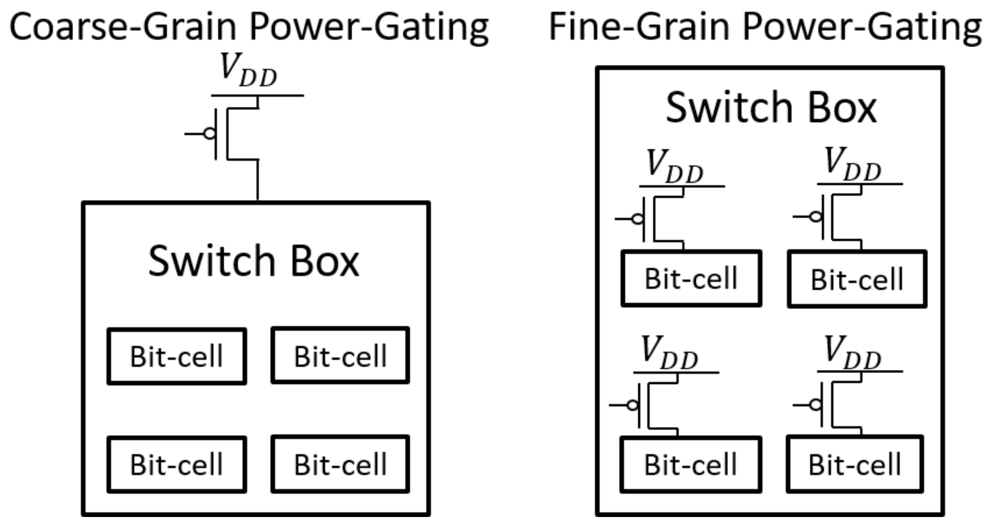
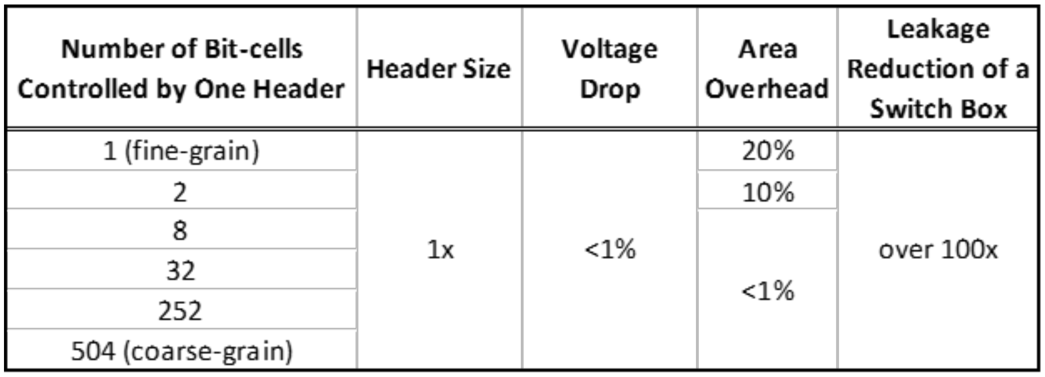


Figure 49: The coarse-grain power-gating and fine-grain power-gating architectures

4.4.2 Header Design for Power-Gating



Number of Bit-cells Controlled by One Header	Header Size	Voltage Drop	Area Overhead	Leakage Reduction of a Switch Box
1 (fine-grain)	1x	<1%	20%	over 100x
2			10%	
8			<1%	
32				
252				
504 (coarse-grain)				

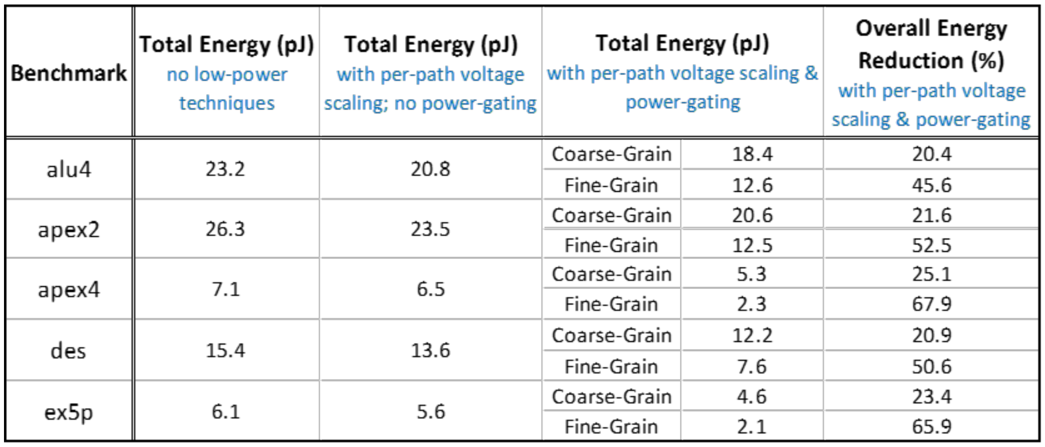
Figure 50: Characterization of the headers for power-gating

In Fig. 50, the characterization of the headers is shown. Since no MCNC benchmarks requires more than 84-tracks to route, all simulations are setup for an 84-track switch box. which has 504 leaking bit-cells. The voltage drop across the header can be ignored. In 100-point Monte Carlo simulations at different voltage, the voltage drop is less than 1% even thought a minimum-sized header is used to power-gate all the 504 bit-cells. If using the minimum-sized header, the area overhead of the header is also less than 1% compared to the area of the switch box. Using fine-grain power-gating leads to higher area overhead of up to 20%, as each bit-cell needs a header. This area overhead is confirmed by making a custom layout of s 84-track low-swing switch box. Finally, the leakage energy of the switch boxes can be reduced by over 100x after using power-gating according to SPICE simulations.

4.4.3 Energy Reduction Results

Fig. 51 shows the overall energy reduction of the low-swing interconnect using both per-path voltage scaling and power-gating. For each benchmark, the energy of the interconnect before and after using power-gating and per-path voltage scaling is listed, and the overall energy reduction is calculated by comparing the two numbers.

The total energy in the figure comprises both leakage energy and dynamic energy, which is obtained by summing up SPICE simulation results of each path implementing the benchmarks.



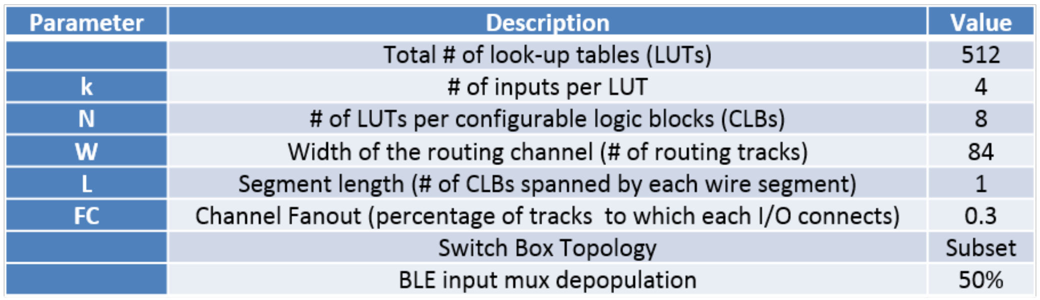
Benchmark	Total Energy (pJ) no low-power techniques	Total Energy (pJ) with per-path voltage scaling; no power-gating	Total Energy (pJ) with per-path voltage scaling & power-gating		Overall Energy Reduction (%) with per-path voltage scaling & power-gating
alu4	23.2	20.8	Coarse-Grain	18.4	20.4
			Fine-Grain	12.6	45.6
apex2	26.3	23.5	Coarse-Grain	20.6	21.6
			Fine-Grain	12.5	52.5
apex4	7.1	6.5	Coarse-Grain	5.3	25.1
			Fine-Grain	2.3	67.9
des	15.4	13.6	Coarse-Grain	12.2	20.9
			Fine-Grain	7.6	50.6
ex5p	6.1	5.6	Coarse-Grain	4.6	23.4
			Fine-Grain	2.1	65.9

Figure 51: The energy reduction of the low-swing interconnect after using power-gating and per-path voltage scaling @ 0.6V in 130nm CMOS

The power-gating architecture has a large impact on the overall energy reduction. In the figure, six different strategies of power-gating are considered, simulated, and used to estimate energy reduction. These strategies are based on the number of bit-cells controlled by each header in a switch box. For example, an 84-track switch box, which can route all the MCNC benchmarks, has 504 bit-cells in total. Since the coarse-grain power-gating only has one header in a switch box, this header need to control all 504 bit-cells. That's why "504" is placed in the slot. The header in a coarse-grain power-gating can be turned-off only when the entire switch box is in idle mode, which is very rare. Take alu4 as example, although there are 307,499 bit-cells in the FPGA are not used, only 81,413 of them can be power-gated, since the active bit-cells distributed in many switch boxes. Using more headers in a switch box theoretically can solve this problem. However, as each header requires an additional bit-cell to control it, using too much headers makes the benefit of power-gating cannot compensate the energy overhead of the introduced bit-cells. The number of bit-cells can be power-gated highly depends on the P&R algorithm. In Fig. 51, the energy of FPGA interconnect implementing benchmarks using no low-power techniques, using only per-path voltage scaling, and additionally using power-gating are given. As a result, the average energy reduction after using all low-power techniques is 22.3% when coarse-grain power-gating is used, relative to a near/sub=threshold FPGA without using per-path voltage scaling and power-gating. When using fine-grain power-gating, the maximum reduction is 56.5%.

4.5 A Custom FPGA Chip

4.5.1 Chip Description



Parameter	Description	Value
	Total # of look-up tables (LUTs)	512
k	# of inputs per LUT	4
N	# of LUTs per configurable logic blocks (CLBs)	8
W	Width of the routing channel (# of routing tracks)	84
L	Segment length (# of CLBs spanned by each wire segment)	1
FC	Channel Fanout (percentage of tracks to which each I/O connects)	0.3
	Switch Box Topology	Subset
	BLE input mux depopulation	50%

Figure 52: Architectural and Circuit-Level Parameters for taped-out FPGA chip

To verify the low-power design ideas and the functionality of a custom tool flow that will be discussed in Chapter 5, Dr. Ayorinde and I built and taped-out a chip of FPGA core fabric with full functionality. The chip is designed for near/sub-threshold operation and has the following low-power features:

- Clustered configurable logic blocks with eight look-up-tables and flip-flops
- Low-swing interconnect optimized for energy efficiency
- Area-optimized folded-subset switch boxes in the interconnect

- Enabling per-path voltage scaling
- Low-power multiplexer-based intra-CLB routing
- Coarse-grain power-gating for both CLBs and switch boxes

The chip also includes additional circuitry for configuration and testing. This includes:

- Scan-chains for configuring look-up-tables, interconnect, per-path voltage scaling, and power-gating
- Scan-chains for controlling a large percentage of the inputs/outputs to the FPGA core fabric
- Direct connections to virtual rails of CLBs and switch boxes to observe power-gating efficiency
- Registers at the inputs and outputs of the FPGA for delay measurements

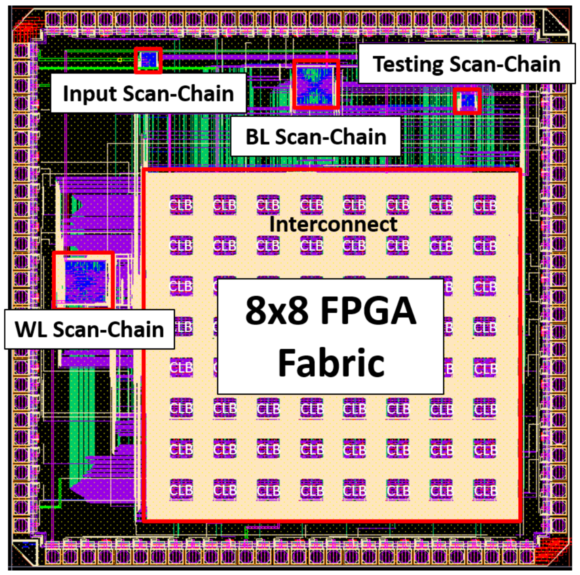
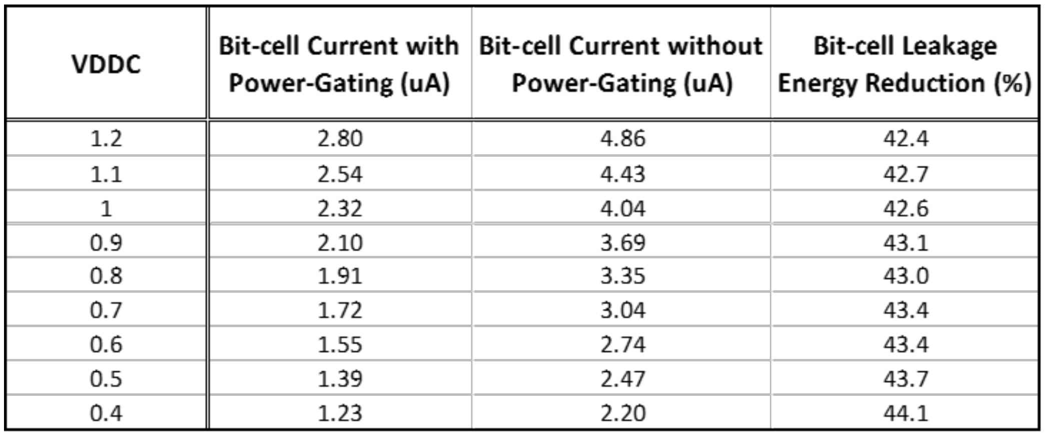


Figure 53: The annotated layout of the of the custom FPGA chip

Fig. 52 displays the important parameters of the FPGA chip. We chose 4-input look-up-tables and a clustering (N) of 8 to build the CLBs due to Dr. Ayorinde's research results [43] and the industry standard. The total number of look-up-tables on the chip is 512, which is large enough to implement meaningful applications. A channel width (W) of 84 allows our chip to route significant benchmarks while using the subset switch box topology. To our knowledge, this is the first FPGA chip has been taped out which is targeted for near/sub-threshold operation and also includes as many look-up-tables and routing resources.



VDDC	Bit-cell Current with Power-Gating (uA)	Bit-cell Current without Power-Gating (uA)	Bit-cell Leakage Energy Reduction (%)
1.2	2.80	4.86	42.4
1.1	2.54	4.43	42.7
1	2.32	4.04	42.6
0.9	2.10	3.69	43.1
0.8	1.91	3.35	43.0
0.7	1.72	3.04	43.4
0.6	1.55	2.74	43.4
0.5	1.39	2.47	43.7
0.4	1.23	2.20	44.1

Figure 54: The measurement results of leakage energy reduction of the custom FPGA chip before and after using power-gating

An annotated screen shot of the layout of the chip is shown in Fig. 53. The 8x8 CLB array is on occupying the majority of area is the FPGA core fabric. A word-line (WL) scan-chain and a bit-line (BL) scan-chain are included to configure the chip WL by WL. Since the number of IOs of the FPGA core fabric (192) is higher than the number of IOs of the chip (112), two additional scan-chains are needed to access and observe a portion of IOs of the FPGA core fabric. There is also a portion of IOs are directly connected to the chip IOs. Each chip IO includes a register enable delay measurement.

4.5.2 Measurement Results

In Fig. 54, the leakage current of the chip at varying V_{DDC} is shown. In this task, no application is mapped to the FPGA, and the values in all bit-cells are reset to zero. At each V_{DDC}, the total leakage current of the bit-cells is measured when the power-gating headers are turned-on and turned-off. As a result, using power-gating reduces the leakage energy of the bit-cells by about 90%. The value of V_{DDC} doesn't affect the percentage of energy reduction much. The size of the power-gating header in this chip is chosen as 30x. Although 1x is good enough shown in simulation, a larger header is used for a guard-band against variation and noise. The area overhead of the header can be ignored in this chip, since the header is placed in the blank area between two CLBs.

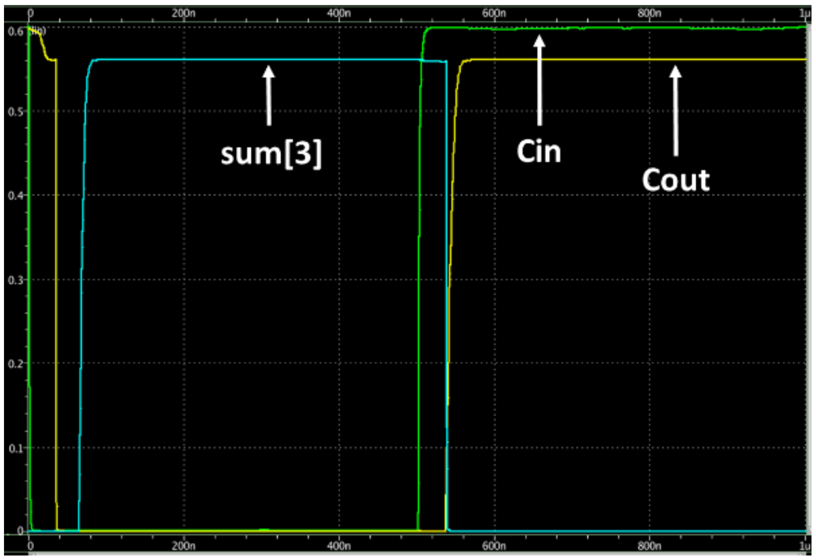
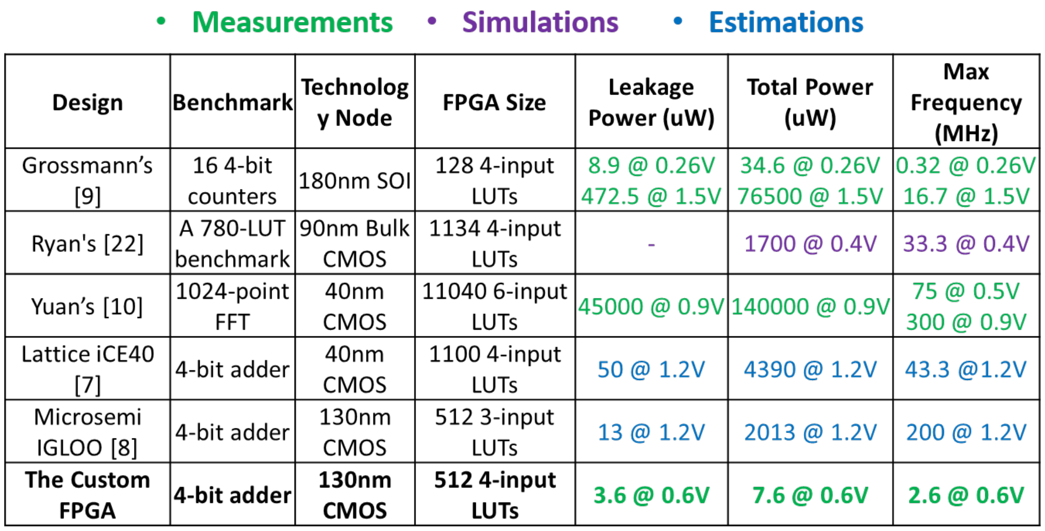


Figure 55: Simulated waveforms of a 4bit-adder

Besides leakage reduction, the functionality of the custom chip and tool flow (introduced in Chapter 5) is also verified. Fig. 55 shows the simulated waveform of the chip implementing a 4-bit adder. The chip is simulated using HSPICE with an initial condition stream of the bit-cells generated by the custom tool flow. Assume the inputs of the 4-bit adder are denoted as 'a' and 'b', the output is denoted as 'sum', and the carry-in and carry-out are denoted as 'Cin' and 'Cout', respectively. In the figure, the inputs are set to a[3]=a[2]=a[1]=a[0]=0 and b[3]=b[2]=b[1]=b[0]=1. By toggling 'Cin' from 0 to 1, the 'sum[3]' is expected to flip from 1 to 0 and 'Cout' from 0 to 1. The waveform shown in the figure exactly matches the expected results.



• Measurements • Simulations • Estimations

Design	Benchmark	Technology Node	FPGA Size	Leakage Power (uW)	Total Power (uW)	Max Frequency (MHz)
Grossmann's [9]	16 4-bit counters	180nm SOI	128 4-input LUTs	8.9 @ 0.26V 472.5 @ 1.5V	34.6 @ 0.26V 76500 @ 1.5V	0.32 @ 0.26V 16.7 @ 1.5V
Ryan's [22]	A 780-LUT benchmark	90nm Bulk CMOS	1134 4-input LUTs	-	1700 @ 0.4V	33.3 @ 0.4V
Yuan's [10]	1024-point FFT	40nm CMOS	11040 6-input LUTs	45000 @ 0.9V	140000 @ 0.9V	75 @ 0.5V 300 @ 0.9V
Lattice iCE40 [7]	4-bit adder	40nm CMOS	1100 4-input LUTs	50 @ 1.2V	4390 @ 1.2V	43.3 @1.2V
Microsemi IGLOO [8]	4-bit adder	130nm CMOS	512 3-input LUTs	13 @ 1.2V	2013 @ 1.2V	200 @ 1.2V
The Custom FPGA	**4-bit adder**	**130nm CMOS**	**512 4-input LUTs**	**3.6 @ 0.6V**	**7.6 @ 0.6V**	**2.6 @ 0.6V**

Figure 56: Comparisons of the custom FPGA and existing low-power FPGAs

Benchmarking the existing FPGAs is extremely important for providing a context in which to look at our work in the development of FPGA fabrics, which will allow us to target our designs for the state-of-the-art. Thus, a 4-bit adder is mapped to the custom FPGA chip for measurement. In Fig. 56, its measured delay and power are compared to three academic works ([9], [10], and [22]) and two commercial products (Lattice iCE40 UltraLite [7] and Microsemi IGLOO [8]). Since the benchmarks used in the academic works are different from us, we cannot do direct comparisons with them. However, the leakage power of the custom FPGA is much smaller then the existing works. Comparing to [9], our FPGA is 2.5x lower static power even though our FPGA is 4x larger than [9] and measured at a higher supply voltage. The rest of the academic works consume even more leakage power than [9]. The commercial products provide more flexibility on applications. Thus, we mapped the 4-bit adder to iCE40 and IGLOO using their released tools for fair comparisons. As a result, the custom FPGA consumes 578x lower power at a 17x lower speed than iCE40 UltraLite, and consumes 277x lower power at a 77x lower speed than IGLOO. The energy of IGLOO is 3.4x higher than the custom FPGA.

4.6 Future Research

In this chapter, a custom tool is developed to assign V_{DDH} and V_{DDL} to each path towards minimize energy. However, the tool is currently using a simple brute-force algorithm, which makes the tool having long run-time for large benchmarks. A better algorithm can improve the tool speed significantly. Furthermore, although the tool can determine which paths need to be connected to V_{DDH} or V_{DDL}, it can't generate configuration bit-stream to enable per-path voltage scaling and power-gating on the chip. In addition, as discussed in 4.4.3, the energy saving of using power-gating highly depends on the P&R tool, which is expected to be power-gating-aware to group all idle bit-cells into several switch boxes. Also, the leakage power saving needs to be revisit at modern technology nodes such as 16nm FinFET, where the percentage of leakage energy is reduced.

Another topic of future research is about voltage regulator. In this research, the energy overhead of the voltage regulators is estimated by theoretical calculation of the equations. However, this is potentially not accurate. Since delay and energy of the FPGA is very sensitive to supply voltage, a tiny inaccuracy in voltage regulator estimation can result in different conclusions on the optimal V_{DDH} and V_{DDL} values. A more realistic model of the voltage regulators is needed.

5 Low-Power FPGA Evaluation Platform

5.1 Motivation

5.1.1 Low-Power Application Suite

It is hard to evaluate a FPGA since it can run arbitrary applications, and the performance and energy of a FPGA varies when implementing different applications. To evaluate a FPGA comprehensively, we need an application suite that has good application diversity, size diversity, and hardware resource usage diversity. Since the goal of this research is about building a low-power FPGA, the requirement on this application suite becomes finding a group of low-power applications in different fields that are suitable to be implemented by FPGAs. At mean time, these applications are expected to use different combination of LUTs, FFs, interconnect resources, and hard IPs.

5.1.2 Application Synthesis

Before using FPGAs, the CAD tools need to perform multiple preparation steps, including 1)translating applications written in Verilog into gate-level descriptions 2)mapping the gates to LUTs, which then wrapped into CLBs or other hardware resources available on the target FPGAs 3)placing the CLBs at certain locations on FPGAs, then connecting them through global interconnect 4)optimizing P&R to meet design constraints. The first and second steps are usually done by benchmark synthesis tools. All FPGA companies have their own synthesis tools, such as Xilinx ISE/Vivado and Altera Quartus. Although these tools are powerful, they are designed for supporting their own FPGAs. Compared to commercial FPGAs that much care about throughput to compete with processors, ultra-low-energy FPGAs has less requirements on speed but highly constrained by power budget. While advanced place and route algorithms play an important role in reducing FPGA power, circuit level improvement also helps. Low-power circuit design techniques such as custom CLB, low-swing interconnect, new architectures, power-gating, and per-path voltage scaling can be applied to low-power FPGA designs. However, commercial synthesis tools do not recognize the architecture of custom designed FPGAs, and cannot map logic correctly to these FPGAs. Thus, a general-purpose synthesis tool is needed. Since the majority of applications are written in Verilog these days, there is no strong need to work on the conversion of C/C++ and VHDL, which are out of scope of this dissertation.

5.1.3 Fast Power Estimation of the Custom FPGA

FPGA fast evaluation is an important topic in FPGA CAD flow developments. There are mainly three reasons designers need FPGA fast evaluations. 1)The near/sub-threshold analysis of energy consumption and timing closure of custom FPGAs require a dramatically increased level of SPICE-level simulation precision. Such highly-accurate simulations can take days even weeks to complete for a full FPGA fabric. FPGA circuit designers need an alternative method to fast evaluate their designs on system level. 2)Designers of custom FPGAs need quick guides towards the optimal architecture parameters and on-chip IP type/size/number (memory blocks, multipliers, adders, etc.) to build their chips. 3)When SoC designers want to implement some on-chip blocks with embedded FPGA slices, they need to quickly know the overhead and trade-offs without dealing with transistor level issues. Besides evaluation speed, the accuracy of evaluation is also important for FPGA circuit designers. Since circuit level improvements usually has smaller impact on the system compared to improvements on architecture or P&R algorithms, inaccurate simulations can hide the impact of circuit innovations. Thus, a tool enabling fast speed and energy estimation of the custom low-power FPGA is required. The only inputs of the tool will be the applications in Verilog and an FPGA architecture description file.

5.2 Prior Art

5.2.1 Low-Power Application Suite

Application suites play a significant role in evaluating and comparing FPGA circuits, architectures, and CAD tools. Unfortunately, existing commonly used application suits are either too large for low-power FPGAs or too simple to fully utilize low-power FPGA resources. For example, none of the designs in the MCNC suite [25] uses hard IPs, which are deployed on all modern FPGAs, and a big portion of the designs are purely combinational. Furthermore, the MCNC suite is written in Berkeley Logic Interchange Format (BLIF) that is very inconvenient to use, since it needs to be modified every time designers make changes to FPGA architecture even though the change is very tiny. On the other hand, the Titan23 [44] suite includes designs with high usage of hard IPs. The number of those IP blocks in each Titan23 benchmark reaches the range of several thousand, which is comparable with the available hard IPs on the latest Xilinx Virtex 7 series FPGAs. However, these applications are too big to fit into low-power FPGAs. For example, the size of commercial low-power FPGA Lattice iCE40 [7] and Microsemi IGLOO [8] ranges from 100 LUTs to 7680 LUTs, while the smallest Titan23 application has 90,779 LUTs. Without applications with proper size and the utilization of hard IPs, we cannot effectively and comprehensively evaluate low-power FPGA circuits and CAD flows.

5.2.2 Application Synthesis

The existing Verilog synthesis tools are also not suitable for low-power FPGAs. The most common used synthesis tools are Xilinx ISE/Vivado, Altera Quartus, and Synopsys Synplify. These tools support all kinds of application formats and create configuration bit-streams very fast in minutes. However, the commercial tools only support the existing FPGAs on the market. Researchers cannot verify their own FPGA circuit or architecture designs using commercial tools. To solve this problem, multiple open source synthesis tools are build, and the most widely used one is ODIN II [45] integrated in Verilog-to-Routing (VTR) [24]. The ODIN II can synthesis applications using the hardware resources on a user-defined FPGA architecture. However, ODIN II only supports limited Verilog syntax. For example, it does not support nested modules, high-impedance 'z', shift operations, 2D arrays, gate-level descriptions, unused modules, and so on. These functions play an irreplaceable position and are very commonly seen in modern RTL designs. Since FPGA circuit designers do not always write RTL benchmarks on these own, but using RTL designs from a third party, they have to revise the RTL before evaluating their custom FPGAs using ODIN II. This process is painful, time-consuming, and unnecessary. Thus, a new synthesis flow supports both custom FPGA architecture and all valid Verilog formats is required.

5.2.3 Fast Power Estimation of the Custom FPGA

The exiting tools cannot meet the requirements of estimating the energy of the custom FPGA as well. Although the commercial tools such as Xilinx ISE/Vivado and Altera Quartus provides fast speed and power estimations of their own products when implementing any applications, they can't evaluate FPGAs either have a different architecture or run at a different voltage. To solve this, VTR [24] is designed to be able to estimate any user-defined FPGA at any voltage. However, VTR does not work well for the custom low-power FPGA for multiple reasons. 1)While designers can define the architecture of CLBs, the interconnect architecture is fixed in VTR. Since interconnect dominate the overall FPGA energy, designers usually apply multiple low-power techniques to the interconnect. For example, I use low-swing architecture, per-path voltage scaling, power-gating. The benefits of these low-power designs can't be counted by VTR flow. 2)VTR estimates delay, power, and area based on user-provided values of sub-circuits and hard IPs in FPGA architecture files. For example, users need to provide the delay from each input of a LUT to its output. These values are obtained from circuit simulations or chip measurements. However, all such values in VTR are based on high performance FPGAs, which is far different from near/sub-threshold operations. 3)The leakage power estimated by VTR does not take consideration of power-gating. In addition to VTR, FPGA-SPICE is another FPGA fast evaluation tool [49]. Although it achieves 14x speed-up compared to simulating full-chips in SPICE, it also does not support low-swing interconnect and per-path voltage scaling.

5.2.4 FPGA Generation and Configuration

To enable low-power FPGA circuit development and simulation, Dr. Ayorinde build a FPGA Generation and Configuration (FGC) tool as shown in Fig. 57 [43]. While VTR can only generating virtual mapping and routing information of the given application and FPGA architecture, FGC add more functions to it, including FPGA schematic generation, circuit initial condition generation for chip simulation, and configuration bit-stream generation for chip measurement. However, FGC still remains the problem of application synthesis and FPGA energy estimation. In this chapter, researches of adding these capabilities to FGC will be discussed.

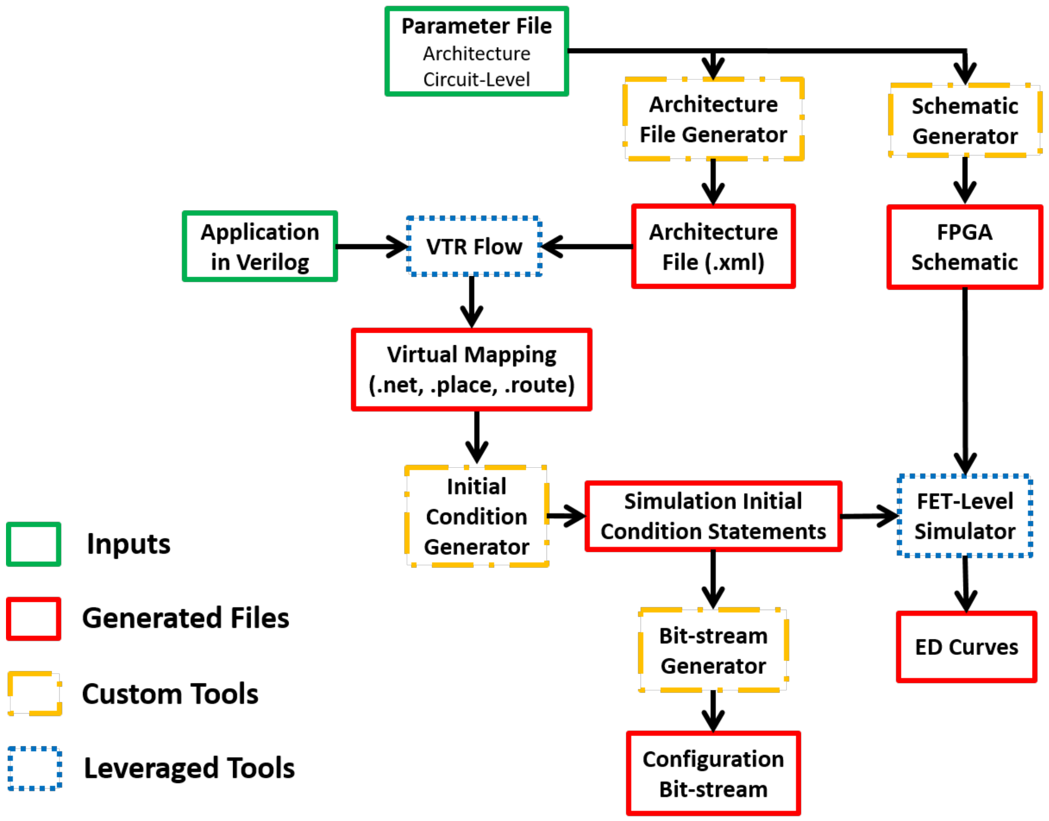


Figure 57: The flow chart of FGC

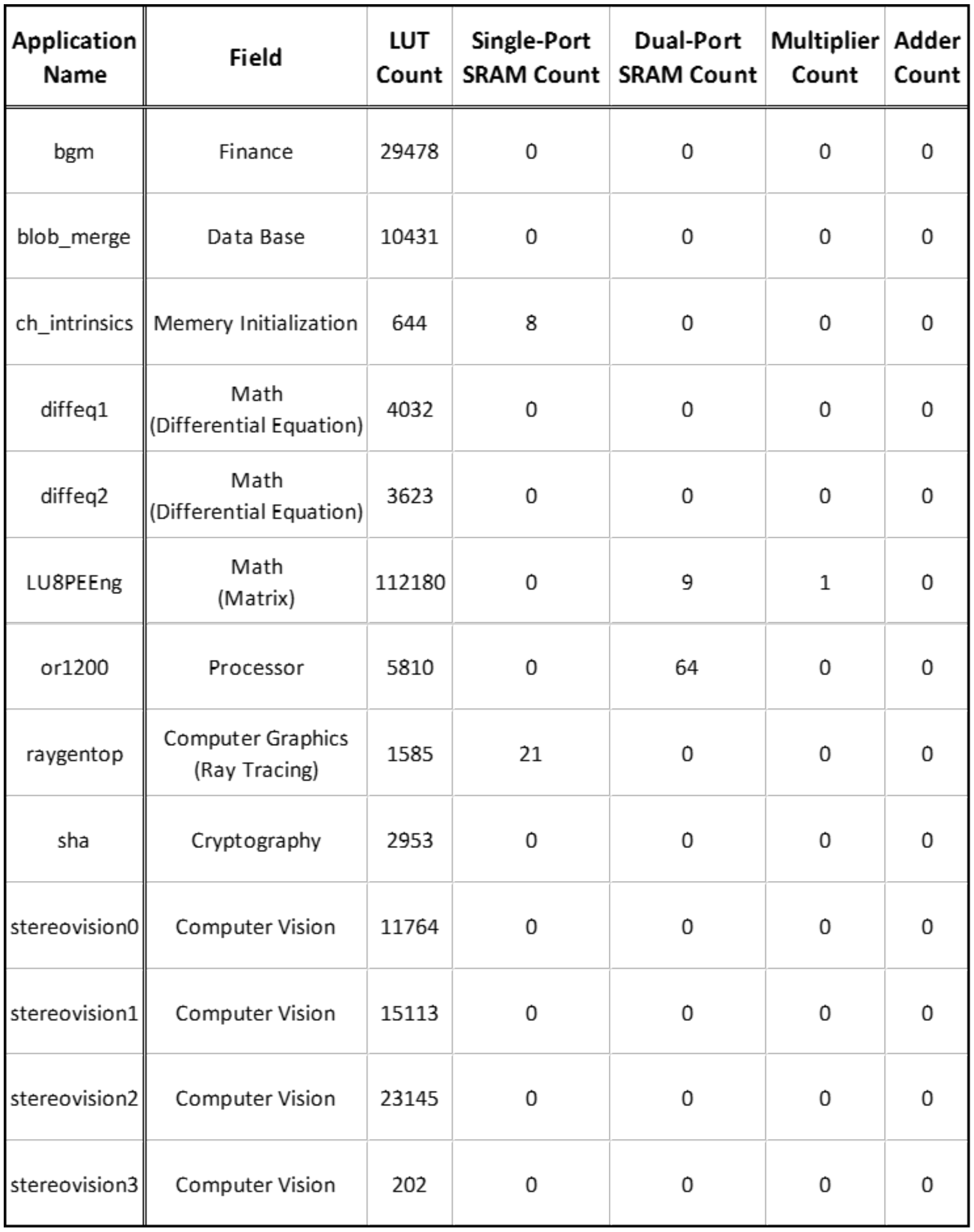
5.3 Application Suite

The members of the low-power application suite are expected to include meet the following requirements.

- use less than 100k LUTs
- including both combinational logic and sequential logic
- including the usage of memory blocks, multipliers, and other hard IPs
- from real low-power applications

Based on these requirements, the proposed application suite is built up with selected applications from the VTR suite [24], applications implemented by Lattice iCE40 [7] and Microsemi IGLOO [8], and battery-less wearable sensor applications from UVa [46] [11].

5.3.1 Selected VTR Applications

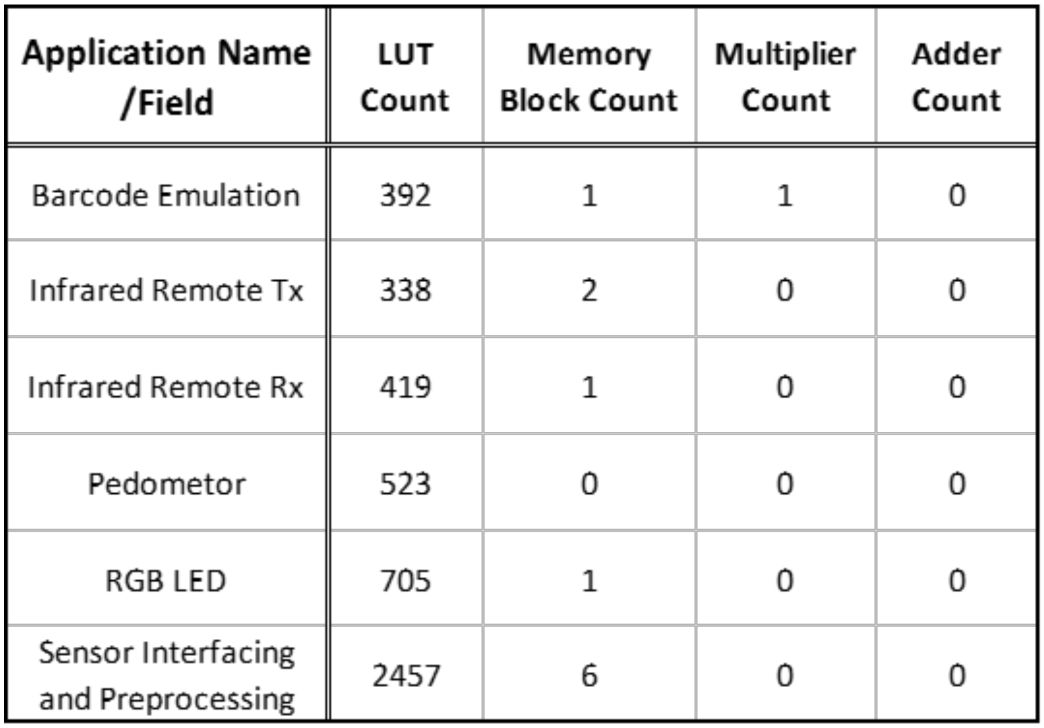


Application Name	Field	LUT Count	Single-Port SRAM Count	Dual-Port SRAM Count	Multiplier Count	Adder Count
bgm	Finance	29478	0	0	0	0
blob_merge	Data Base	10431	0	0	0	0
ch_intrinsics	Memery Initialization	644	8	0	0	0
diffeq1	Math (Differential Equation)	4032	0	0	0	0
diffeq2	Math (Differential Equation)	3623	0	0	0	0
LU8PEEng	Math (Matrix)	112180	0	9	1	0
or1200	Processor	5810	0	64	0	0
raygentop	Computer Graphics (Ray Tracing)	1585	21	0	0	0
sha	Cryptography	2953	0	0	0	0
stereovision0	Computer Vision	11764	0	0	0	0
stereovision1	Computer Vision	15113	0	0	0	0
stereovision2	Computer Vision	23145	0	0	0	0
stereovision3	Computer Vision	202	0	0	0	0

Figure 58: VTR Applications Characterizations

Some of the benchmarks released together with VTR meet the requirements of the expected low-power application suite. The number of 4-input LUTs needed to implement these applications is between 202 to 112K, and on average 17K. In addition, they use varying number of memory blocks (single port or dual port) and multipliers. These applications are too small to evaluate high performance FPGAs, which usually equipped over 1M LUTs and 1K hard IPs. However, while only less than 1% of the resources can be utilized, the majority of the hardware available on low-power FPGAs will be used to implement these applications. This collection also has diversity of the field/industry, include image processing, memory control, math, soft processor, cryptography, computer vision, and so on. The details of the selected VTR benchmarks are shown in the Fig. 58.

5.3.2 Selected Commercial Applications



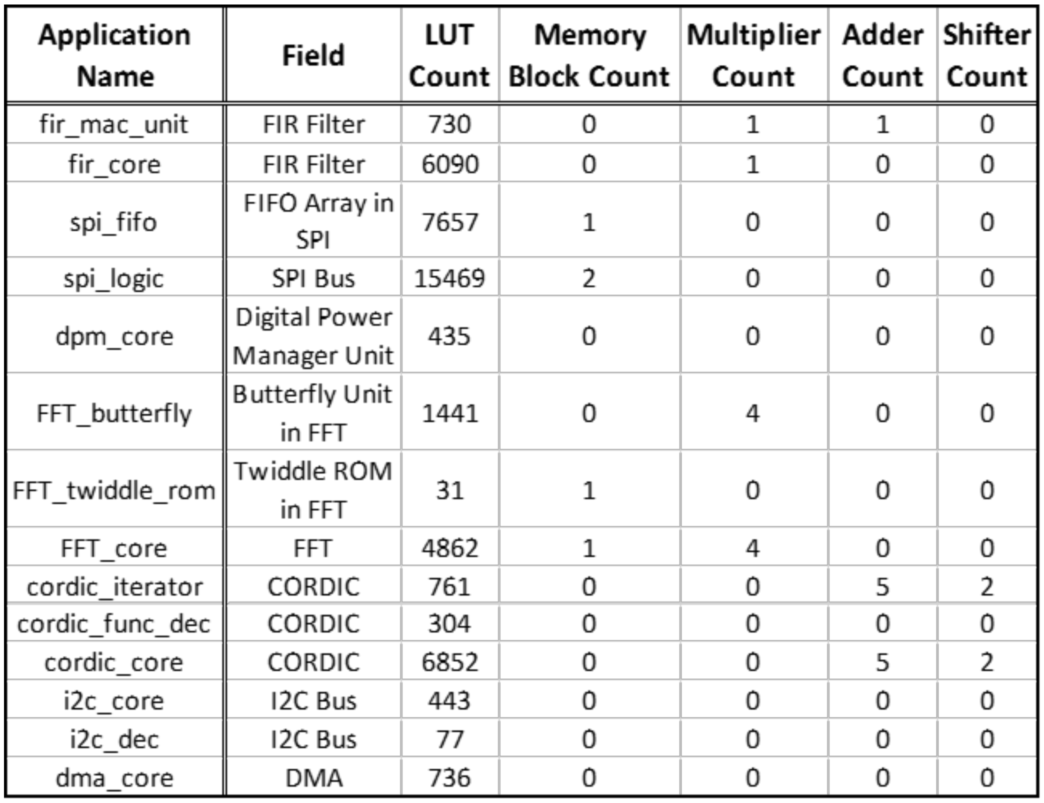
Application Name /Field	LUT Count	Memory Block Count	Multiplier Count	Adder Count
Barcode Emulation	392	1	1	0
Infrared Remote Tx	338	2	0	0
Infrared Remote Rx	419	1	0	0
Pedometor	523	0	0	0
RGB LED	705	1	0	0
Sensor Interfacing and Preprocessing	2457	6	0	0

Figure 59: Commercial Applications Characterizations

Another portion of members of the low-power application suite comes from the usage of commercial FPGAs. The iCE40 and IGLOO have two main categories of usage, including 1) portable miniature systems 2) interfaces. The details of these applications are shown in the Fig. 59. For low-power SoC solutions of portable applications, such as barcode emulator, pedometer, and sensors, the low-power FPGAs are more widely used than high performance FPGAs due to the strict energy budget. In addition, low-power FPGAs can be used to implement interfaces or supplemental units as a separate chip on board, such as the interfaces, IO expander, and remote Rx/Tx. Most of these applications can also be implemented by ASICs, however, the hardware flexibility of FPGAs dramatically reduces the costs of design, error-fixing, and algorithm upgrade. Finally, these applications are small in size (392 - 2457 equivalent 4-input LUTs) and use heterogeneous hard IPs.

5.3.3 Additional Ubiquitous Computing Applications

The last portion of the application suite comes from on-going ultra-low-energy research. As the size of devices shrinks for decades, integrated circuits become more and more faster and lower energy. The entire circuit community agrees that the age of ubiquitous computing is coming. People expect to have sensors everywhere around us, helping to collect data from the environment, analyzing data, and even making decisions for us for straightforward tasks. To archive this, ultra-low-energy computing systems are needed. These devices are expected to have long life-time, local intelligence, and sharing information with each other and higher level of controllers throw wireless communication. Among the existing ubiquitous computing applications, the most promising one is personal health sensors. Although these sensors are currently implemented by ASICs, FPGAs are essential for some of the blocks. For example, for digital signal processing units in the sensors, hardware flexibility is needed to frequently update the quickly changing algorithms without re-spin the chips. In addition, as an interface with other devices in the market, IOs can be implemented by embedded FPGA slices to enable post-silicon revisions of IO type and bandwidth to meet the rapidly changing requirements of customers. FPGA are also good at implementing on-chip interfaces such as SPI and I2C. Other than these applications, embedded FPGA slices can also be used as hardware encryption in the future when personal health data protection becomes important.



Application Name	Field	LUT Count	Memory Block Count	Multiplier Count	Adder Count	Shifter Count
fir_mac_unit	FIR Filter	730	0	1	1	0
fir_core	FIR Filter	6090	0	1	0	0
spi_fifo	FIFO Array in SPI	7657	1	0	0	0
spi_logic	SPI Bus	15469	2	0	0	0
dpm_core	Digital Power Manager Unit	435	0	0	0	0
FFT_butterfly	Butterfly Unit in FFT	1441	0	4	0	0
FFT_twiddle_rom	Twiddle ROM in FFT	31	1	0	0	0
FFT_core	FFT	4862	1	4	0	0
cordic_iterator	CORDIC	761	0	0	5	2
cordic_func_dec	CORDIC	304	0	0	0	0
cordic_core	CORDIC	6852	0	0	5	2
i2c_core	I2C Bus	443	0	0	0	0
i2c_dec	I2C Bus	77	0	0	0	0
dma_core	DMA	736	0	0	0	0

Figure 60: Ubiquitous Computing Applications Characterizations

Fig. 60 shows commonly used ubiquitous computing blocks for low-power FPGAs, including FIR filter, FFT, CORDIC, SPI bus, FIFO, dynamic power management unit, I2C bus, and so on. Most of these blocks are already used in ultra-low-energy wireless body sensor network SoCs [46] [11], and are also very useful in SoCs of other ubiquitous computing applications, such as smart home, smart car, and environmental sensors. The size of these applications varies from 31 to 15k equivalent 4-input LUTs. Some of the applications also take advantage of hard IPs.

5.4 Benchmark Synthesis Flow

5.4.1 Overview

As discussed above, the custom synthesis flow should be able to support both custom FPGA architectures and all common-used Verilog syntax. To address this, I add a pre-synthesis step to ODIN II. The key idea is to use commercial ASIC synthesis tools to firstly convert Verilog into gate-level descriptions with some custom modifications so that ODIN II can recognize, then use ODIN II to map applications to custom FPGAs.

5.4.2 Description of the Flow

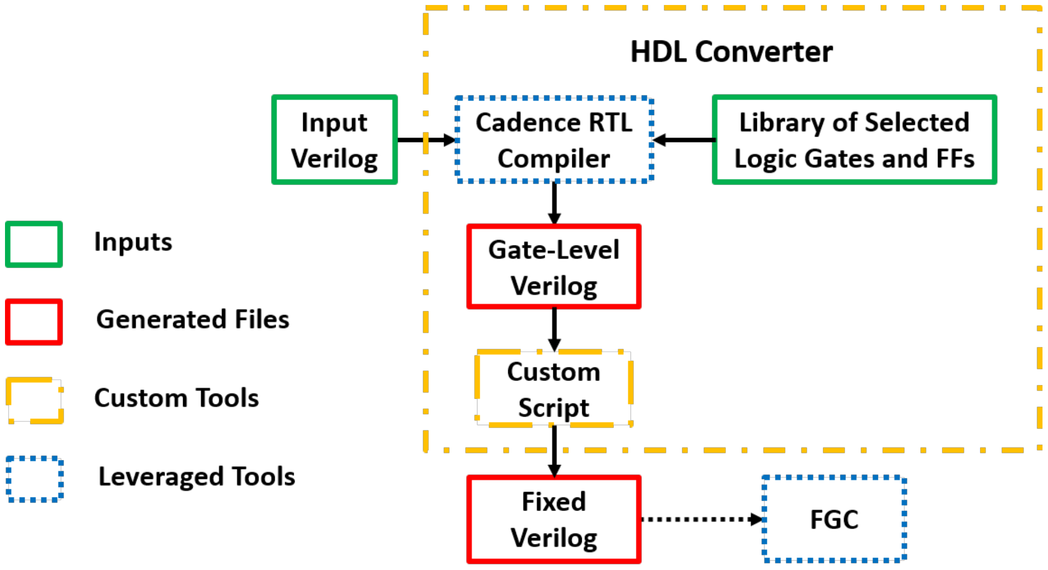


Figure 61: The Custom Benchmark Synthesis Flow

Fig. 61 describes shows the flow chart of the custom synthesis tool. The green boxes represent original input files to the tool; the blue represents the existing tools leveraged; the red represents the generated internal and final output files; the yellow represents the custom scripts. As described before, the key idea is to use commercial ASIC synthesis tools for an additional pre-synthesis step before using ODIN II. Here, Cadence RTL Compiler is chosen to convert the input file into a gate-level Verilog using only "assign" statements basic logic elements, including NAND gates, NOR gates, inverters, flip-flops, and buffer. This is because ODIN II has very strict requirements on the syntax of "initial" block, "always" block, and loops. So, pre-synthesizing all of these statements in the original Verilog into gate-level descriptions guarantees the functionality of ODIN II. To perform pre-synthesis, a customized library file ("Library of Selected Logic Gates and FFs" shown in the figure) is created as a input of Cadence RTL Compiler. This library only contains the standard cells described above.

Ideally, ODIN II should now be able map the gate-level Verilog to LUTs and IPs on FPGAs. However, the gate-level Verilog still needs additional modifications in both syntax and content to meet the requirements of using ODIN II. For example, ODIN II does not understand the functionality of standard cells. To solve this problem, a behavior-level module for each standard cell in the gate-level Verilog should be added before running ODIN II. Another example is ODIN II requires the order and number of pins in instances exactly match to the definitions of modules. So additional work of reordering active pins and removing unused pins in instances is needed. There are also many other syntax issues. Since fixing these issues by human is very complicated and easy to make mistakes, a custom script is made to automatically to this. The ODIN-ready version of the application is called "Fixed Verilog" in Fig. 61. The entire process described above is wrapped into a "HDL Converter" written in Python. In one click, the “Input Verilog” can be converted to “Fixed Verilog” in several minutes.

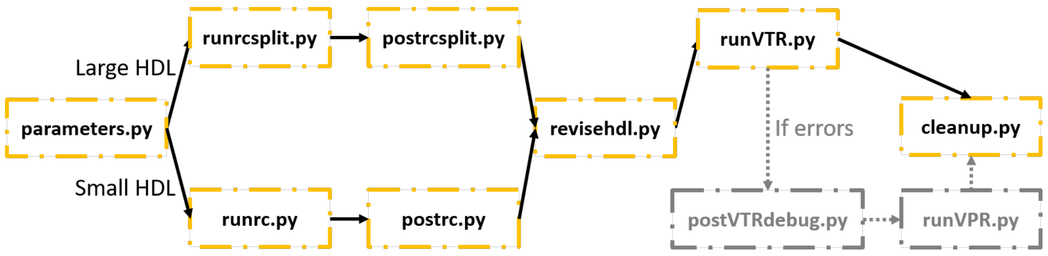


Figure 62: Flow Chart of “HDL Converter” in the Synthesis Tool

The “HDL converter” includes multiple files with different functions. Fig. 62 shows the flow chart of “HDL converter”. The detailed function of each script is described below.

- **parameters.py:** This is the start point of the flow. Users need to specify the path to the input Verilog and FPGA architecture file, the name of the top module of the input Verilog, and the name of clock. This script will use this provided info to modify other scripts, which can’t run correctly without running “parameters.py” first.
- **runrc.py:** This script generates files needed to run Cadence RTL Compiler, then run it.
- **postrc.py:** Cadence RTL Compiler will generate a lot of internal files and save them to the folder of “HDL converter”. This script deletes these intermediate files, and move the useful outputs into a pre-defined folder for next steps.
- **revisehdl.py:** This script converts the “Gate-Level Verilog” to the “Fixed Verilog”. The final ODIN-ready Verilog will be created after running this script.
- **runVTR.py:** This script generates files required for running VTR and run it using the “Fix Verilog”. The application synthesis process will be done after

this step. All the packing, place, and routing information of the application will be saved in a pre-defined folder.

- **cleanup.py:** This script deletes all the files generated by HDL converter.

The core scripts described above works for most of the applications. However, there are additional scripts to deal with special situations.

- **runrcsplit.py:** Different versions of Cadence RTL Compiler have different limits on the maximum supported gate count of the input Verilog. Excepts the most expensive version, other versions cannot synthesis large applications. To improve the impact of this tool, I modified it so that institutions and organizations using entry-level licenses can still use this tool. To solve the problem, running "runrcsplit.py" instead of "runrc.py" can split large Verilog inputs into multiple small pieces and synthesis them one by one. The synthesis process of each piece is same as running "runrc.py".

- **postrcsplit.py:** "postrcsplit.py" is also designed for synthesizing large Verilog files. Compared to "postrc.py", "postrcsplit.py" combines the synthesized outputs into one file, then performs the same functionality of "postrc.py".

- **postVTRdebug.py:** The basic flow covers most of the problems observed in the debugging process. However, in some cases, ODIN II fails when synthesizing a Verilog without any error message. By contacting the authors of ODIN II and VTR, this is a bug in their tool without easy solution. However, the GDB debugging tool embedded in Linux can help to generate these error messages. "postVTRdebug.py" is a script to perform GDB debugging in these cases.

- **runVPR.py:** Once fixing the errors indicated by GDB, the BLIF file (that logically equivalent to the input Verilog but written in a syntax that matches

FPGA architecture) will be generated. It is an intermediate file in VTR flow. By running "runVPR.py", the tool will continue from this intermediate point and finally generates the packing, place, and routing information of the input Verilog.

5.4.3 Integration of the Synthesis Flow and FGC

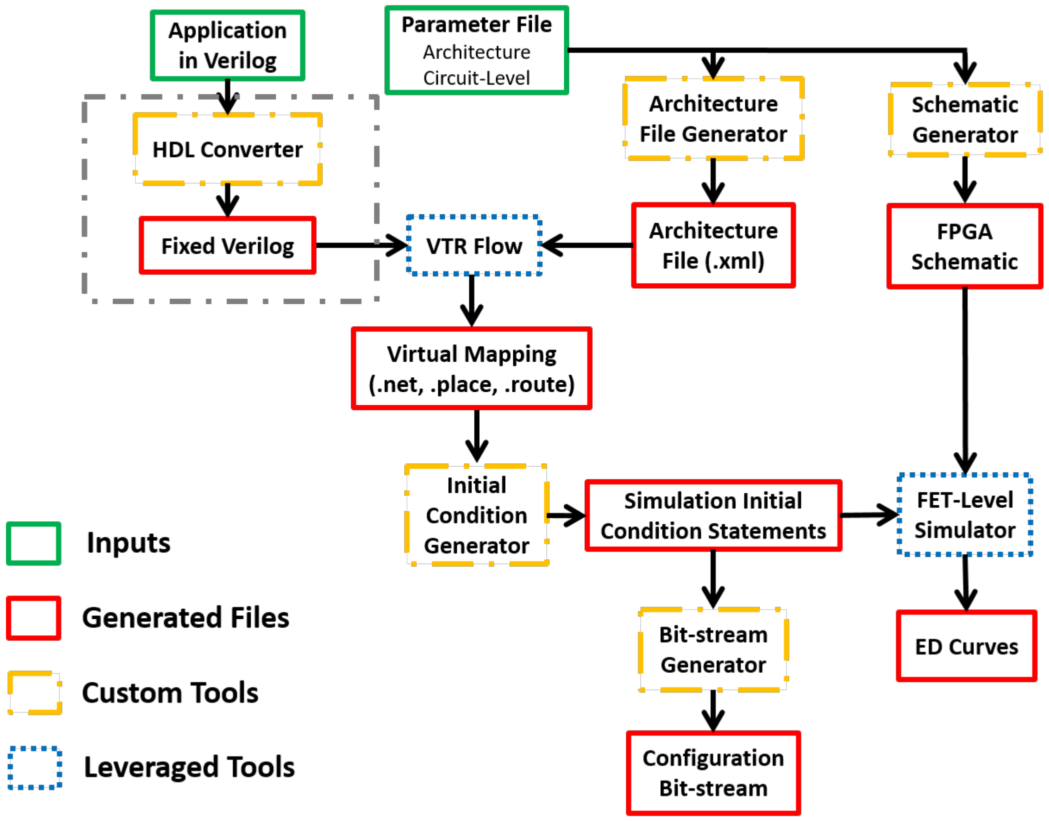


Figure 63: Integration of the Synthesis Flow and FGC

The Verilog synthesis flow can either be used separately or be integrated into FGC flow. The FGC has three key steps: 1) synthesizing input Verilog to BLIF format, 2) mapping the application in BLIF format to the target FPGA architecture and performing P&R, 3)converting P&R output info into FPGA configuration bit-stream. The synthesis flow can be directly integrated into the FGC flow by adding a new step before running FGC, as the grey dotted-lie box shown in Fig. 63. Since the input files of the synthesis flow is a subset of inputs to FGC, no additional scripts are needed to do the integration.

5.5 Fast Power Estimation Flow of the Custom FPGA

5.5.1 Description of the Fast Power Estimation Flow of the Custom Low-Power FPGA

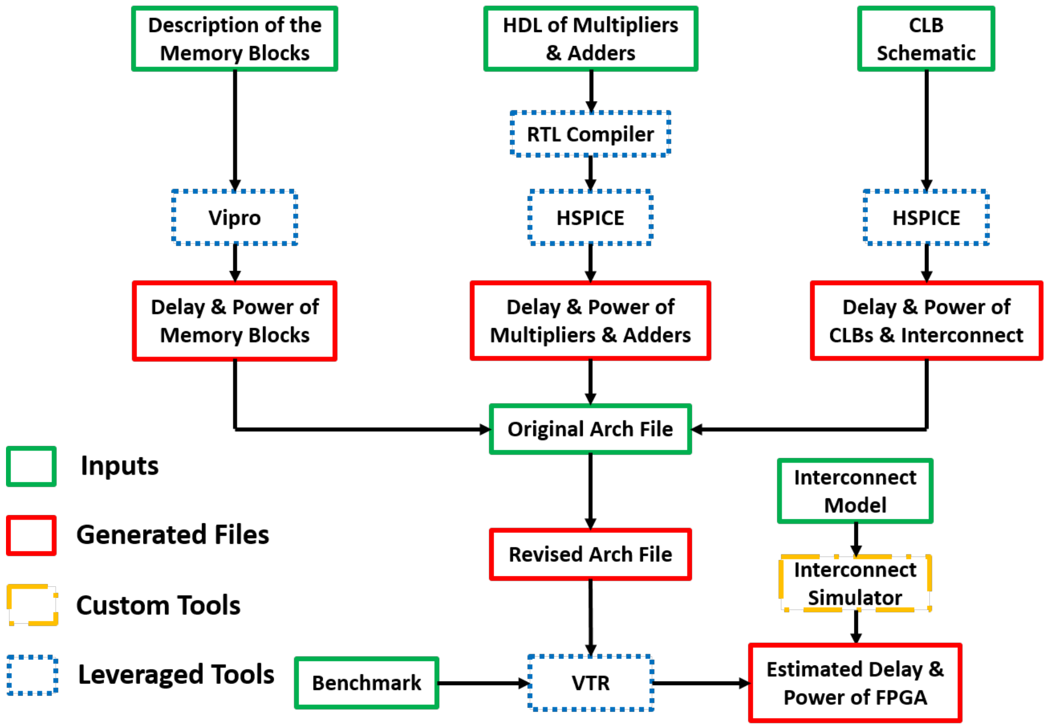

Description of the Memory Blocks
HDL of Multipliers & Adders
CLB Schematic
RTL Compiler
Vipro
HSPICE
HSPICE
Delay & Power of Memory Blocks
Delay & Power of Multipliers & Adders
Delay & Power of CLBs & Interconnect
Original Arch File
Inputs
Generated Files
Custom Tools
Leveraged Tools
Interconnect Model
Revised Arch File
Interconnect Simulator
Benchmark
VTR
Estimated Delay & Power of FPGA

Figure 64: The Fast Power Estimation Flow of the Custom Low-Power FPGA

Fig. 64 shows the details of the custom flow. Since VTR is a good platform of FPGA evaluation, the custom flow is based on VTR, but with several improvements suitable for estimating the custom FPGA. The idea is using custom scripts and existing tools to estimate the energy of each sub-circuit or IP block comprising the FPGA, then creating a FPGA architecture file using the estimated values, and finally running

VTR using that architecture file as an input. The only exception is the interconnect, which is estimated independently without using VTR estimation flow. A different strategy is used to estimate the energy of each sub-circuit, as discussed below.

- **CLBs:** The VTR flow provides multiple power estimation methods. The most accurate one is allowing VTR to run SPICE simulations for all sub-circuits of each CLB, then add them up. Since VTR supports custom CLBs, this estimation is accurate. However, this method is still slow. An alternative method is based on user-provided characterizations of sub-circuits, energy-per-toggle of each input of the CLBs, and the leakage energy of one CLB, the P&R info, and the activity factor of each CLB input pin. This method is selected in the custom flow. The user-provided numbers of the custom FPGA are pre-simulated in HSPICE. Once changes are made in circuit, only the modified part needs to be re-simulated.

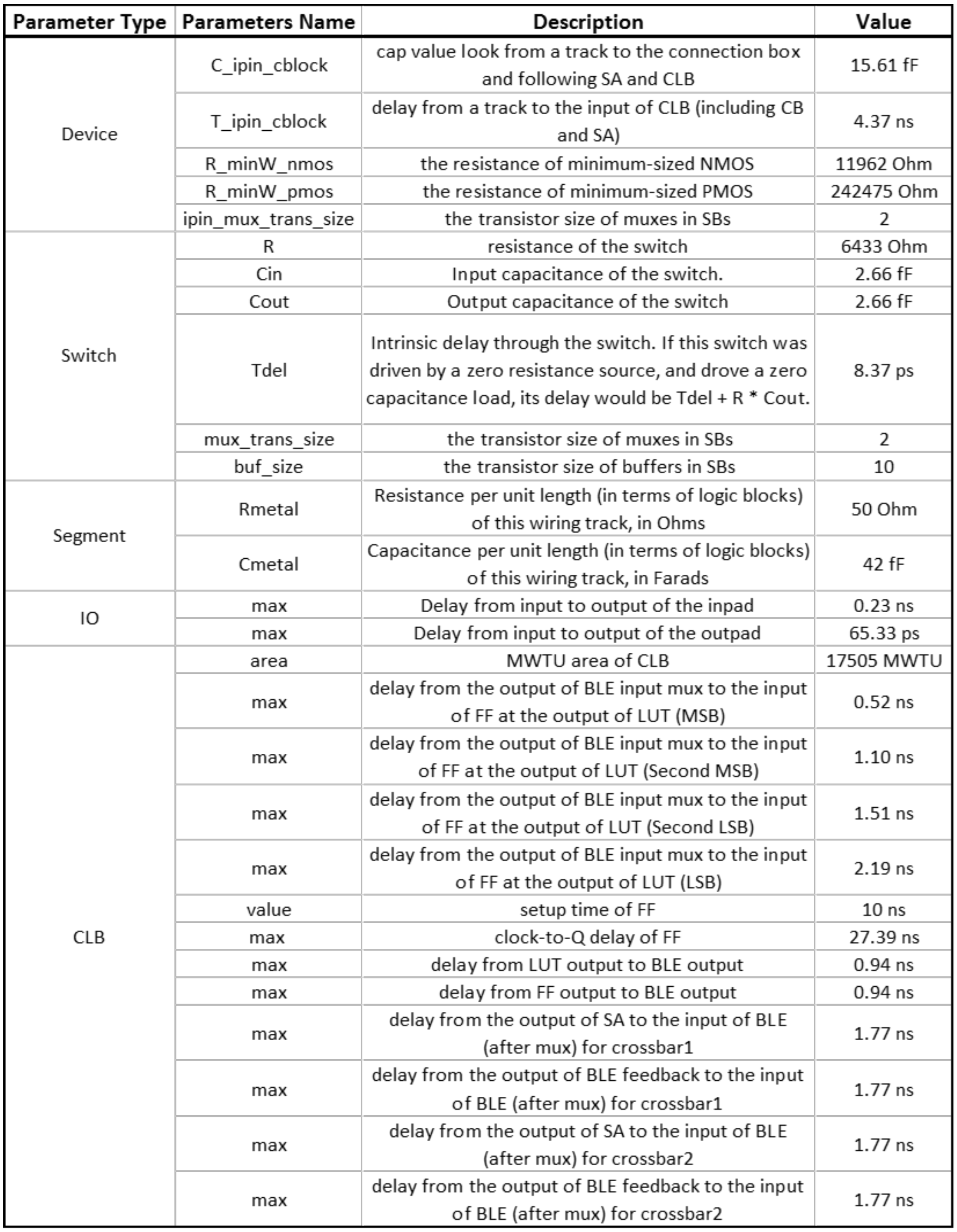
- **Global Interconnect:** The VTR flow only works for uni-directional buffer-based full-swing interconnect. If users specify another interconnect design in the architecture file, VTR flow will stop working. For the custom low-power FPGA, since low-power techniques (power-gating, per-path voltage scaling, low-swing, etc.) are applied on the interconnect, VTR will overestimate the energy of the interconnect. For this reason, I pre-simulated the custom interconnect with varying path lengths. A custom script is then created to estimate the total energy of the interconnect by summing up the energy of each path based-on the path distribution information and the transition rate of each path from intermediate files of VTR. There is no need to re-simulate the interconnect when using the flow.

- **SRAM Blocks:** VTR estimates the energy of memory blocks by using user-provided energy-per-toggle numbers. However, simulating memory blocks to obtain these numbers is very time-consuming. Thus, I choose to use Vipro [47] [48], which is a fast memory evaluation tool made by researchers at UVA. Vipro estimates memory energy by running HSPICE for each of its sub-circuit. After providing column number, row number, bank number, and word size, Vipro will run and can complete in minutes with less than 10% loss in accuracy.

- **Multipliers:** VTR estimates multiplier energy in the same way as memory blocks. To my best knowledge, there is no open source tools that are both fast and accurate for estimating multipliers. Since multipliers are widely used and simple in structure, I choose to synthesis the Verilog of the multipliers in Cadence RTL Compiler, import them into Cadence Virtuoso, then pre-simulate them in HSPICE. The transistor-level simulations guarantee the accuracy of the results. Again, the SPICE simulation does not need to be re-ran when using the flow.

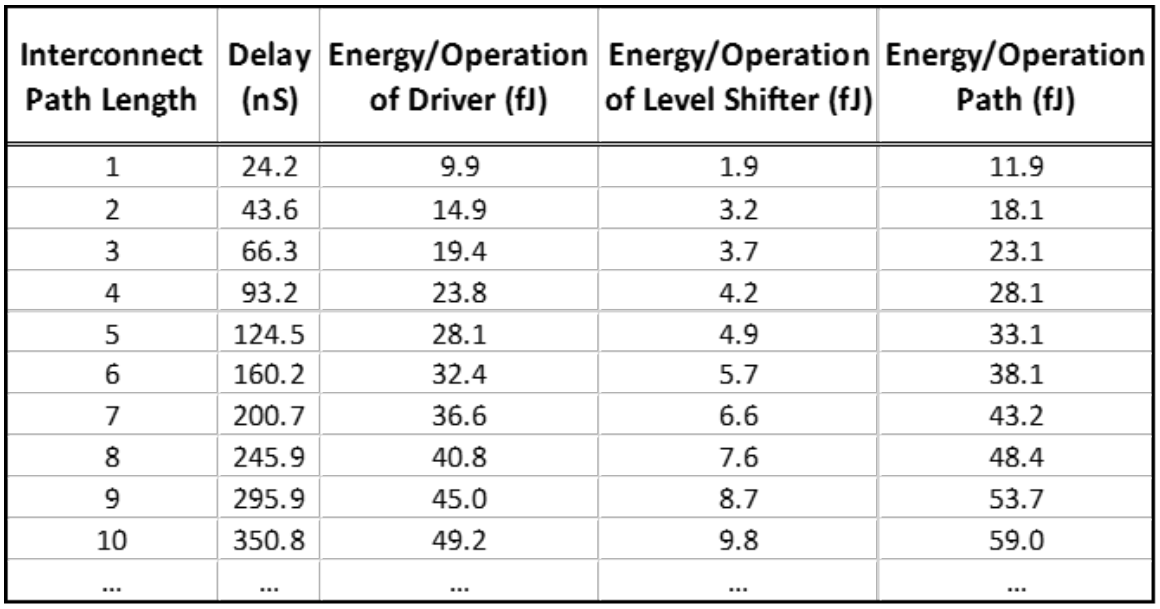
- **Others:** Other hard IPs are beyond the scope of this dissertation. It is possible that new IPs are needed to be integrated into the custom FPGA in the future. These blocks include but not limited to FFT, FIR, CORDIC, I2C, SPI, CAM, etc. Since Vipro supports CAM, using Vipro to estimate CAM is a good idea. Besides, the method used to estimate multipliers can also be applied to other blocks, as they are either small or have open source Verilog codes on the internet.

5.5.2 Parameters Used in the Flow



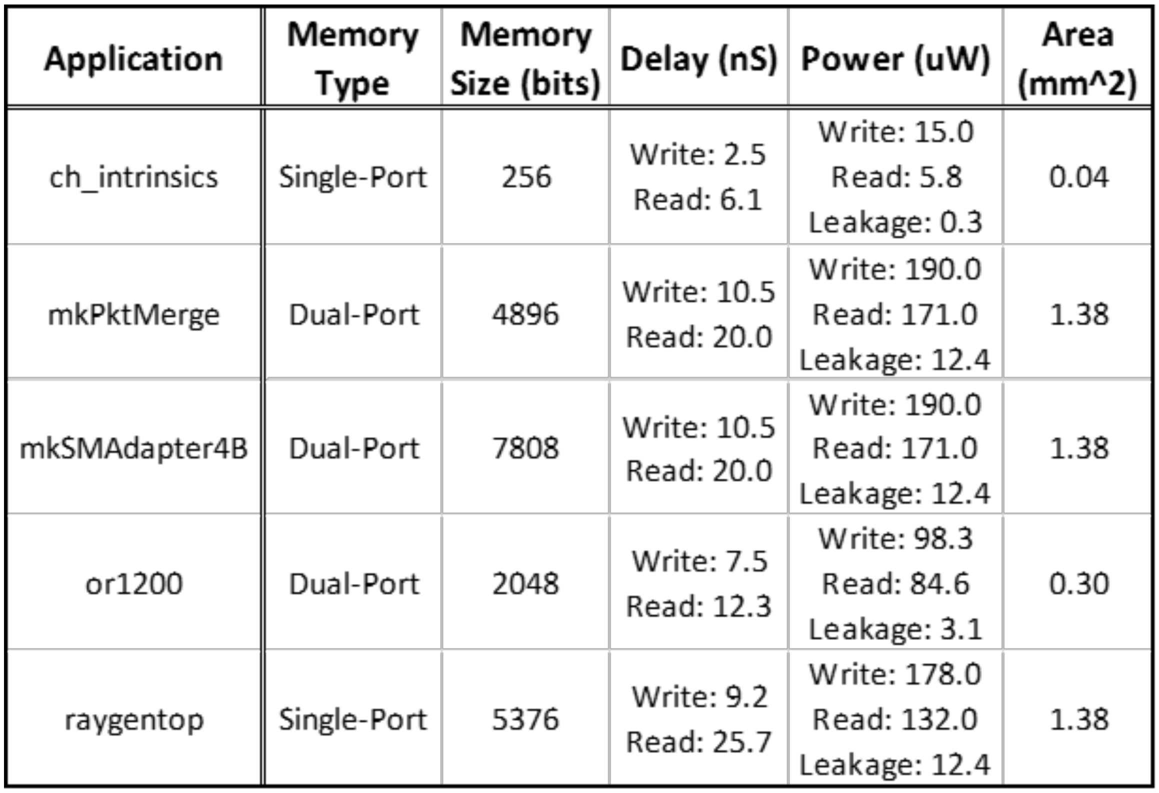
Parameter Type	Parameters Name	Description	Value
Device	C_ipin_cblock	cap value look from a track to the connection box and following SA and CLB	15.61 fF
	T_ipin_cblock	delay from a track to the input of CLB (including CB and SA)	4.37 ns
	R_minW_nmos	the resistance of minimum-sized NMOS	11962 Ohm
	R_minW_pmos	the resistance of minimum-sized PMOS	242475 Ohm
	ipin_mux_trans_size	the transistor size of muxes in SBs	2
Switch	R	resistance of the switch	6433 Ohm
	Cin	Input capacitance of the switch.	2.66 fF
	Cout	Output capacitance of the switch	2.66 fF
	Tdel	Intrinsic delay through the switch. If this switch was driven by a zero resistance source, and drove a zero capacitance load, its delay would be Tdel + R * Cout.	8.37 ps
	mux_trans_size	the transistor size of muxes in SBs	2
	buf_size	the transistor size of buffers in SBs	10
Segment	Rmetal	Resistance per unit length (in terms of logic blocks) of this wiring track, in Ohms	50 Ohm
	Cmetal	Capacitance per unit length (in terms of logic blocks) of this wiring track, in Farads	42 fF
IO	max	Delay from input to output of the inpad	0.23 ns
	max	Delay from input to output of the outpad	65.33 ps
CLB	area	MWTU area of CLB	17505 MWTU
	max	delay from the output of BLE input mux to the input of FF at the output of LUT (MSB)	0.52 ns
	max	delay from the output of BLE input mux to the input of FF at the output of LUT (Second MSB)	1.10 ns
	max	delay from the output of BLE input mux to the input of FF at the output of LUT (Second LSB)	1.51 ns
	max	delay from the output of BLE input mux to the input of FF at the output of LUT (LSB)	2.19 ns
	value	setup time of FF	10 ns
	max	clock-to-Q delay of FF	27.39 ns
	max	delay from LUT output to BLE output	0.94 ns
	max	delay from FF output to BLE output	0.94 ns
	max	delay from the output of SA to the input of BLE (after mux) for crossbar1	1.77 ns
	max	delay from the output of BLE feedback to the input of BLE (after mux) for crossbar1	1.77 ns
	max	delay from the output of SA to the input of BLE (after mux) for crossbar2	1.77 ns
	max	delay from the output of BLE feedback to the input of BLE (after mux) for crossbar2	1.77 ns

Figure 65: Table of parameters of sub-circuits of the custom low-power FPGA used for fast power estimation @ 0.6V in 130nm CMOS



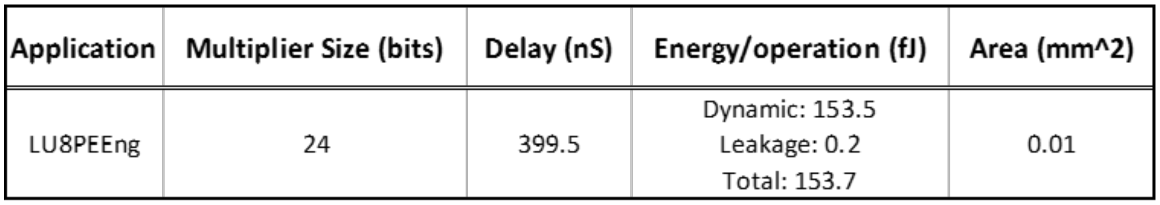
Interconnect Path Length	Delay (nS)	Energy/Operation of Driver (fJ)	Energy/Operation of Level Shifter (fJ)	Energy/Operation Path (fJ)
1	24.2	9.9	1.9	11.9
2	43.6	14.9	3.2	18.1
3	66.3	19.4	3.7	23.1
4	93.2	23.8	4.2	28.1
5	124.5	28.1	4.9	33.1
6	160.2	32.4	5.7	38.1
7	200.7	36.6	6.6	43.2
8	245.9	40.8	7.6	48.4
9	295.9	45.0	8.7	53.7
10	350.8	49.2	9.8	59.0
...	...	...	...	...

Figure 66: Table of parameters of the custom interconnect used for fast power estimation @ 0.6V in 130nm CMOS



Application	Memory Type	Memory Size (bits)	Delay (nS)	Power (uW)	Area (mm^2)
ch_intrinsics	Single-Port	256	Write: 2.5 Read: 6.1	Write: 15.0 Read: 5.8 Leakage: 0.3	0.04
mkPktMerge	Dual-Port	4896	Write: 10.5 Read: 20.0	Write: 190.0 Read: 171.0 Leakage: 12.4	1.38
mkSMAdapter4B	Dual-Port	7808	Write: 10.5 Read: 20.0	Write: 190.0 Read: 171.0 Leakage: 12.4	1.38
or1200	Dual-Port	2048	Write: 7.5 Read: 12.3	Write: 98.3 Read: 84.6 Leakage: 3.1	0.30
raygentop	Single-Port	5376	Write: 9.2 Read: 25.7	Write: 178.0 Read: 132.0 Leakage: 12.4	1.38

Figure 67: Table of parameters of the embedded memory blocks used for fast power estimation @ 0.6V in 130nm CMOS



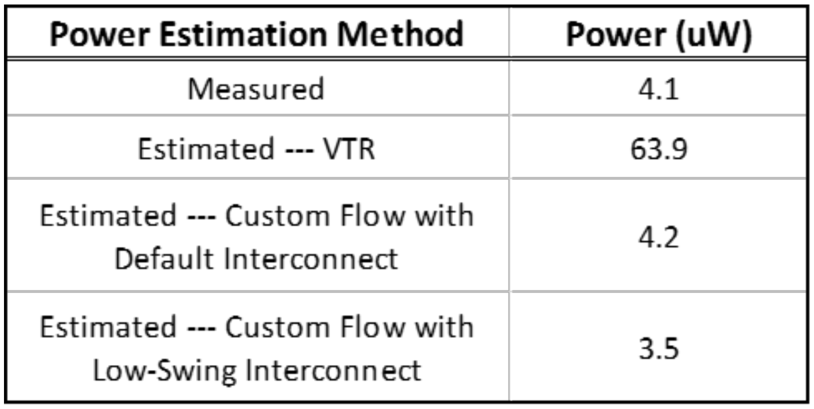
Application	Multiplier Size (bits)	Delay (nS)	Energy/operation (fJ)	Area (mm^2)
LU8PEEng	24	399.5	Dynamic: 153.5 Leakage: 0.2 Total: 153.7	0.01

Figure 68: Table of parameters of embedded multipliers used for fast power estimation @ 0.6V in 130nm CMOS

The method used for generating or simulating the parameters of each sub-circuit of the custom FPGA has been described. In Fig. 65, 66, 67, and 68, the values of all these parameters used for fast power estimation are shown. All simulations are setup in 130nm CMOS technology at 0.6V. Since the length of the interconnect path varies from 1 to over 300, it is not feasible to show all the data in this dissertation, but only the data points of ten shortest paths are shown. To minimize leakage energy and area, the size of memory blocks and multipliers should be just as large as needed. Thus, parameters of these IPs in different size are shown to be used for different applications, which are also indicated in the figures.

5.5.3 Accuracy of the Flow

Due to the limitation of VTR, a small application 4-bit adder is used to evaluate the accuracy of the custom flow. As shown in Fig. 56, the measured total power of a 4-bit adder implemented by the custom FPGA is 7.6uW at 0.6V. However, the leakage power was measured for the entire chip, while only 62 out of 64 CLBs and switch boxes are purely leaking. Thus, this part of leakage should be deducted for fair comparison. As a result, the measured power of the active circuits becomes 4.1uW. The VTR power estimation flow does not accurate. It gives a 63.9uW (15.6x higher) when estimating the same 4-bit adder. As a comparison, the result of the custom flow is 4.2uW, only 2.4% higher than silicon. However, this result is based-on the default interconnect, which is buffered unidirectional. By replacing the interconnect power with the power of the low-swing design, the estimated result reduces to 3.5uW, which is 14.6% lower than the measurement result.



Power Estimation Method	Power (uW)
Measured	4.1
Estimated --- VTR	63.9
Estimated --- Custom Flow with Default Interconnect	4.2
Estimated --- Custom Flow with Low-Swing Interconnect	3.5

Figure 69: Comparisons of the measurement result and the estimated result of a 4-bit adder

5.5.4 Integration to FGC

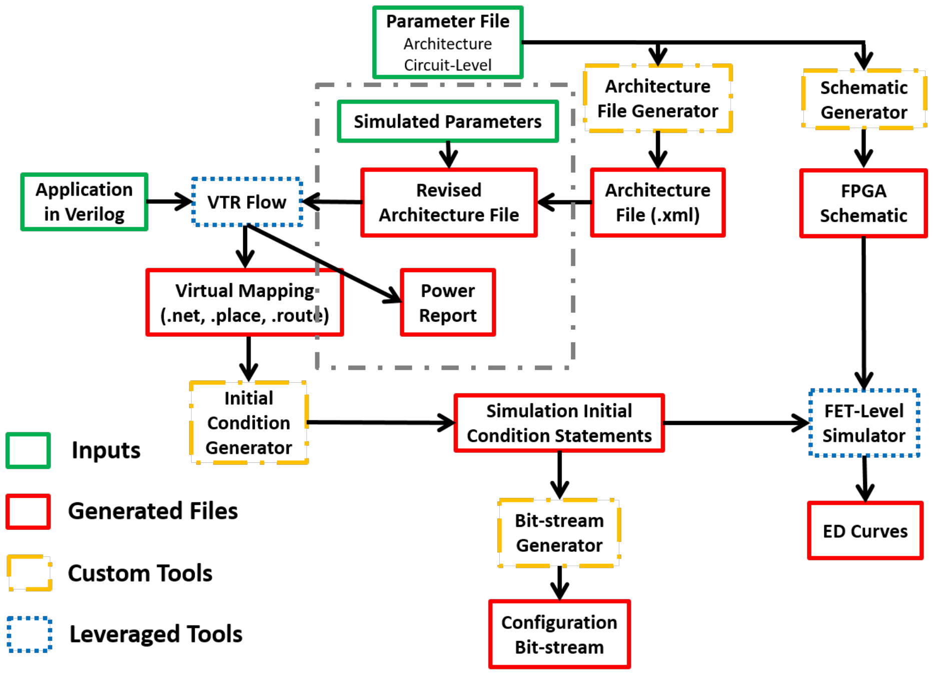


Figure 70: Integration of the Power Estimation Flow to FGC Flow

VTR has two main functions: 1) application P&R 2) FPGA power estimation. Although FGC uses VTR, it only takes advantage of the P&R part. The custom power estimation flow uses the other part of VTR by providing VTR a more accurate architecture file. The ultimate goal of all the methods described in 5.5.1 is to create this architecture file for the custom near/sub-threshold FPGA. Thus, the custom power estimation flow can be seen as an intermediate step of FGC with an additional output of power report, as shown in Fig. 70. The steps in the grey dotted-line box represents the power estimation flow.

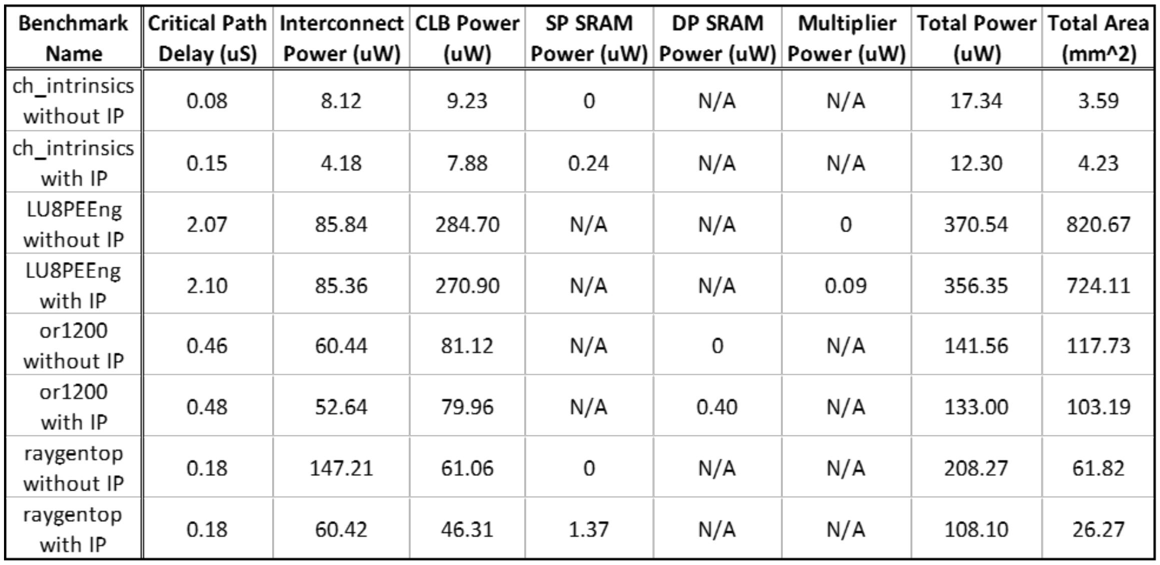
5.6 IP Integration

5.6.1 Overview

Low-power FPGAs are naturally suitable to implement ubiquitous computing sensor nodes, which require both flexibility and low-power. In addition to collecting data, it is desired that those sensor nodes have local intelligence to extract information and control locally. For example, body sensor nodes are expected to analysis personal health information locally and clearly show to users. Implementing signal processing algorithms by FPGAs usually requires large number of LUTs, which is not energy and area efficient. Thus, some commonly used algorithms can be implemented by hard IP blocks on FPGAs to dramatically improve energy efficiency and reduce area. For example, in [4], FPGAs with hard IPs are on average 12% faster, 49% less energy, and 47% less area than FPGAs without hard IPs. Although IP blocks are widely used in both commercial and academic FPGAs, the number of those blocks is easily over 1000, close to or larger than the total area of low-power FPGAs. The energy efficiency improvement numbers are also obtained by evaluating large high-performance FPGAs, which is inaccurate in near/sub-threshold operations. The energy efficiency improvement after integrating IPs on low-power FPGAs is also needed to be estimated.

5.6.2 Results

The size and number of the embedded hard IPs need to be carefully selected to meet the requirements of the low-power applications described early in this chapter. To minimize leakage energy and area, the exact size and number of each IP needed in a given application are used. For example, "ch_intrinsics" has eight instances of the 256-bit single-port SRAM block in its Verilog, so the exact same number and size of single-port SRAM blocks are added into the FPGA architecture file. The detailed info of each IP used in each application is shown in Fig. 67 and 68. Compared to high-performance FPGAs, which has over 1000 hard IPS, the number of selected IPs on the custom FPGA is no more than 64, which is very suitable for implementing low-power applications.



Benchmark Name	Critical Path Delay (uS)	Interconnect Power (uW)	CLB Power (uW)	SP SRAM Power (uW)	DP SRAM Power (uW)	Multiplier Power (uW)	Total Power (uW)	Total Area (mm^2)
ch_intrinsics without IP	0.08	8.12	9.23	0	N/A	N/A	17.34	3.59
ch_intrinsics with IP	0.15	4.18	7.88	0.24	N/A	N/A	12.30	4.23
LU8PEEng without IP	2.07	85.84	284.70	N/A	N/A	0	370.54	820.67
LU8PEEng with IP	2.10	85.36	270.90	N/A	N/A	0.09	356.35	724.11
or1200 without IP	0.46	60.44	81.12	N/A	0	N/A	141.56	117.73
or1200 with IP	0.48	52.64	79.96	N/A	0.40	N/A	133.00	103.19
raygentop without IP	0.18	147.21	61.06	0	N/A	N/A	208.27	61.82
raygentop with IP	0.18	60.42	46.31	1.37	N/A	N/A	108.10	26.27

Figure 71: Comparisons of the Custom Low-Power FPGA with and without Hard IPs

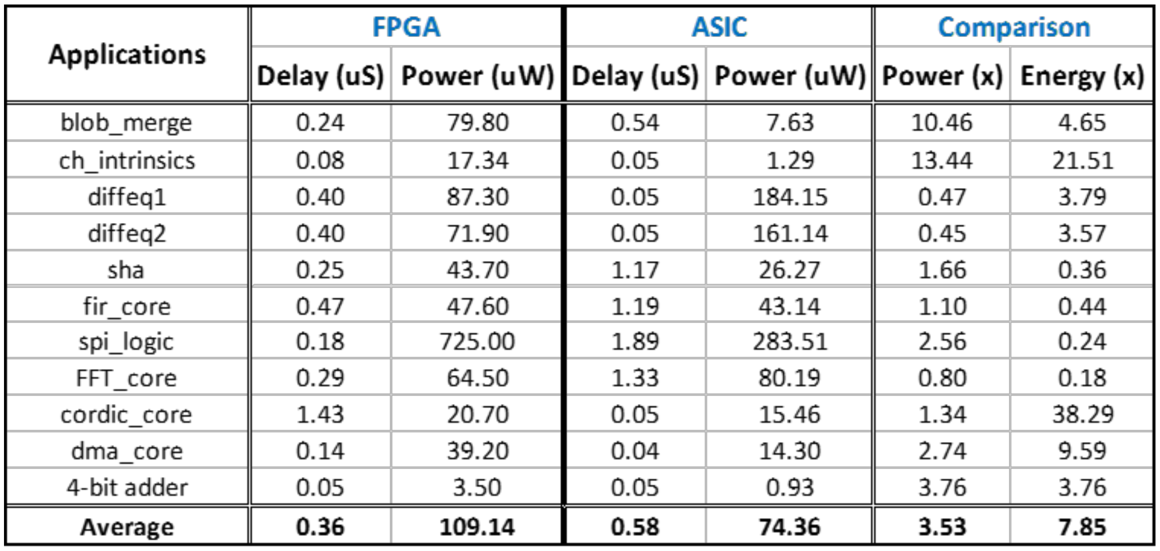
Fig. 71 shows the comparisons of the custom low-power FPGA implementing selected applications when using and not using hard IPs, which include single-port SRAM, dual-port SRAM, and multiplier. By using these IPs, the power, area, and delay of the FPGA changes as follows.

- **Power:** The power of the listed applications is all reduced, and by 21.8% on average.
- **Area:** The area of "ch intrinsics" increases by 17.9% after using memory blocks. For other benchmarks, the area reduces by 27.2% on average.
- **Delay:** The delay of "ch intrinsics" increase by 79.4% after using memory blocks. For other benchmarks, the delay doesn't change much, increases by 1.6% on average.

Using hard IPs is not easy for ODIN II users. During application synthesis, ODIN II cannot automatically understand and decide which part of the RTL code can be implemented by hard IPs. The only way to take advantage of the hard IPs is to modify the RTL code and explicitly use hard IPs as instances. For a large Verilog, this method can be very complicated since users have to re-write a big portion of code and re-evaluate the code. To make things worse, when the users want to enable or disable the usage of some of the IPs, they always need to modify the code. For this reason, not all of the low-power applications introduced in 5.3 are shown here. To solve this problem, the fundamental engine of VTR needs to be modified, which is far beyond the scope of this dissertation.

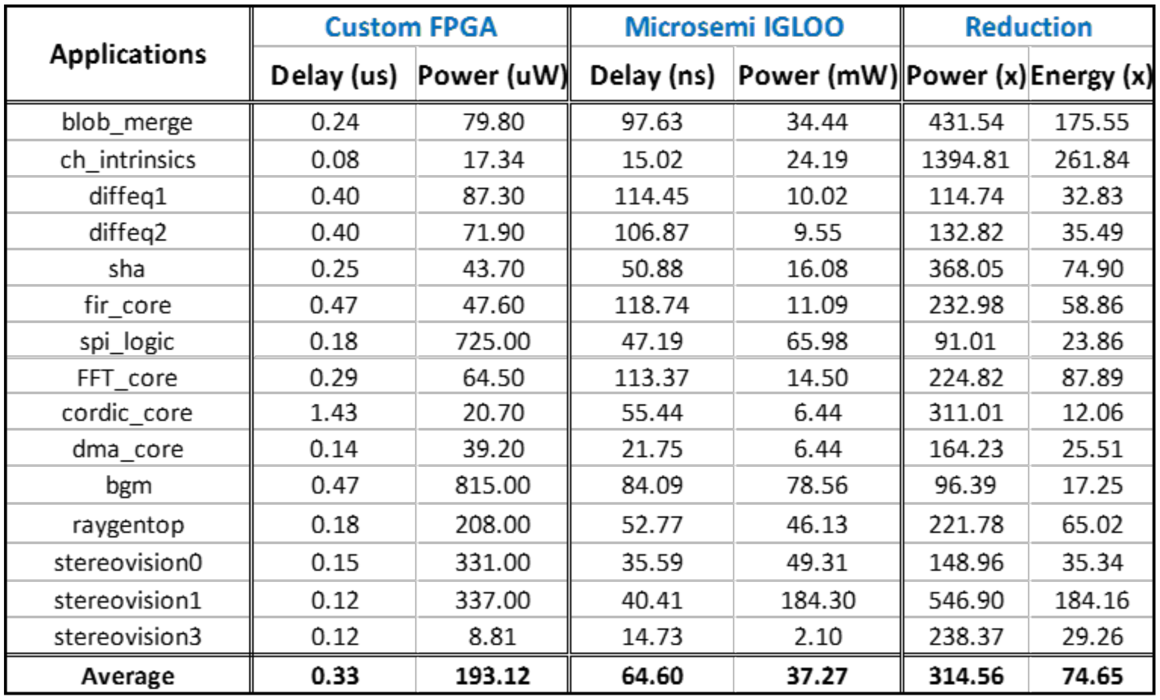
5.7 Results

In [4], researchers compared commercial high-performance FPGAs and ASICs. They suggest FPGAs on average consume 14x more active power compared to ASICs implementing same applications. It worth to do the same comparison for low-power applications to assist the selection of different computing platforms. To do so, I used the custom flow to estimate the power of the custom FPGA implementing the applications in Fig. 72. The estimation results of ASICs are obtained by running Cadence RTL Compiler. As a result, the custom low-power FPGA consumes 3.52x more power and 7.85x power-delay product than ASICs on average when implementing the applications listed in Fig. 72. The comparisons between the custom FPGA and the Microsemi IGLOO, which is lowest power FPGA on the market, are made. As indicated by Fig. 73, the custom FPGA has 315x average power reduction and 75x average energy reduction compared to IGLOO when implementing the custom low-power application suite. The data of IGLOO is obtained by running their Microsemi's power estimation tool. To make fair comparison, only the core power of IGLOO is considered. Also, same activity factors and toggle rates are applied to both FPGAs.



Applications	FPGA		ASIC		Comparison	
	Delay (uS)	Power (uW)	Delay (uS)	Power (uW)	Power (x)	Energy (x)
blob_merge	0.24	79.80	0.54	7.63	10.46	4.65
ch_intrinsics	0.08	17.34	0.05	1.29	13.44	21.51
diffeq1	0.40	87.30	0.05	184.15	0.47	3.79
diffeq2	0.40	71.90	0.05	161.14	0.45	3.57
sha	0.25	43.70	1.17	26.27	1.66	0.36
fir_core	0.47	47.60	1.19	43.14	1.10	0.44
spi_logic	0.18	725.00	1.89	283.51	2.56	0.24
FFT_core	0.29	64.50	1.33	80.19	0.80	0.18
cordic_core	1.43	20.70	0.05	15.46	1.34	38.29
dma_core	0.14	39.20	0.04	14.30	2.74	9.59
4-bit adder	0.05	3.50	0.05	0.93	3.76	3.76
Average	**0.36**	**109.14**	**0.58**	**74.36**	**3.53**	**7.85**

Figure 72: Comparisons of the custom low-power FPGA and ASICs when implementing the custom low-power application suite



Applications	Custom FPGA		Microsemi IGLOO		Reduction	
	Delay (us)	Power (uW)	Delay (ns)	Power (mW)	Power (x)	Energy (x)
blob_merge	0.24	79.80	97.63	34.44	431.54	175.55
ch_intrinsics	0.08	17.34	15.02	24.19	1394.81	261.84
diffeq1	0.40	87.30	114.45	10.02	114.74	32.83
diffeq2	0.40	71.90	106.87	9.55	132.82	35.49
sha	0.25	43.70	50.88	16.08	368.05	74.90
fir_core	0.47	47.60	118.74	11.09	232.98	58.86
spi_logic	0.18	725.00	47.19	65.98	91.01	23.86
FFT_core	0.29	64.50	113.37	14.50	224.82	87.89
cordic_core	1.43	20.70	55.44	6.44	311.01	12.06
dma_core	0.14	39.20	21.75	6.44	164.23	25.51
bgm	0.47	815.00	84.09	78.56	96.39	17.25
raygentop	0.18	208.00	52.77	46.13	221.78	65.02
stereovision0	0.15	331.00	35.59	49.31	148.96	35.34
stereovision1	0.12	337.00	40.41	184.30	546.90	184.16
stereovision3	0.12	8.81	14.73	2.10	238.37	29.26
Average	**0.33**	**193.12**	**64.60**	**37.27**	**314.56**	**74.65**

Figure 73: Comparisons of the custom low-power FPGA and Microsemi IGLOO when implementing the custom low-power application suite

5.8 Future Research

The custom tool and flow can be further improved in five directions.

- **Expanded Application Suite:** The low-power FPGA application suite includes diverse types of on-chip buses, building blocks of personal health systems, hardware security, computer visions, etc. However, since IoT plays an important role in low-power applications, more IoT applications can be added to the suite. Since many IoT applications are under-developed and remains secret in companies, not all the RTL can be accessed. However, future researchers can build RTL prototypes of these applications based on their ideas and roughly estimated its performance, power, and area on FPGAs. These applications include but not limited to smart homes, smart grid, and autonomous driving.

The low-power FPGAs have potentials to implement blocks comprising these applications or the entire application.

- **Advanced Automation:** Currently, the FPGA evaluation platform can estimate delay, power, and area of the custom FPGA. However, users need to manually change the parameters of CLBs, interconnect, and hard IPs in the architecture file when evaluating other FPGAs. Although the value of these parameters is obtained from reliable external tools (like Cadence RTL Compiler or Vipro), the process is troublesome and is easy to make errors. It is desired that the FPGA evaluation flow and the external tools can be linked together in the future. This means the flow can automatically run the external tools and modify the corresponding parameters in the architecture file.

- **Design Constraints:** Although the custom flow can evaluate FPGAs, it cannot optimize FPGAs base on the input design constraints. VTR do support Synopsys Design Constraints (SDC). However, only timing constraint is mentioned in their user manual (no power, and area constraints). In addition, since VTR is not based on Synopsys tools, the functionality and reliability of VTR on timing constraints need to be further verified.

- **Additional Hard IPs:** So far, the custom FPGA evaluation flow supports embedded memory blocks and multipliers on FPGAs. However, as more and more IoT applications are coming out, it is possible that more types of hard IPs need to be integrated to low-power FPGAs. For example, although the embedded multipliers help improving hardware performance and power significantly when implementing FIRs, the FIR itself can be customized and used as a new type of hard IPs on low-power FPGAs.

- **Auto IP Recognition:** As discussed, the current flow cannot recognize RTL descriptions that can use hard IPs automatically, making enabling and disabling IP usage very complicated. This can be solved in the future by introducing RTL template for finding potential IPs in applications. By searching potential usage of IPs and modify the syntax in the original RTL, it is possible to enable auto IP recognition.

6 Conclusion

Low-power miniature systems for ubiquitous computing such as wireless sensor networks have been developing rapidly in the past years. The growing demand on collecting and analyzing information from surrounding environment drives researchers and engineers focus on developing internet-of-things. This demand indicates the need of ultra-low-power computing platforms. In this dissertation, the circuit/architecture and tool flow of an ultra-low-power FPGA are explored and developed. More specifically, three research topics have been discussed in detail:

- Energy Efficient FPGA Interconnect
- Per-Path Voltage Scaling and Power-Gating
- Low-Power FPGA Evaluation Platform

6.1 Energy Efficient FPGA Interconnect

As the major energy consumer, an energy-efficient interconnect is required for ultra-low-power FPGAs. In this dissertation, a low-swing interconnect architecture, which has been proven to be an energy-efficient design in near/sub-threshold, is carefully modeled, optimized, and evaluated. The simulation and chip measurement results indicate the following recommended circuit parameters and supply voltage of the interconnect in terms of energy efficiency.

- Interconnect Structure: Low-Swing
- Supply Voltage: $V_{DD} = 0.4/0.5\text{V}$, $V_{DDC} = V_{DD} + 0.2\text{V}$
- Driver Size: 10/20x
- Routing Switch Structure: Pass-Gates

- Routing Switch Size: 2/4x
- Connection Box Structure: 2-Stage Multiplexer
- Connection Switch Size: 2/4x
- Number of Inserted Level Shifters (except the one at the destinations of paths): 0

The above recommendations are valid for interconnect paths less than 40, which consume over 95% of interconnect energy when implementing MCNC benchmarks. In additional, the physical implementation of the interconnect is also optimized, reducing the area of an 84-track switch box by 26x. As an overall conclusion, the work of this project reduces FPGA interconnect delay and energy by 68.4% and 47.5% respectively when implementing MCNC benchmarks.

6.2 Per-Path Voltage Scaling and Power-Gating

The multi-V_{DD} or per-path voltage scaling is an attractive energy reduction technique that can be used on FPGAs. However, no existing work implemented it on FPGA interconnect due to an area overhead of up to 43%. In this dissertation, the low-swing interconnect enables per-path voltage scaling with near zero area overhead. In addition, strategies of applying power-gating on configuration bit-cells are explored and simulated. A custom tool flow is developed to evaluate the energy reduction of using per-path voltage scaling and power-gating on FPGA interconnect when implementing MCNC benchmarks. The results indicate:

- Optimal voltage for per-path voltage scaling in terms of energy reduction: V_{DDH} = 0.6V, V_{DDL} = 0.5V

- Energy reduction of using per-path voltage scaling without considering voltage regulator overhead: 20.1%
- Energy reduction of using per-path voltage scaling including voltage regulator overhead: 10.1%
- Energy reduction of using per-path voltage scaling and power-gating including voltage regulator overhead: 22.3% - 56.5%

A 512-LUT FPGA using low-swing interconnect, per-path voltage scaling, and power-gating is fabricated in 130nm CMOS. The measurement result shows a 43% leakage energy reduction by using power-gating. The exact leakage power of the chip is as low as 3.6uW at 0.6V, which is 2.5x smaller than the lowest existing work that is 4x smaller than ours at a much lower operating voltage of 0.26V. In addition, when implementing a 4-bit adder, the custom FPGA consumes 7.6uW at 0.6V, which is 277x lower than Microsemi IGLOO.

6.3 Low-Power FPGA Evaluation Platform

To evaluate the custom ultra-low-power FPGA, an application suite is created by exploring and selecting existing benchmarks written in Verilog, including personal hearth sensors, commercial low-power applications, interfaces, and so on. Since no existing benchmark synthesis flow supports both custom FPGA architectures and all Verilog syntax, a custom flow is developed. Its functionality is verified using the custom low-power application suite. In order to fast estimate the power consumption of the custom FPGA, a power estimation flow is also developed. The difference between the estimated power and the measured power from the chip of a 4-bit adder is within 15%. Both flows are integrated into the existing VTR flow and FGC. Comparisons between the custom FPGA and ASICs are made using the custom flow.

When implementing the proposed application suite, the custom FPGA consumes 3.52x higher power and 7.85 higher energy on average than the ASIC counterpart. When compared to Microsemi IGLOO, the custom FPGA on average consumes 315x less power and 75x less energy.

Appendix A List of Publications

A.1 Publications

- **Qi, He**, Oluseyi Ayorinde, Yu Huang, and Benton H. Calhoun. “Optimizing energy efficient low-swing interconnect for sub-threshold FPGAs”, In Field Programmable Logic and Applications (FPL), 2015 25th International Conference on, pp. 1-4. IEEE, 2015.
- **Qi, He**, Oluseyi Ayorinde, and Benton H. Calhoun. “An energy-efficient near/sub-threshold FPGA interconnect architecture using dynamic voltage scaling and power-gating”, In Field Programmable Technology (FPT), 2016 International Conference on, pp. 20-27. IEEE, 2016.
- **Qi, He**, Oluseyi Ayorinde, and Benton H. Calhoun. “An Ultra-Low-Power FPGA for IoT Applications”, S3S 2017, In Press.
- Ayorinde, Oluseyi, **He Qi**, Yu Huang, and Benton H. Calhoun, “Using island-style bi directional intra-CLB routing in low-power FPGAs”, In Field Programmable Logic and Applications (FPL), 2015 25th International Conference on, pp. 1-7. IEEE, 2015.

A.2 Pending Publications

- Ayorinde, Oluseyi, **He Qi**, and Benton H. Calhoun, FGC: A Toolflow for Generating and Configuring Custom FPGAs.

Appendix B Glossary of Terms

- FPGA - abbreviation of “Field Programmable Gate Array”
- CLB - abbreviation of “Configurable Logic Block”, which consists a cluster of interconnected BLEs
- BLE - abbreviation of “Basic Logic Element”, which consists a LUT, a FF and a multiplexer
- LUT - abbreviation of “Look-Up-Table”
- FF - abbreviation of “Flip-Flop”
- CB - abbreviation of “Connection Box”, which is a cluster of switches connecting CLBs and the global interconnect
- SB - abbreviation of “Switch Box”, which is a cluster of routing switches at the intersection of horizontal tracks and vertical tracks in the global interconnect
- SP - abbreviation of “Switch Point”, which is a sub-circuit of switch box
- IoT - abbreviation of “Internet-of-Things”
- VTR - abbreviation of “Verilog-to-Routing”, which is an open source tool to do custom FPGA place & route
- MCNC - a widely used FPGA benchmark suite in academia
- DVS - abbreviation of “Dynamic Voltage Scaling”
- ULP - abbreviation of “Ultra-Low-Power”
- EDP - abbreviation of “Energy-Delay-Product”, which is usually used to evaluate the energy-efficiency of circuit
- PDP - abbreviation of “Power-Delay-Product”, which is usually used to evaluate the power-efficiency of circuit
- FPGA Tile - a FPGA sub-circuit that consists of a CLB, a SB, and multiple CBs
- Segment Length - the number of FPGA tiles that an interconnect wire segment spans
- Configuration Bit-cell - a 5T cross-coupled memory cell used for FPGA configuration
- Subset - a type of connectivity pattern of the switch boxes

- Folded-Subset - an advanced physical implementation of Subset
- WL - abbreviation of “Word Line”, which is the control signal line of the configuration bit-cells
- BL - abbreviation of “Bit Line”, which is the data signal line of the configuration bit-cells
- BLIF - abbreviation of “Berkeley Logic Interchange Format”, which is a file format of FPGA benchmarks
- FGC - abbreviation of “FPGA Generation and Configuration”, which is an open source tool used for FPGA configuration bit-stream generation
- Vipro - an open source tool for speed and power estimation of memory blocks

References

[1] Betz, Vaughn, Jonathan Rose, and Alexander Marquardt. Architecture and CAD for deep-submicron FPGAs. Vol. 497. Springer Science & Business Media, 2012.

[2] Shang, L., A. Kaviani, and K. Bathala. "Dynamic Power Consumption inthe Virtex-II FPGA Family." ACM/SIGDA International Symposium on Field Programmable Gate Arrays.

[3] Poon, Kara KW, Andy Yan, and Steven JE Wilton. "A flexible power model for FPGAs." FPL. Vol. 2. 2002.

[4] Kuon, Ian, and Jonathan Rose. "Measuring the gap between FPGAs and ASICs." IEEE Transactions on computer-aided design of integrated circuits and systems 26.2 (2007): 203-215.

[5] Calhoun, Benton H., Alice Wang, and Anantha Chandrakasan. "Modeling and sizing for minimum energy operation in subthreshold circuits." IEEE Journal of Solid-State Circuits 40.9 (2005): 1778-1786.

[6] Rabaey, Jan M., Anantha P. Chandrakasan, and Borivoje Nikolic. Digital integrated circuits. Vol. 2. Englewood Cliffs: Prentice hall, 2002.

[7] Semiconductor, Lattice. "iCE40 Ultra Family Data Sheet." DS1048 Version 1 (2015).

[8] Microsemi Corporation, "IGLOO nano FPGA Fabric (User's Guide)," Version 1.4, March 2008.

[9] Grossmann, Peter J., Miriam E. Leeser, and Marvin Onabajo. "Minimum energy analysis and experimental verification of a latch-based subthreshold FPGA."

IEEE Transactions on Circuits and Systems II: Express Briefs 59.12 (2012): 942-946.

[10] Yuan, Fang-Li, et al. “A multi-granularity FPGA with hierarchical interconnects for efficient and flexible mobile computing.” IEEE Journal of Solid-State Circuits 50.1 (2015): 137-149.

[11] Klinefelter, Alicia, et al. “21.3 A 6.45 W self-powered IoT SoC with integrated energy-harvesting power management and ULP asymmetric radios.” Solid-State Circuits Conference-(ISSCC), 2015 IEEE International. IEEE, 2015.

[12] Rahman, Arifur, and Vijay Polavarapuv. “Evaluation of low-leakage design techniques for field programmable gate arrays.” Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. ACM, 2004.

[13] Ciccarelli, Luca, Andrea Lodi, and Roberto Canegallo. “Low leakage circuit design for FPGAs.” Custom Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004. IEEE, 2004.

[14] Anderson, Jason H., and Farid N. Najm. “Low-power programmable FPGA routing circuitry.” IEEE transactions on very large scale integration (VLSI) systems 17.8 (2009): 1048-1060.

[15] Gayasen, Aman, et al. “A Dual-VDD Low Power FPGA Architecture.” FPL. 2004.

[16] Li, Fei, Yan Lin, and Lei He. “Vdd programmability to reduce FPGA interconnect power.” Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design. IEEE Computer Society, 2004.

[17] McMurchie, Larry, and Carl Ebeling. "PathFinder: a negotiation-based performance-driven router for FPGAs." Proceedings of the 1995 ACM third international symposium on Field-programmable gate arrays. ACM, 1995.

[18] Alexander, Michael J., and Gabriel Robins. "New performance-driven FPGA routing algorithms." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 15.12 (1996): 1505-1517.

[19] Huda, Safeen, Jason Anderson, and Hirotaka Tamura. "Charge recycling for power reduction in FPGA interconnect." Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on. IEEE, 2013.

[20] Lewis, David, et al. "The stratix routing and logic architecture." Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays. ACM, 2003.

[21] Lemieux, Guy, et al. "Directional and single-driver wires in FPGA interconnect." Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on. IEEE, 2004.

[22] Ryan, Joseph F., and Benton H. Calhoun. "A sub-threshold FPGA with low-swing dual-VDD interconnect in 90nm CMOS." Custom Integrated Circuits Conference (CICC), 2010 IEEE. IEEE, 2010.

[23] Ryan, Joseph. On the Robust Design of Sustainably Managed, Reconfigurable Circuits for Ultra Low Energy Systems in Nanoscale CMOS. Diss. University of Virginia, 2011.

[24] Betz, Vaughn, and Jonathan Rose. "VPR: A new packing, placement and routing tool for FPGA research." International Workshop on Field Programmable Logic and Applications. Springer, Berlin, Heidelberg, 1997.

[25] Yang, S. “Logic Synthesis and Optimization Benchmarks Version 3.0. 1991.” Microelectronics center of North Carolina.

[26] Calhoun, Benton H., Alice Wang, and Anantha Chandrakasan. “Modeling and sizing for minimum energy operation in subthreshold circuits.” IEEE Journal of Solid-State Circuits 40.9 (2005): 1778-1786.

[27] Sverdlik, Yevgeniy. Heres How Much Energy All US Data Centers Consume. June 27, 2016. Available via http://www.datacenterknowledge.com/archives/2016/06/27/heres-how-much-energy-all-us-data-centers-consume/

[28] Koomey, Jonathan. “Growth in data center electricity use 2005 to 2010.” A report by Analytical Press, completed at the request of The New York Times 9 (2011).

[29] Timmer, John. “Datacenter energy costs outpacing hardware prices.” Ars Technica (2009).

[30] Calhoun, Benton H., and Anantha P. Chandrakasan. “Ultra-dynamic voltage scaling (UDVS) using sub-threshold operation and local voltage dithering.” IEEE Journal of Solid-State Circuits 41.1 (2006): 238-245.

[31] Pillai, Padmanabhan, and Kang G. Shin. “Real-time dynamic voltage scaling for low-power embedded operating systems.” ACM SIGOPS Operating Systems Review. Vol. 35. No. 5. ACM, 2001.

[32] Pouwelse, Johan, Koen Langendoen, and Henk Sips. “Dynamic voltage scaling on a low-power microprocessor.” Proceedings of the 7th annual international conference on Mobile computing and networking. ACM, 2001.

[33] Kim, Wonyoung, David M. Brooks, and Gu-Yeon Wei. “A fully-integrated 3-level DC/DC converter for nanosecond-scale DVS with fast shunt regulation.” Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International. IEEE, 2011.

[34] Howard, Jason, et al. “A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS.” Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International. IEEE, 2010.

[35] Chow, Chun Tak, et al. “Dynamic voltage scaling for commercial FPGAs.” Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on. IEEE, 2005.

[36] Nomura, Shuou, et al. “A 9.7 mw aac-decoding, 620mw h. 264 720p 60fps decoding, 8-core media processor with embedded forward-body-biasing and power-gating circuit in 65nm cmos technology.” Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International. IEEE, 2008.

[37] Pilo, Harold, et al. “A 64Mb SRAM in 22nm SOI technology featuring fine-granularity power gating and low-energy power-supply-partition techniques for 37% leakage reduction.” Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International. IEEE, 2013.

[38] Nair, Pradeep S., Santosh Koppa, and Eugene B. John. “A comparative analysis of coarse-grain and fine-grain power gating for FPGA lookup tables.” Circuits and Systems, 2009. MWSCAS’09. 52nd IEEE International Midwest Symposium on. IEEE, 2009.

[39] Bsoul, Assem AM, and Steven JE Wilton. "An FPGA with power-gated switch blocks." Field-Programmable Technology (FPT), 2012 International Conference on. IEEE, 2012.

[40] Drake, Alan, et al. "A distributed critical-path timing monitor for a 65nm high-performance microprocessor." Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International. IEEE, 2007.

[41] Muhtaroglu, Ali, Greg Taylor, and Tawfik Rahal-Arabi. "On-die droop detector for analog sensing of power supply noise." IEEE Journal of solid-state circuits 39.4 (2004): 651-660.

[42] Lamoureux, Julien, and Steven JE Wilton. "Activity estimation for field-programmable gate arrays." Field Programmable Logic and Applications, 2006. FPL'06. International Conference on. IEEE, 2006.

[43] Ayorinde, Seyi. "Circuit Design and Configuration for Low Power FPGAs." (2015).

[44] Murray, Kevin E., Scott Whitty, Siyuan Liu, Jason Luu, and Vaughn Betz. Titan: Enabling large and complex benchmarks in academic CAD. In Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on, pp. 1-8. IEEE, 2013.

[45] Jamieson, Peter, et al. "Odin ii-an open-source verilog hdl synthesis tool for cad research." Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on. IEEE, 2010.

[46] Zhang, Yanqing, et al. "A Batteryless 19μ W MICS/ISM-Band Energy Harvesting Body Sensor Node SoC for ExG Applications." IEEE Journal of Solid-State Circuits 48.1 (2013): 199-213.

[47] Liu, Ningxi, and Benton Calhoun. “Design Optimization of Register File Throughput and Energy Using a Virtual Prototyping (ViPro) Tool.” VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on. IEEE, 2016.

[48] Nalam, Satyanand, et al. “Virtual prototyper (ViPro): An early design space exploration and optimization tool for SRAM designers.” Proceedings of the 47th Design Automation Conference. ACM, 2010.

[49] Tang, Xifan, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. ”FPGA-SPICE: A simulation-based power estimation framework for FPGAs.” Computer Design (ICCD), 2015 33rd IEEE International Conference on. IEEE, 2015.