NORTHEASTERN UNIVERSITY
Graduate School of Engineering

Thesis Title: Design and Analysis of Minimum Energy FPGAs

Author: Peter Grossmann

Department: Electrical and Computer Engineering

Approved for Thesis Requirements of the Doctor of Philosophy Degree

---

Thesis Advisor: Prof. Miriam Leeser	Date
Thesis Advisor: Prof. Marvin Onabajo	Date
Thesis Committee Member: Dr. Steven Vitale	Date
Thesis Committee Member: Prof. Benton H. Calhoun	Date
Department Chair: Prof. Ali Abur	Date

Graduate School Notified of Acceptance:

Dean: Prof. Sara Wadia-Fascetti	Date

NORTHEASTERN UNIVERSITY
Graduate School of Engineering

Thesis Title: Design and Analysis of Minimum Energy FPGAs

Author: Peter Grossmann

Department: Electrical and Computer Engineering

Approved for Thesis Requirements of the Doctor of Philosophy Degree

---

Thesis Advisor: Prof. Miriam Leeser	Date
Thesis Advisor: Prof. Marvin Onabajo	Date
Thesis Committee Member: Dr. Steven Vitale	Date
Thesis Committee Member: Prof. Benton H. Calhoun	Date
Department Chair: Prof. Ali Abur	Date

Graduate School Notified of Acceptance:

Dean: Prof. Sara Wadia-Fascetti	Date

Copy Deposited in Library:

Reference Librarian	Date

Design and Analysis of Minimum Energy FPGAs

A PhD Dissertation

by

Peter Grossmann

Advisors:

Dr. Miriam Leeser
and
Dr. Marvin Onabajo

Dissertation Committee:

Dr. Steven Vitale

Dr. Benton H. Calhoun

ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

NORTHEASTERN UNIVERSITY

Boston, Massachusetts
2013

Abstract

Embedded systems continue to become smaller, demand greater compute capability, and target deployment in more energy-starved environments. System power budgets of less than 1 mW are increasingly common, while standby power is brought as close to zero as possible. While field programmable gate arrays (FPGAs) have historically been used as compute engines in low power systems, they have not kept pace with application-specific integrated circuits (ASICs) and microprocessors in meeting the needs of these ultra low power systems. Research in both ASICs and microprocessors has extended voltage scaling into the subthreshold region of transistor operation, sacrificing performance in exchange for dramatic power savings. For some ultra low power systems such as wireless sensor networks and implantable biomedical devices, performing a computation with minimum energy consumption rather than within a certain time frame is the goal. It has been shown that to minimize energy for ASICs and microprocessors, subthreshold operation is typically required. For FPGAs, the answer remains largely unexplored—the first subthreshold FPGA has only recently been fabricated, and minimum energy operation of FPGAs has not been thoroughly studied.

This research presents multiple steps forward in the design and analysis of FPGAs targeting minimum energy operation. A fabricated FPGA test chip capable of single-supply subthreshold operation is presented, with measurement results demonstrating

FPGA programming and operation as low as 260 mV. The capability to minimize energy per clock cycle at subthreshold supply voltages for a high activity factor test case is also shown, indicating that the flexible nature of FPGAs does not inherently prevent their energy minimum occuring below threshold. A simulation flow for performing pre-fabrication chip-level minimum energy analysis for FPGAs has also been developed in this work. By combining industry-standard integrated circuit design verification software with academic FPGA software and custom scripts, the minimum energy point sensivity of an FPGA to its programming was investigated. The FPGA was programmed with 21 different ISCAS '85 benchmarks, and a minimum energy supply voltage was estimated for each with a nominal input activity factor. The benchmarks had minimum energy points ranging from 0.42-0.54 V, or slightly above threshold. The minimum energy point was not a strong function of benchmark circuit size or input count, suggesting that the topology of the benchmark circuit influenced the FPGA minimum energy point.

Acknowledgements

First, a thank-you to my co-advisors, Miriam Leeser and Marvin Onabajo, for providing focus, keeping me on track, and showing me how to transform a fun project into actual academic research, and to Prof. Benton Calhoun from the University of Virginia for bringing his years of experience in subthreshold circuit design to my dissertation committee.

This work was funded by the MIT Lincoln Laboratory Scholars Program. Thanks to the Scholars Program committee for providing this opportunity, and to Bill Keicher in particular for his vision in making the Scholars Program what it is.

Thanks to all the others at MIT Lincoln Laboratory for the varied, enthusiastic, and continuous support provided to this research. Thanks to Steven Vitale for serving as my Lincoln Scholars mentor. Thanks to Peter Wyatt for participation in my proposal and thesis reading. Thanks to the Group 83 office (George Turner, Paul Juodawlkis, Simon Verghese, Paula Jones, Erin Jones-Ravgiala) for turning me loose to Northeastern and then still reading through all my material en route to release review. Thanks to several of the Division 8 CMOS designers (Jon Frechette, George Jordy, Jim Wey) for fielding a variety of wacky questions related to subthreshold circuit design and verification. Big thanks to Tony Soares and WeiLin Hu for their contributions and guidance in test program development for the test chip measurements described in Chapter 3. Big thanks again to LLCAD (especially Rich Ralston, Bob D'Ambra, Patrick Cable) not only for the basic support of the CAD infrastructure that all chip designers at Lincoln benefit from, but for the support of this work specifically that required extra attention in terms of license purchases and installations, disk space management, and making sure I didn't run so many simulations that I ground the rest of the lab to a halt.

Thanks also to the folks at Cadence who assisted in getting me up to speed with their software, especially Paul Osepa with Liberate and Jeff Flieder with RTL Compiler. Finally, thanks to my parents, Gary and Becky, for listening to conference talk dry runs over the phone from 3000+ miles away, and for reminding me that I'm PhD material.

Contents

List of Tables

List of Figures

1 INTRODUCTION

Field Programmable Gate Arrays (FPGAs) offer system designers a flexible, energy-efficient platform for computing that requires moderate costs in terms of designer time and hardware expense, as well as reconfigurability to adapt to diverse applications or changing environments after fabrication of integrated systems. While not as high performance or energy-efficient as application-specific integrated circuits (ASICs), nor as efficient and fast to design for as microprocessors, the advantages of cost and reconfigurability over ASICs and performance over microprocessors make FPGAs the best choice for an increasing number of applications. Currently, designers of the lowest-power systems, such as wireless sensor networks and implantable medical devices, prefer custom ASICs [1] or ultra-low power microcontrollers. While ultra-low power designers have begun to adopt the Actel Igloo FPGA [2], there are few published results using it. Comparison with microcontrollers, most notably the Texas Instruments MSP 430 shows that all but the smallest FPGAs in the Igloo family have higher static power consumption [3][4]. At the same time, the amount of logic available even on larger Igloos limits some applications [5]. A more energy-efficient FPGA is needed to improve the absolute power consumption of FPGAs, provide more logic for the same power budget, or ideally do both.

The most powerful tool for power reduction in integrated circuits is voltage scaling. Current state of the art low power FPGAs have not yet maximized the benefits of voltage scaling to achieve minimum energy operation, meaning that there is room to improve simply by designing an FPGA to operate at a lower supply. Figure 1.1 shows energy consumption per operation trends for ASICs as a function of supply voltage. As the supply voltage is reduced, the time required to perform a given operation increases, increasing the amount of static energy consumption associated with that operation. However, the dynamic energy decreases. When the dynamic energy savings no longer make up for the static energy penalty an optimal supply voltage, or minimum energy point, is reached. The minimum energy point is the desired operating point of a circuit to maximize energy efficiency. Currently FPGAs (and indeed, most ASICs) do not operate at their minimum energy point.

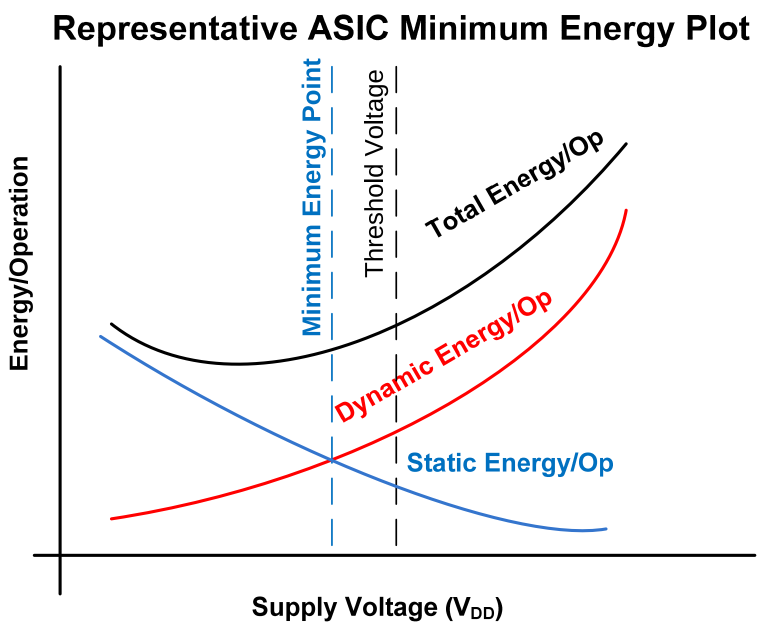


Figure 1.1 Hypothetical minimum energy analysis plot

Research in ASICs has shown that while the minimum energy point of a circuit is dependent upon the details of the circuit itself and its activity factor, the minimum energy point obtained for a given circuit is generally near or below the threshold voltage of the transistors [6]. While subthreshold transistor operation yields minimum energy operation, there are tradeoffs: the exponential relationship between current and voltage below threshold results in drastic increases in circuit delay and vulnerability to process, voltage, and temperature variations. The increased variability of subthreshold circuits restricts which circuit topologies can be used; tall series stacks and ratioed circuits in particular must be avoided [7].

Research in subthreshold circuit design and minimum energy operation of FPGAs is much less mature than that of ASICs. Only one fabricated subthreshold FPGA has been previously demonstrated [8]. Meanwhile, the implications of FPGA's reconfigurability on their energy efficiency remain unstudied. The research in this dissertation presents progress in both the design and analysis of FPGAs for minimum energy operation with the following contributions:

- **Study of multiplexer design choices for subthreshold operation of unidirectional FPGA routing fabrics**: Although prevalent in commercial FPGAs, unidirectional routing fabrics have not been previously considered for subthreshold FPGAs. After evaluating several multiplexer options for speed, area, power, and robustness to process variation, this work shows that static CMOS multiplexers are the best all-around choice for use in subthreshold FPGA unidirectional routing fabrics.

- **Replacement of SRAM with latches for low voltage FPGA programming without write assist circuitry:** SRAM is widely used as the storage element for FPGA configuration bits. Operating SRAM at subthreshold requires additional circuitry to address stability issues caused by signal contention during writes and reduced noise margin during reads. While the read operation is important for FPGA use of SRAM cells, write stability is the higher priority. The large number of configuration bits required for FPGAs makes adding circuitry to assist with write stability undesirable. This work shows that by using latches instead of SRAM for configuration bit storage, low voltage operation can be achieved without such circuitry.

- **Measurement results for an FPGA test chip showing low voltage operation with a single, subthreshold supply**: this work presents a fabricated test chip in the IBM 0.18 μm SOI process. The FPGA can be programmed and operate as low as 0.26 V by using the circuit design choices described above.

- **Measurement results showing that FPGA activity factors can be made high enough to bring the minimum energy point below threshold:** Previous low voltage measurement results for FPGAs left uncertainty as to whether the inherent energy inefficiency of FPGAs reconfigurable resources permitted minimum energy operation to occur at subthreshold supply voltages as they do for ASICs. In this work, measurement results show that for sufficiently high activity factors, an FPGA can have a subthreshold minimum energy point.

- **Definition of a technique for a simulation-based, chip-level minimum energy analysis of FPGAs:** Minimum energy analysis is typically conducted at the chip level on fabricated chips or at the block level using SPICE simulations. While commercial ASIC design and verification tools provide a straightforward path for performing simulation-based minimum energy analysis at the chip level for ASICs, taking the same approach for FPGAs is complicated by the need to integrate FPGA CAD tools into the analysis flow. This work presents a technique based on combining Cadence software for ASICs and the Verilog-to-Routing (VTR) academic tool flow from the University of Toronto for FPGAs with custom scripts to unify timing and energy consumption data across both tool sets in support of minimum energy point estimation. This is the first such effort of its kind.

- **Simulation results for the first chip-level multi-benchmark study of FPGA minimum energy operation:** Previous energy consumption studies of

FPGAs have either avoided calculation of the minimum energy point, reported results only at the block level, or reported results only for single benchmark circuits. In this work, simulations are conducted at the chip level specifically to investigate the minimum energy point sensitivity to FPGA programming. To do this, power-delay product vs. supply voltage plots are generated for 21 ISCAS '85 benchmarks mapped to an FPGA designed for subthreshold operation. It is shown that the minimum energy point is dependent upon benchmark circuit topology as well as FPGA architecture and utilization, and that the minimum energy point varies by 0.12 V across the benchmarks tried.

To provide context, Chapter 2 provides background on subthreshold circuit design, an introduction to FPGA architecture and CAD tools, and a summary of low power FPGA research up to and including recent research in subthreshold FPGA operation. Chapter 3 presents the subthreshold FPGA test chip fabricated for this work and the simulation studies that influenced its design; the architecture choices made for the test chip; and the measurement results described above. Chapter 4 presents the technique used to perform minimum energy point analysis, the architecture used in analysis, and the results of analyzing 21 benchmark circuits mapped to that architecture and simulated across a range of near- and subthreshold supply voltages. Chapter 5 summarizes the key results obtained and proposes directions for future work.

2 BACKGROUND

Embedded systems typically employ one or more of three types of resources for computing: application-specific integrated circuits (ASICs), microprocessors, and field-programmable gate arrays (FPGAs). Table 2.1 shows the key tradeoffs between these types of resources. ASICs are the most energy-efficient by both relevant metrics: speed and energy consumption. However, they are the most expensive, least flexible, and require the most designer effort to produce. Microprocessors and microcontrollers require the least development time, are the least expensive on average, and offer great flexibility, but offer the least performance. This performance penalty limits their potential for energy efficiency. FPGAs offer a middle ground between ASICs and microprocessors for development time, performance, and cost. Their reconfigurable hardware resources arguably make them even more flexible than microprocessors.

Table 2.1. Tradeoffs between ASICs, Microprocessors, and FPGAs

	ASIC	Microprocessor	FPGA
Performance (operations/time)	Very High	Moderate	High
Energy Consumption	Low	High	Moderate
Cost	Very High	Low	Moderate
Development Time	Very High	Low	Moderate
Flexibility	Low	High	Very High

The tradeoffs described above have typically been evaluated in contexts where energy consumption, no matter how important it is in an absolute sense, remains less important than performance. For an emerging class of embedded systems, reducing energy consumption is a higher priority than performance. This stems from strict power budgets enforced by the one or more of the following: long battery lifetime requirements, limited battery capacity due to size and weight requirements, and/or the goal of operating on energy that can be harvested from the system environment [9]. In many cases, energy-constrained systems are able to prioritize energy consumption over performance because performance requirements are relaxed—slack exists in the system latency budget, enabling the system to either run at reduced clock frequency or to power down while not in use. In other cases, system design choices are made in which compute capability is sacrificed in order to meet energy requirements. For example, in remote sensing applications, a sensor node might be forced to perform less on-sensor data processing or collect less total data in order to meet a battery lifetime requirement. Similarly, an implantable computing application becomes limited in the power that can be expended performing local computations by the small power budgets imposed by strict thermal requirements.

The desire to bring greater compute capability to energy-starved locations such as those above has motivated research into minimum energy computing paradigms. This represents a subtle, but important, shift in approach from low power computing. Design for minimum energy consumption presupposes that there is no strict latency requirement that must be met, and instead assumes that system operation is dictated solely by finding the operating point that minimizes total energy consumption for a given task. Traditional low power design, meanwhile, seeks to minimize computational power consumption under the constraint of a specific performance goal. The relationship between these two paradigms can be captured in a surprisingly simple way: whereas low power designers frequently evaluate their designs in terms of operations per watt, minimum energy designers frequently evaluate their designs in terms of joules per operation.

In theory, a shift from low power computing to minimum or near-minimum energy computing should not disrupt the relationship between ASICs, microprocessors, and FPGAs described in Table 2.1. Before this comparison can be extended to minimum energy operating conditions, minimum energy FPGA research must catch up to its counterparts in ASICs and microprocessors. To do this, additional FPGA research is also required to catch up to ASICs and microprocessors in extending voltage scaling techniques to subthreshold supply voltages. While subthreshold ASIC research is becoming somewhat mature, and subthreshold microprocessors have been demonstrated [10], the first complete, fabricated subthreshold FPGA was presented only recently [8]. In terms of minimum energy analysis, numerous ASIC results have been published [11], a subthreshold microprocessor has reported a minimum energy point below threshold [10], while the only minimum energy point reported for FPGAs was performed on a design not optimized for low voltage operation, and the minimum energy voltage obtained was well above threshold [12].

The following sections provide an overview of the topics for which good understanding is required to investigate FPGA operation at subthreshold supply voltages and to determine an FPGA's minimum energy point. Section 2.1 introduces subthreshold circuit design in terms of the issues most relevant to FPGAs: digital logic gate design, SRAM design, and coping with subthreshold sensitivity to process variation. Section 2.2 introduces FPGA architecture and CAD tools. Section 2.3 summarizes previous work in low power FPGA design at nominal (or "superthreshold") supply voltages, work in subthreshold FPGA design, and previous studies of minimum energy operation of ASICs and microprocessors.

2.1 Subthreshold Circuit Design

The design of digital circuits that can operate at subthreshold supply voltages is now fairly well understood. Recent publications have established good rules of thumb for coping with the reduced noise margins and increased sensitivity to process variation present in subthreshold operation [7]. By avoiding circuits that rely on ratioed transistor sizes, use tall series stacks, or contain parallel off transistors that lack corresponding on transistors (e.g. tiny XOR), CMOS digital logic can be made to reliably operate at supply voltages below 0.3 V in a variety of process technologies.

The key building blocks for implementing FPGA configurable logic blocks (CLBs)—SRAM, multiplexers, and flip-flops—have all been demonstrated at subthreshold supply voltages. The possible building blocks for FPGA routing resources—pass transistors, transmission gates, and tri-state buffers—have also been considered, along with other basic combinational gates needed to form a standard cell library. In addition to verification at the circuit level, subthreshold integrated circuits performing complex functions have been successfully fabricated, giving ample evidence that a well-designed FPGA architecture should be successfully scalable to full-chip dimensions. The Subliminal processor was implemented in 0.13 μm bulk CMOS, demonstrating correct operation at a minimum supply voltage of 0.2 V [10]. More recently, Intel demonstrated an IA-32 processor operating as low as 0.28 V [13]. Other reported subthreshold computing efforts include a 16-bit microcontroller based on the MSP430 architecture [14], a 16-bit 1024-point FFT [11] and a MAC [15].

The task of the subthreshold FPGA designer is thus to select which implementation of each building block is best suited for a given set of design goals in terms of area, power, and speed. Beyond those metrics that are common to superthreshold circuits, subthreshold circuit designers also characterize robustness of the circuit to process variation more carefully. For digital circuits, this is often quantified by reporting the ratio of standard deviation to mean of propagation delay and (for sequential circuits) setup and hold times. Subthreshold circuit designers also often report the minimum supply voltage at which the circuit functions correctly and the supply voltage at which the energy efficiency is maximized, or the minimum energy point. In general, the minimum energy point is not the minimum supply voltage for correct operation because the increase in leakage power resulting from reduced supply voltage overcomes the reduction in dynamic power [6]. These metrics should be used in evaluating subthreshold FPGA components.

Among FPGA building blocks, subthreshold SRAM design in particular has received a great deal of attention and will be discussed first. The key results for successful combinational and sequential logic design will be described second in the context of standard cell libraries. Finally, the high impact of process variation on subthreshold circuit design is addressed.

2.1.1 Subthreshold SRAM Design

Because reduced supply voltage hurts signal-to-noise margin and write stability in SRAM [7], several alternatives to the traditional six-transistor (6T) SRAM bit cell designs have been considered for subthreshold circuits. Figure 2.1, Figure 2.2, and Figure 2.3 show a representative sample of designs using six [16], eight [17], and ten [18] transistors. A conventional 6T SRAM cell is provided for reference in Figure 2.4.

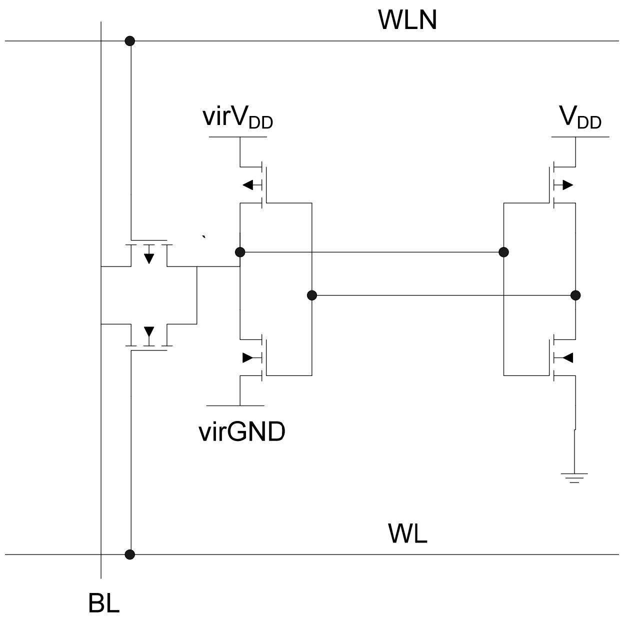


Figure 2.1 Single-ended 6T SRAM bit cell

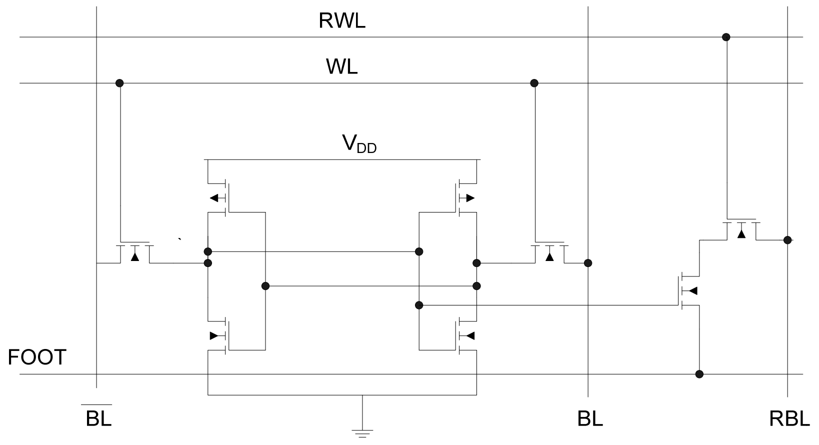


Figure 2.2 8T SRAM bit cell

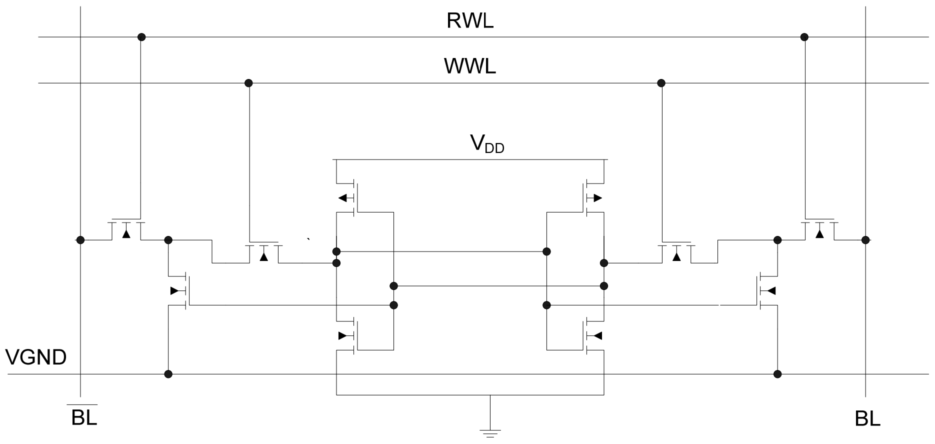


Figure 2.3 10T SRAM bit cell

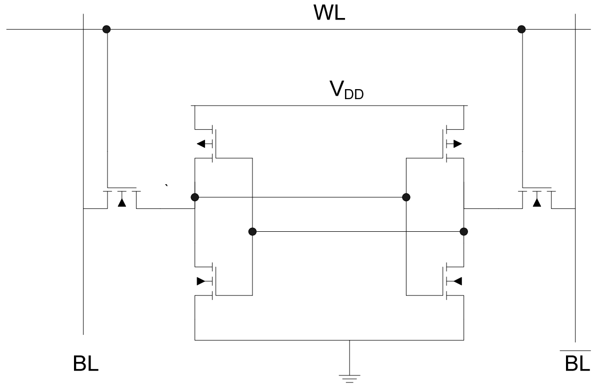


Figure 2.4 Conventional SRAM bit cell

Each of the above designs makes a particular tradeoff to try to improve upon the 6T cell for subthreshold operation. The 8T cell in Figure 2.2 isolates read operation from writes to address write stability [17]. The single-ended bitline of the 8T cell comes with a delay penalty for reads in addition to the area penalty of two extra transistors [19]. The 10T cell addresses the read penalty by isolating reads and writes in a complementary fashion at the expense of area. The single-ended 6T cell attempts to recoup this area while still maintaining robust reads and writes. Read robustness is obtained by use of the full transmission gate between the bitline and cross-coupled inverters [16]. The supply terminals of one of the cross-coupled inverters are gated to prevent contention during writes; the gated supplies are denoted as virVDD and virGND in Figure 2.1.

Although a number of SRAM design alternatives have been proven for subthreshold operation, none of these circuits has targeted use for FPGA lookup tables or configuration bits. The SRAM design for FPGA lookup tables has the benefit that it is not necessary to globalize read access, since SRAM is used only for LUT and configuration bit storage that is localized to each CLB. Write margin is thus more important for FPGAs than read stability, and both problems are less challenging due to the absence of large shared word lines and bit lines. Design of FPGA configuration bit storage cells for subthreshold operation will be addressed in more detail in Chapter 3.

2.1.2 Subthreshold Standard Cell Design

Standard cell design has been investigated down to near the theoretical limits of supply voltage scaling. Fabrication results have shown ring oscillator functionality down to supply voltages of 0.067 V and three-input NAND and NOR gates to 0.1 V [20]. Complete standard cell libraries have been developed and used in circuits operating at 0.3 V [14][7].

Evaluation of a conventional superthreshold standard cell library for use at subthreshold supply voltages in bulk CMOS is presented in [7]. It was found that with no design changes, the library cells had minimum operating voltages of 0.3-0.4 V across process corners. For combinational cells that failed at higher operating voltages, multiple devices operating in series or in parallel were found to be limiting factors. Excluding standard cells with tall series stacks and redesigning other cells to eliminate excessively parallel paths resulted in a reduced standard cell library with all cells having minimum operating voltage of 0.3 V or less across process corners.

Tradeoffs in standard cell design for partially-depleted SOI have also been analyzed [21]. Based on inverter simulations, traditional CMOS was shown to outperform dual-threshold MOS in both speed and energy consumption. Pseudo-NMOS and dual-supply inverters were also investigated and showed static energy consumption 100-1000X higher than single-supply static CMOS. Transmission gate and NAND-NAND multiplexers were compared in terms of energy and delay and found to have similar results, with the transmission gate delays proving more symmetric. Taking the results in sum, it was concluded that a conventional static CMOS standard cell library provided the best all-around power and performance for subthreshold SOI design.

It has been demonstrated that effective combinational logic can be implemented for subthreshold circuits using traditional approaches [7][21]. A subthreshold FPGA architecture may therefore make use of such cells to implement logic as needed in support of lookup tables and interconnect, such as buffers and multiplexers. However, less robust circuits that may still work at superthreshold supplies are not necessarily portable to subthreshold. Additionally, process variation must be accounted for as noted in section 2.1.3.

A variety of flip-flop architectures have been evaluated for use in subthreshold operation. The standard cell library evaluation in [7] showed that flip-flops which rely on ratioed feedback are less robust than those using gated feedback, requiring larger supply voltages to work correctly across process corners. In SOI technology, a comparison of a static and dynamic flip-flop at 0.35 V found that the dynamic flip-flop had superior setup time at the expense of increased propagation delay, with a slight energy benefit [21]. Minimum operating voltage and variation were not studied. Simulations of a static flip-flop, a sense-amplifier flip-flop, and a dynamic C^2MOS flip-flop in both 90 nm and 65 nm processes demonstrated that the tradeoffs in selecting a flip-flop are process-dependent, with the static flip-flop having the overall lowest minimum operating voltage and best delay at 65nm, but the worst overall delay at 90 nm [22]. The dynamic flip-flop had the best delay characteristics at 90 nm but the largest minimum supply voltage at both nodes. A flip-flop architecture based upon gate-diffusion input (GDI) multiplexers in a 90 nm process has also been proposed [23]. The design consumed only 180 pW at 0.2 V vs. 3.3 nW for the 90 nm static flip-flop in [22], but clock-to-Q delay was 201 ns vs. 24.3 ns for the static flip-flop.

As with combinational logic, the body of work performed to date suggests that although there is not a need for novel subthreshold-specific cell design, there is a need to investigate different flip-flop architectures' effectiveness in a given process and evaluate which flip-flop is best suited to the process and the goals of the design in terms of operating voltage, delay, and energy efficiency. In selecting a flip-flop for use in a subthreshold FPGA, a flip-flop that has a good balance of delay characteristics, energy consumption, and area is needed since all three are crucial to low power FPGA design and often represent conflicting design goals. While the studies above suggest that for a specific FPGA implementation a novel flip-flop may sometimes be best, flip-flops account for only a tiny fraction of FPGA area and power consumption, so the impact of a novel flip-flop on the FPGA as a whole is limited. Throughout this work, a static D flip-flop with gated feedback will be used in order to permit additional focus on other design considerations.

2.1.3 Subthreshold Circuits and Process Variation

Trends in CMOS process technology scaling have shown that as transistor lengths have decreased, device performance sensitivity to process variation has increased [24]. A particularly important cause of this for subthreshold circuits is the impact of random dopant fluctuation (RDF), which can cause variation in transistor threshold voltage [25]. The random nature of these variations necessitates the use of Monte Carlo simulations during subthreshold circuit design to estimate the range of possible values for delay and power consumption that a circuit might exhibit. Where simulation runtime makes Monte Carlo simulation prohibitive, process corner models should be used to obtain upper and lower bounds on delay and power consumption. Both approaches are used in this work.

Increased process variation sensitivity with technology scaling has been studied specifically in the context of subthreshold circuit design. Calhoun et al., using predictive technology models (PTMs) tuned for low power processes, reported a 5X increase in threshold voltage variability when scaling from 90 nm to 22 nm [26]. This variability in turn reduces noise margins for digital circuits, decreasing their energy efficiency and increasing their minimum operating voltage. Calhoun et al. reported a decrease in minimum operating voltage of 50 mV across all nodes when increasing lengths to 1.3X their minimum value [26]. The improvement in minimum operating voltage corresponds to improved noise margins and robustness against process variation, while improving net energy efficiency. Bol et al. reported a maximum energy efficiency improvement for an 8x8 multiplier of 5X when increasing effective minimum transistor length at the 32 nm node by 58% [27].

2.2 FPGA Architecture and CAD

Successful design of an FPGA requires a broad approach. A high-level understanding of FPGA architecture, and the CAD tools that facilitate their use, is as important as understanding transistor-level design of FPGA building blocks. The following sections provide an introduction to the architecture of the island-style FPGAs used in this work, brief descriptions of some of the alternatives to island-style that have been considered previously, and an overview of the required functions of FPGA CAD tools and the FPGA CAD flow used in this work.

2.2.1 Island Style FPGAs

Among commercial FPGAs, the most prevalent architecture is island-style. As such, it is the logical choice for a first architecture to implement using subthreshold logic. Figure 2.5 shows a representative block diagram of an island-style FPGA. The array element of an island-style FPGA is frequently called a tile; each tile is outlined with dashed lines in Figure 2.5. Each tile contains a configurable logic block (CLB), shown as a red box, and two types of routing resources: connection boxes (shown in blue in Figure 2.5) consist of those resources that select what signals are routed to CLB inputs; switch boxes (shown in purple in Figure 2.5) consist of all other routing resources and define the routing channels that connect adjacent tiles.

Each CLB is a self-contained configurable device. In a simple architecture, the CLB could be nothing more than a single basic logic element (BLE) consisting of a lookup table (LUT), implemented using SRAM, plus a flip-flop and a multiplexer [28]. The lookup table may then be programmed to compute an arbitrary logic function of K inputs. Figure 2.6 shows a block diagram of a BLE for K=4.

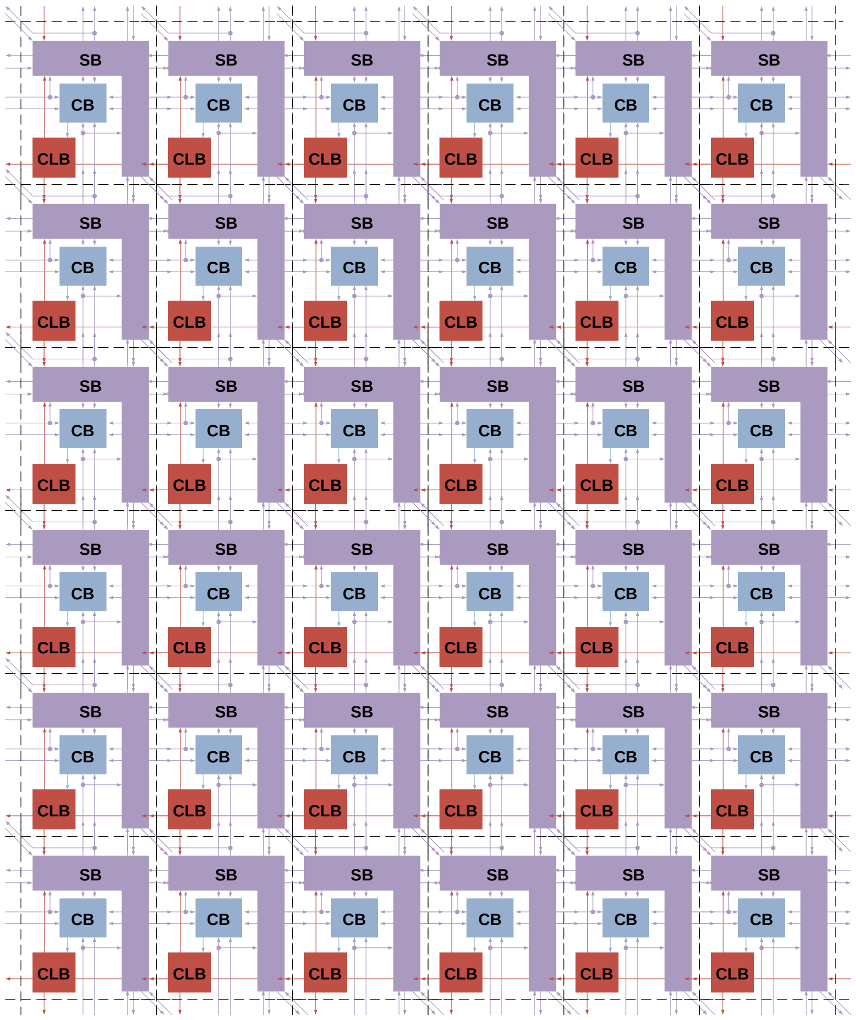


Figure 2.5 Representative block diagram of an island-style FPGA.

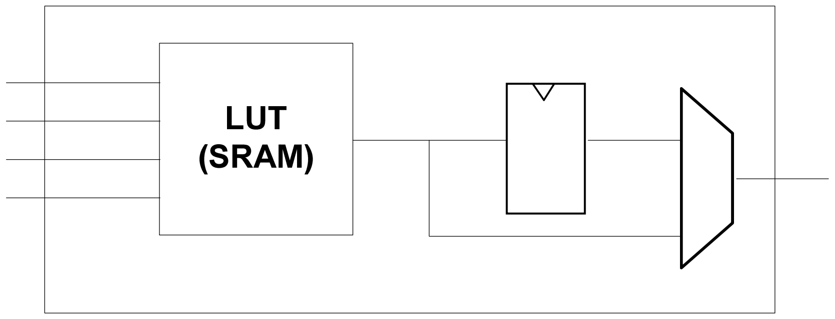


Figure 2.6. Hypothetical FPGA basic logic element BLE [28].

As island-style architectures have grown in complexity, the CLB's complexity has grown as well. Support for larger logic functions might be added. Special features that accelerate popular logic functions, such as paths intended specifically to chain together adjacent BLEs for fast addition, might also be included [28]. To obtain better routing solutions, a CLB is often organized as a cluster of BLEs that share local routing resources. Figure 2.7 shows a clustered CLB style commonly used in academia. Here, no special functions are provided. A group of BLEs shares a common pool of inputs through a programmable switch network called a crossbar. In general, a crossbar may permit any CLB input to connect to a given BLE (called a full crossbar), or a CLB input may only be routable to a subset of the BLEs. The common pool of inputs includes the BLE outputs. This enables the implementation of circuits in which registered outputs feed backward, such as finite state machines. The number of inputs to the CLB can be made independent of the number BLEs or each BLE's input count, although studies have shown that a number of inputs $I = 2*N + 2$, where N is the number of BLEs in the cluster, is a good choice [29].

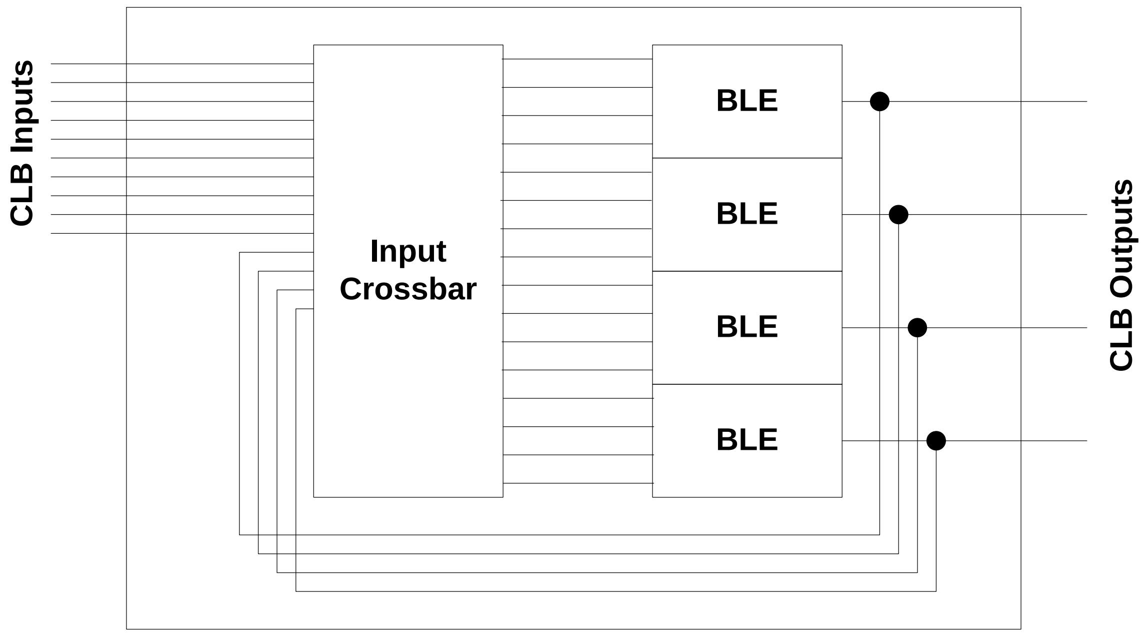


Figure 2.7 Clustered CLB architecture.

Both connection boxes and switch boxes consist of programmable switch networks defining whether a connection between two or more wires is open or closed. These switches can either be bidirectional, permitting any wire to be the signal source when the switch is closed, or unidirectional, defining a notion of which wire connected to the switch is an input and which wire is an output.

For bidirectional routing resources, a fundamental switch network element is the programmable interconnect point, or pip [28]. In switch boxes, a common pip implementation consists of six switches that can be configured to connect any or all of four intersecting wires, as shown in Figure 2.8. A collection of multiple pips forms a complete switch box. The switches in each pip can be implemented with a single pass transistor to save area; if more electrical robustness is needed, transmission gates may be used. The transistor gates are controlled with configuration bits that determine whether each is on or off. In connection boxes, a pass transistor or transmission gate can be placed between a CLB input and any wire to which the CLB input is allowed to connect.

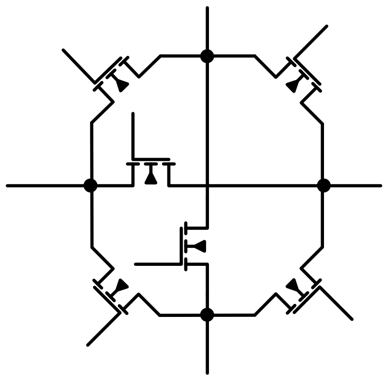


Figure 2.8. Programmable interconnect point (pip)

For unidirectional routing resources, the fundamental switch network element is the multiplexer. Configuration bits determine the value of the select bits of the multiplexers in the network. A connection between wires in the switch network is defined by which input of the multiplexer is selected as the output. The number of choices available to produce the output determines the size of the multiplexer required. With this approach, multiplexers of the desired sizes can implement switch boxes and connection boxes of arbitrary dimensions and connectivity. As a concrete example, a switch box unit cell equivalent to the pip in Figure 2.8 is shown in Figure 2.9. It consists of four 4:1 multiplexers, each of which may select any of the four inputs to the unit cell. Each 4:1 mux produces an output, meaning that the unidirectional unit cell has twice as many total I/O as its equivalent pip. It was shown by Lemieux et al. that FPGA CAD tools can exploit this I/O increase and compensate for the extra complexity required to implement a particular switch network with unidirectional signal flow [30].

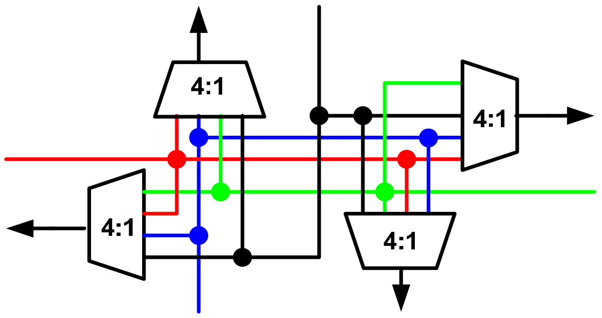


Figure 2.9. Unidirectional switch box unit cell

For a simple architecture, an island-style FPGA thus requires relatively few circuit-level components: SRAM for lookup tables and programming bits, flip-flops and multiplexers for CLBs, switches for programmable interconnect points, and any additional gates needed to implement control circuits for programming. Provided that these circuits can be made to function at subthreshold supply voltages, a subthreshold FPGA is feasible.

2.2.2 Alternative FPGA Architectures

Numerous alternatives to island-style FPGAs have been considered. In Hierarchical Synchronous Reconfigurable Arrays (HSRAs), the routing resources are organized in a tree structure rather than a grid [31]. The Triptych architecture [32] integrates local routing resources with logic resources for horizontal communication, augmented by segmented vertical routing channels for global routing. The integration of routing resources with logic resources increases circuit density for a given array size, with a tradeoff that some array elements must be used for routing only. RaPiD is an example of a coarse-grained architecture, in which the logic resources are comprised of 16-bit ALUs, multipliers, and memories rather than lookup tables [33]. Rothko exploits three-dimensional fabrication technology, consisting of multiple tiers of Triptych-style logic blocks with local routing connections between adjacent tiers [34].

These four examples provide a glimpse at the variety of design tradeoffs in FPGA architecture that have been researched. A complete architecture study, however, is beyond the scope of this work, and henceforth only island style FPGAs will be considered.

2.2.3 FPGA CAD Tools

The CAD tools used to program FPGAs have a level of sophisitication comparable to those used in designing ASICs. This stems from their need to perform many of the same basic functions as ASIC software tools in order to map a design to an FPGA: synthesis, logic optimization, timing anlaysis, placement, and routing. In addition to these mapping features, it is also necessary to translate the placement and routing information into a bitstream containing the explicit configuration bit settings required to program the FPGA with the mapped design. Figure 2.10 shows a CAD flow for these steps based upon the Verilog-to-Routing (VTR) CAD flow used in this work and throughout academia. The VTR flow contains three separate software components. The first tool in the flow is ODIN II, a logic synthesis tool [35]. ODIN II accepts a Verilog HDL description of a design as input and produces a Berkeley Logic Interchange Format (BLIF) netlist as output. The BLIF netlist format is particularly useful for FPGAs as it organizes a design into a set of K-input, one output truth tables that can be directly mapped to FPGA lookup tables if the BLIF truth tables have input counts less than or equal to the input count of the FPGA lookup tables. Proper use of ODIN II ensures that this is always the case.

The second tool in the VTR flow is ABC, used for logic optimization [36]. ABC accepts a BLIF netlist and configuration script as inputs and performs a set of logic optmization routines defined in the configuration script to optimize the number of lookup tables required to implement the design described in the BLIF netlist. For the benchmark circuits used in this work, a 50% reduction in lookup table count compared to ODIN II output was not unusual.

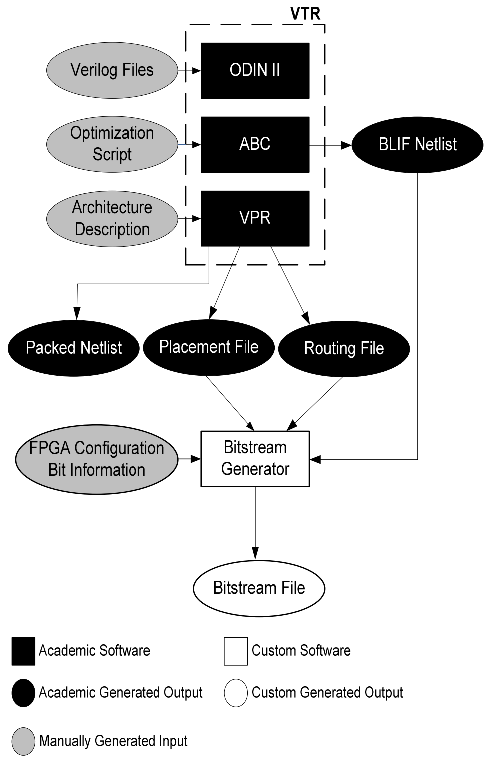


Figure 2.10. Verilog-to-Routing (VTR) CAD flow

VPR (Versatile Place and Route) 6.0 [37] is the heart of the VTR flow and produces placement and routing solutions for the optimized BLIF netlists produced by ABC. VPR accepts as input an XML-based architecture file as input along with the BLIF netlist. The architecture file permits nearly arbitrary logic function descriptions and enables them to be paired with a few types of island-style routing architectures of arbitrary routing channel width. This enables VPR to be compatible with most FPGA architectures whose routing resources match the supported types. For architectures in which multiple lookup tables are clustered into a single CLB, as described in section 2.2.1, VPR performs a packing step prior to place-and-route that assigns groups of lookup tables from the BLIF netlist to the same CLB in a manner intended to minimize the routing required between adjacent tiles. The placement and routing solutions are provided in output text files that can be easily parsed for post-processing if needed.

The VTR flow does not contain a solution for bitstream generation because the architecture file provided to VPR does not presume a particular hardware implementation for the architecture it describes. Since the details of the bitstream are hardware-specific, each particular implementation of a VPR-compatible FPGA architecture must have a software solution for converting the VPR place-and-route solution into the correct bitstream for that implmentation. A bitstream generator developed as part of this work is shown in the sample CAD flow in Figure 2.10; its implementation is described in section 4.5.1.

2.3 Related Work

The programmable nature of FPGAs encourages a holistic approach to low power design. Because the logic and routing resources are generic, it is worthwhile to optimize them at the circuit level as much as possible, and to make use of the best available process technology. Just as it is necessary to provide FPGA users with as much capability to optimize for speed and area as possible, it is also necessary to provide the means to optimize their designs for low power consumption. This comes in the form not only of circuit and architecture features amenable to low power design, but CAD tools that are able to exploit the hardware features provided. A recent overview of low power design techniques for FPGAs expresses this multi-level approach, dividing a large body of low power FPGA research between process, circuit, architecture, system, and CAD [38].

Although relatively little has been published discussing subthreshold FPGAs, a fabricated subthreshold FPGA has been presented, and the overall body of work reflects a growing interest in this topic. By considering this work in conjunction with the considerable body of low power FPGA research, an overall approach to energy-efficient FPGA design can be taken. An overview of the techniques employed to date in superthreshold FPGAs follows, with particular attention to those techniques most relevant to implementing an ultra-low voltage FPGA.

2.3.1 Superthreshold Low Power FPGAs

Commercial FPGA vendors now offer FPGAs targeting low power applications, and the power gap between such offerings and ultra low power system requirements is shrinking. The Actel Igloo family of FPGAs shows promise for ultra low power use. It offers a FlashFreeze standby mode, with typical power dissipation in this mode of 5 μW in a chip offering 30,000 equivalent gates and 768 flip-flops [3]. The Igloo nano product line extends this feature to 2 μW for a version advertising 10,000 equivalent gates and 260 flip-flops [39]. A similar product family offered by Lattice Semiconductor, the iCE40 LP FPGA, offers greater logic resources (1280-7680 LUT/flip-flop pairs) but has significantly higher static power consumption when idle—48-192 μW depending on the LUT count [40]. These standby power figures are critical for many ultra low power systems, which can spend well over 90% of their time in standby. These chips also permit users to exploit voltage scaling by offering I/Os operating as low as 1.2V. While these products' power profiles suggest sub-milliwatt average power consumption is attainable, it is clear that a tradeoff still exists between resource amount and static power that may not satisfy the most demanding system requirements.

An early academic endeavor in fabricating a low power FPGA was LP_PGAII, a low voltage FPGA in a 0.25 μm process [41]. In this chip, an architecture with five-input, three-output CLBs consisting of four 3-input LUTs was implemented. This topology was chosen for its energy efficiency in implementing both random and arithmetic circuits. The routing fabric consisted of three levels of hierarchy—nearest neighbor connections for the lowest level, switchboxes for the intermediate level, and a tree structure for global routes. Another key feature of this chip was its operating voltages of 1.5V for logic resources and 0.8V for low-swing interconnect and clock resources. Both of these voltages are well below the standard operating voltage for 0.25 μm technology.

A self-synchronous FPGA was designed for high-speed, energy-efficient superthreshold operation [42]. The minimum energy point and minimum operating voltage were found to be 0.6 V and 0.37 V, respectively [12]. The minimum energy point of this circuit is hundreds of millivolts higher than many others reported for ASICs [11][43]. These results indicate that the FPGA was not optimized for low voltage operation, although the minimum operational V_{DD} reported is below threshold.

Although the FPGA designs described above are low voltage, they do not yet push the limits of voltage scaling. Design efforts that do are described next.

2.3.2 ASIC Minimum Energy Research

Minimum energy operation of ASICs has been studied from several angles, providing a number of useful lessons learned for performing similar studies on FPGAs. Work by Calhoun et al. captures several of these, developing an equation for computing the minimum energy point of a circuit assuming that that point occurs below threshold, investigating sizing choices for minimum energy operation and concluding that minimum size transistors are usually best, and validating these efforts with both simulation and measurements of a fabricated test chip [44]. Additional work in modeling minimum energy point has been performed by Blaauw and Zhai, showing the influence of process variation on minimum energy point and the dependency of minimum energy point on activity factor and logic depth [6]. Bol et al. introduced the concept of practical minimum energy; that is, the realistic minimum energy point that can be achieved in the presence of process variation [45]. With this as the revised metric, sizing strategies were evaluated on an 8-bit multiplier to show the benefits of increased transistor length to minimize practical energy.

In addition to simulations performed in the work above, work focusing purely on the analysis of specific circuits has also been performed. Andersson et al. explored the dependence of activity factor on the minimum energy point of a datapath consisting of multiple adders and multipliers, using gate-level Verilog simulation to show a range of minimum energy points of over 0.1V [46]. Mishra et al. performed minimum energy analysis to demonstrate energy efficiency improvement for an ultra-low power adder design over a conventional design using SPICE simulations [47]. Kim and Agrawal simulated a number of ISCAS '85 benchmarks to find the minimum energy point improvement that can be obtained by using a second supply voltage to obtain further energy improvement on non-critical timing paths without loss of performance for the circuit as a whole [48]. These efforts are similar to the FPGA minimum energy operation investigation that will be presented in the next section, and each contain elements of the minimum energy analysis approach that will be presented in Chapter 4.

At the chip level, fabricated test chips have reported minimum energy points determined through measurements. The Subliminal processor reported minimum energy point at 0.36 V [10], while the FFT processor in [11] attained minimum energy operation at 0.35 V. These results are consistent with model predictions that the minimum energy point is usually below threshold. A notable absence from the chip level efforts is the attempt to perform larger-scale simulations to investigate chip-level minimum energy point prior to fabrication. This will be investigated in Chapter 4.

2.3.3 Previous Work in Subthreshold FPGA Design

The first demonstration of configurable logic at subthreshold supply voltages was a programmable logic array (PLA) rather than an FPGA. The PLA in [49] implemented a BPSK transmitter. Body biasing and synchronization to a signal called a beat clock were employed to mitigate process variation. However, this circuit did not address the problem of end-user reconfigurability.

Several papers show circuits for use in a subthreshold FPGA. A recent study specifically investigated FPGA building blocks for subthreshold in a Berkeley Predictive Technology model of a 22nm technology [50]. A 4-LUT with multiplexer-based output selection and a full adder were simulated across a range of temperatures, threshold voltage variations, and channel lengths at supply voltages ranging from 0.15 V to 0.35 V. Both cells showed delays in the tens to hundreds of nanoseconds depending upon operating conditions. A subthreshold flip-flop design implemented in SOI and targeting reconfigurable computing has also been reported [51], as has a level shifter circuit implemented in SOI for translating between subthreshold and superthreshold logic levels for I/O blocks [52].

Sankaranarayanan and Guthaus presented a minimum energy analysis of a theoretical FPGA based upon scaling up block-level SPICE simulations of CLB, switch box, and connection box components designed in the TSMC 0.18 μm process to obtain power estimates and using VPR for critical path estimation [53]. Timing information was provided to VPR from delay calculations performed in the SPICE simulations. A complete FPGA design was not presented or simulated. Power delay product vs. voltage and critical path delay vs. voltage were provided for a single ISCAS '85 benchmark, noting that other benchmarks tried produced similar results. The input activity factor used in the block-level simulations was varied and from 0.25-1.0. A minimum energy point was obtained at 200 mV, reflecting the older process technology and the high activity factors used. Process variation was not addressed.

Calhoun et al. discuss several key concerns for subthreshold FPGA design and show simulation of a more complex FPGA implementation [54]. From these results, an FPGA test chip was fabricated containing two architecture variants; one with tri-state buffer-based interconnect, and one with pass transistor interconnect using low-swing, dual-V_{DD} signaling and asynchronous sense amplifiers to mitigate the concerns described above [8]. The design with low-swing interconnect consumed 4.7X less energy at constant delay than the design with conventional interconnect, and 22X less energy than the conventional design with V_{DD} set to the 90 nm process standard value of 1.2V. The static power consumption for a representative benchmark using 780 out of 1134 LUTs was approximately 30 μW at 0.4V. The design without low-swing interconnect contained only 912 LUTs and consumed approximately 20 μW of static power for the same benchmark circuit. The fraction of total power consumption that this static power represents was not reported.

Other work has considered subthreshold FPGA design in conjunction with related goals. Subthreshold FPGA operation was considered as part of a study on variation-aware CAD for energy-efficient FPGAs [55]. A subthreshold FPGA was also recently used as a demonstration vehicle for graphene interconnect design [56]. These efforts add to a rapidly growing body of work geared toward pushing FPGA voltage scaling to its limits.

3 DESIGN OF A SUBTHRESHOLD FPGA TEST CHIP

This chapter presents a pair of subthreshold FPGA test chips designed in the IBM 0.18 µm SOI process. The goals of these test chips were:

- Demonstrate low voltage subthreshold operation of the components required to implement a conventional clustered, island-style FPGA with unidirectional routing fabric.
- Select the best all-around two-input multiplexer for a subthreshold FPGA, balancing area, delay, power consumption, and robustness.
- Choose a robust, area-efficient alternative to the conventional 6T SRAM bit cell for configuration bit storage.
- Investigate minimum energy operation of FPGAs, and whether the FPGA minimum energy point can occur below threshold.

The resulting test chips showed FPGA programming and operation at supply voltages as low as 0.26 V using ganged tri-state multiplexers with interruptible latches for configuration bit storage. Minimum energy operation for a high-activity test pattern yielded a subthreshold minimum energy point. The following sections show how these results were obtained. Section 3.1 describes the Spectre simulation analysis used to select multiplexers and SRAM alternatives for use in the test chips. Section 3.2 presents the FPGA architecture constructed from these building blocks and compares it to previous work. Section 3.3 presents fabrication and test results. Section 3.4 summarizes.

3.1 Subthreshold FPGA Building Block Design

For FPGAs, the most important digital logic gates are the inverter, the multiplexer, and the configuration storage bit element (usually SRAM). The D flip-flop, while integral to basic logic element (BLE) design, is not as heavily utilized as these other elements. As it has already been well-studied by subthreshold circuit designers, a detailed study was deemed unnecessary. This left two major design tasks at the leaf cell level: choose the best multiplexer design for low voltage FPGA operation, and choose a suitable leaf cell for configuration bit storage—either a subthreshold-optimized SRAM cell or an alternative with good robustness at low voltages. The evaluations peformed to make these choices are presented next.

3.1.1 Multiplexer Evaluation

Figure 3.1 shows the multiplexer alternatives that were considered for the test chips. Four conventional topologies were evaluated: ganged tri-state, logic function, buffered transmission gate, and buffered pass transistor. The ganged tri-state, transmission gate, and pass transistor circuits were also evaluated in dynamic threshold (DTMOS) configurations, i.e. with transistor gates connected to bodies.

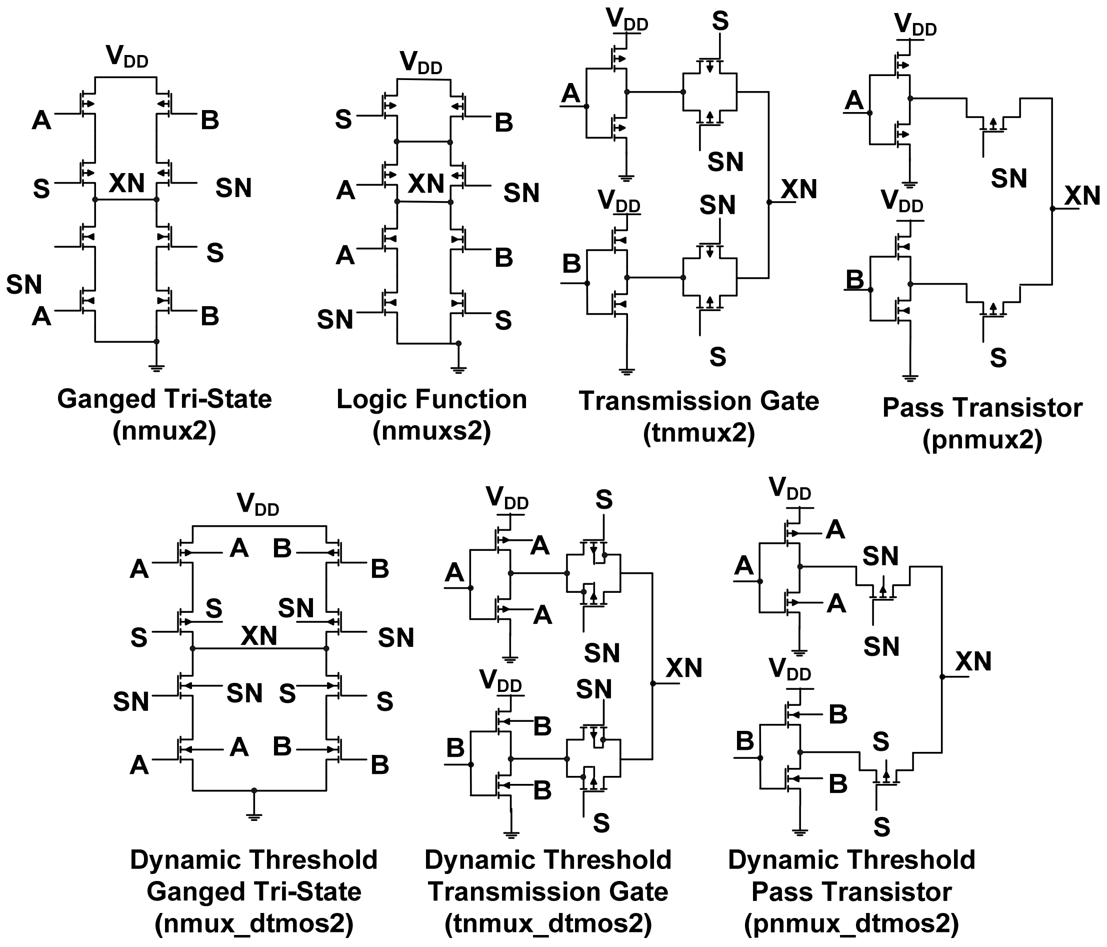


Figure 3.1. Multiplexer configuration alternatives.

Table 3.1 compares the current consumption, delay, and area of the multiplexer options in Figure 3.1. Total current and static current are averaged over all input combinations. Delay is averaged separately for cases where the data inputs and select inputs are transitioning. Both conventional and DTMOS pass transistor multiplexer results are omitted due to the poor robustness of these circuits as discussed below. Results are normalized to the ganged tri-state mux, which appears to be the best all-around performer of the options considered. Although the transmission gate multiplexers are faster, the delay improvement is outweighed by the power penalty. The results also show that for a 2.6X area penalty and 1.3X delay penalty, a 2.2X power reduction can be obtained with DTMOS ganged tri-state muxes. This makes it unclear whether the standard or DTMOS configuration would provide better overall results. For this reason, both a conventional and a DTMOS chip were fabricated. The multiplexer choices used for each are highlighted in green in Table 3.1.

Table 3.1. Power/Performance/Area Evaluation for Multiplexer Alternatives

	nmux2	**nmuxs2**	**nmux_ dtmos2**	**tnmux2**	**tnmux_ dtmos2**
Avg. Current	1.00	0.98	0.45	2.32	0.75
Avg. Static Current	1.00	1.07	0.32	2.31	0.66
A/B-XN Delay	1.00	0.95	1.40	0.86	1.18
S/SN-XN Delay	1.00	1.20	1.28	0.66	1.09
Area	1.00	1.00	2.60	0.99	2.59

To investigate the relative robustness of these circuits, Spectre [57] simulations with foundry-supplied IBM 0.18 μm SOI process device models were performed with each topology in Figure 3.1. The figure of merit for robustness was the ratio of standard deviation to mean (σ/μ) of the multiplexer propagation delay; a lower σ/μ is used as an indication of a more robust circuit. For all multiplexers, transistors were sized with PMOS W/L = 1 μm/0.18 μm and NMOS W/L = 0.5 μm/0.18 μm. To introduce process variations to the large number of voltage and multiplexer combinations without excessive simulation runtime, each test case was simulated over 100 Monte Carlo iterations. Note that each circuit was both driven and loaded by multistage inverter networks that were also subject to variation. This adds an element of realism to the simulations by requiring that the devices under test function correctly even in the presence of potentially very weak drivers. Correct operation of the multiplexer at a given voltage was defined as the existence of valid 10-90% rise and fall times for all 100 iterations.

Figure 3.2 shows the σ/μ of the various multiplexers as a function of supply voltage for high-to-low propagation delay (τ_{plh}), while Figure 3.3 shows the σ/μ for low-to-high propagation delay (τ_{phl}). The separate graphs show that output rising edges are generally more vulnerable to variation. All multiplexers met the pass criterion above 0.1 V except for the DTMOS pass transistor mux. It is therefore omitted from Figure 3.2 and Figure 3.3. The standard pass transistor mux is much less robust than the rest for output falling edges. Significant increases in σ/μ versus the nominal V_{DD} for this process (1.5 V) begin at around 0.5 V for all multiplexer types. At subthreshold (below 0.4 V in this process), the DTMOS circuits exhibit significantly lower τ_{plh} σ/μ compared to the other types. Since the magnitude of these σ/μ values are large, they will dictate the robustness of the overall circuit. This suggests that the DTMOS multiplexers have a meaningful robustness advantage over their conventional counterparts in this process technology.

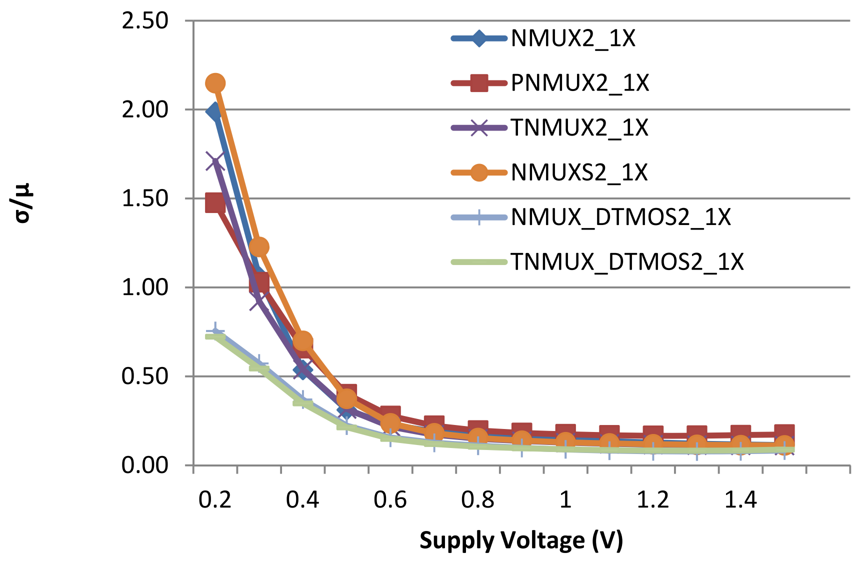


Figure 3.2. Simulated standard deviation to mean ratio of the FPGA multiplexer propagation delay vs. supply voltage for output rising edges

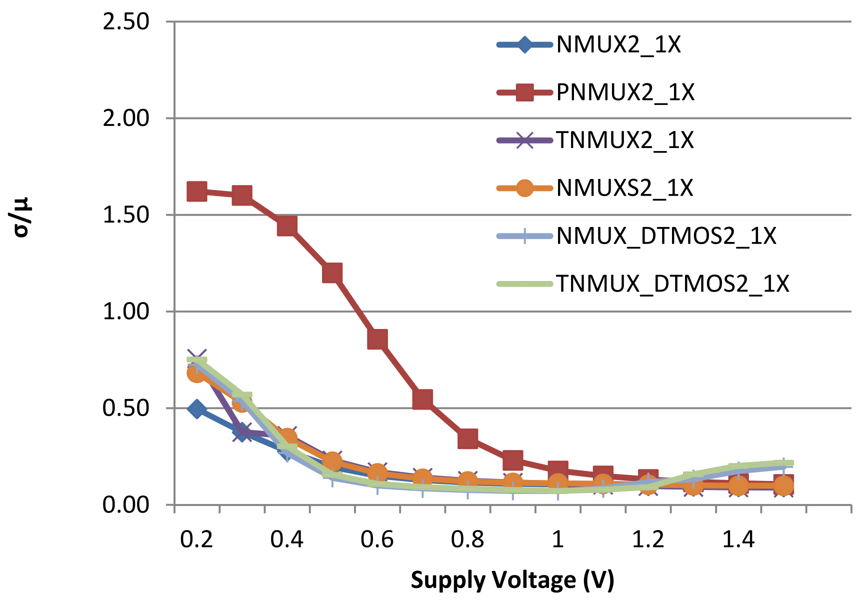
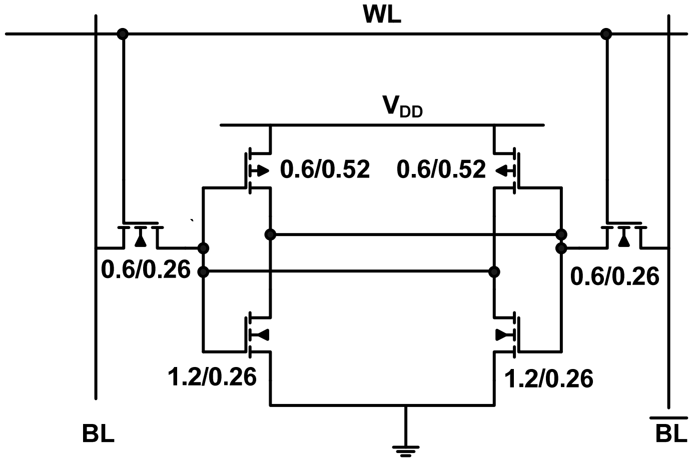


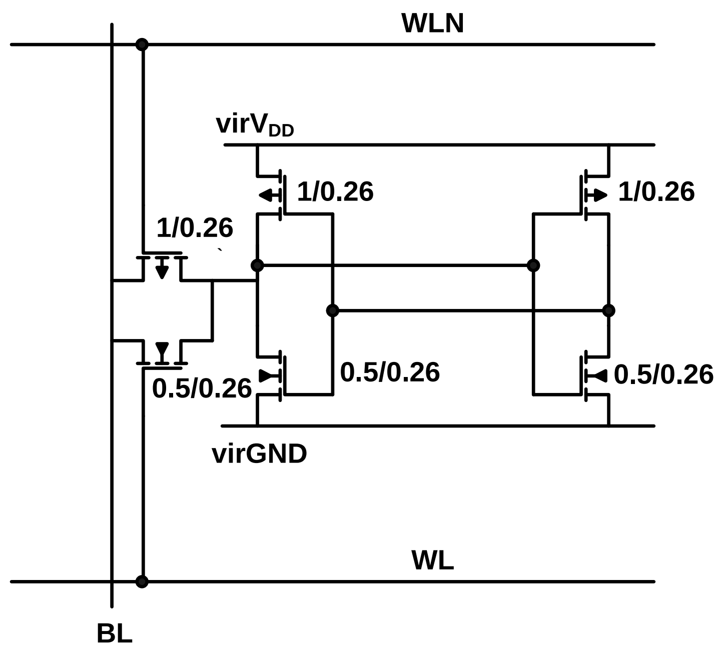
Figure 3.3. Simulated standard deviation to mean ratio of the FPGA multiplexer propagation delay vs. supply voltage for output falling edges.

3.1.2 Replacement of SRAM with Latches

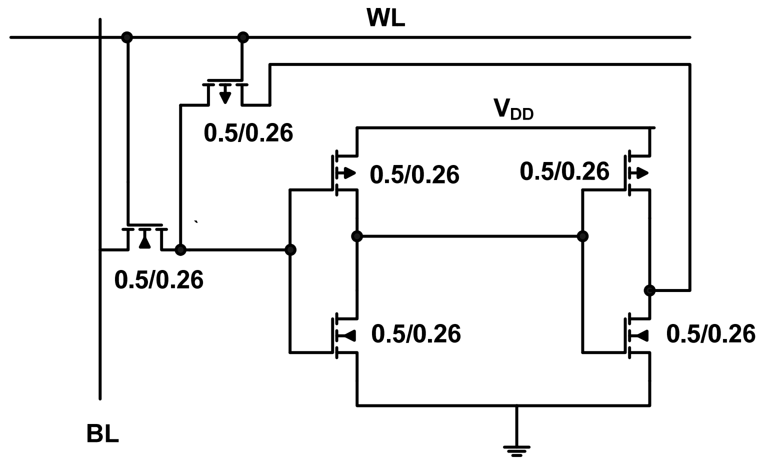
While it is known that the conventional 6T SRAM bit cell, shown in Figure 3.4 (a), is not appropriate for large memory arrays at subthreshold supply voltages [17], their suitability for use as configuration storage in FPGAs is less clear. Configuration bits in FPGAs are rarely written, and the speed of the writes is less critical. Because configuration storage outputs are directly wired to the circuits that they program, reads are only necessary for off-chip verification that programming was successful. This is also a rare operation. The particular write stability and read noise issues cited with operating conventional SRAM are specific to memory design. In FPGAs, localization of word and bit lines can potentially mitigate both write margin and read stability. . However, the contention between the access transistors and the cross-coupled inverter outputs remains an issue for the 6T SRAM cell in subthreshold. An SRAM bit cell that directly addresses this contention without adding extra transistors is shown in Figure 3.4 (b), mitigates this contention by power-gating the cross-coupled inverter supply and using a full transmission gate for the write driver [16].



(a)



(b)



(c)

Figure 3.4. (a) Conventional 6T SRAM; (b) 6T subthreshold SRAM bit cell [16]; (c) 6T interruptible latch.

This work proposes to eliminate the contention entirely by using an interruptible latch instead of SRAM to store FPGA configuration bits. The benefits of using the latch are increased robustness vs. conventional 6T SRAM without paying the area penalty of power gating circuitry and/or extra bit cell transistors in subthreshold SRAM solutions. Figure 3.4 (c) shows the proposed schematic of the latch. Transistor count is maintained at six by using an NMOS pass transistor for write access and a PMOS pass transistor for feedback to a pair of inverters whose outputs are not cross-coupled, meaning that no node in the circuit is driven by two transistors simultaneously.

Simulations were conducted to compare the speed and robustness of the latch to both the conventional SRAM in Figure 3.4 (a) and the subthreshold SRAM in Figure 3.4 (b). Each circuit makes use of the transistor dimensions indicated in Figure 3.4. All transistor lengths in all designs are set to 260 nm so that all three circuits obtain the benefit of increased transistor layout area in mitigating process variations.

A Spectre post-layout simulation deck exercises the propagation delay from the word line to the output as a function of both the supply voltage (varied from 0.1-0.5 V) and the word line pulse width (varied from 10 ns to 1 μs). Word line pulse width was shown as the preferred criterion for SRAM bit cell write stability at subthreshold [58]. For the subthreshold SRAM bit cell, the supply was reduced by $0.25V_{DD}$ and the ground rail was raised by $0.25V_{DD}$ whenever the word line was high. The same methodology for introducing process variation was used in these simulations as in the multiplexer simulations in the previous section. “Success” was defined by obtaining a valid propagation delay result for both rising and falling output transitions.

Figure 3.5 shows each circuit’s minimum pulse width as a function of supply voltage. The conventional SRAM bit cell and the latch meet the pass criterion at 0.25 V and above, while the subthreshold SRAM meets it at 0.2 V and above. The subthreshold

SRAM also has the smallest propagation delays since it drives the inverters with a full transmission gate. Figure 3.6 shows a plot of the standard deviation to mean ratio of propagation delay for each cell. As with the multiplexers, significant differences between τ_{phl} and τ_{plh} values were found. At lower voltages (0.25-0.3 V) the τ_{phl} variation of the SRAM cells are much greater than that of the latch. However, the τ_{plh} variation of the latch and the subthreshold bit cells are comparable to that of the conventional SRAM at low voltages, while the conventional SRAM has the best τ_{plh} variation at near-threshold voltages (0.4-0.5 V). Overall, the latch exhibited the least delay variation. Because it offered low delay variation without the area overhead of write assist circuitry required by the faster subthreshold SRAM bitcell, the latch was chosen as the configuration bit storage element for this work.

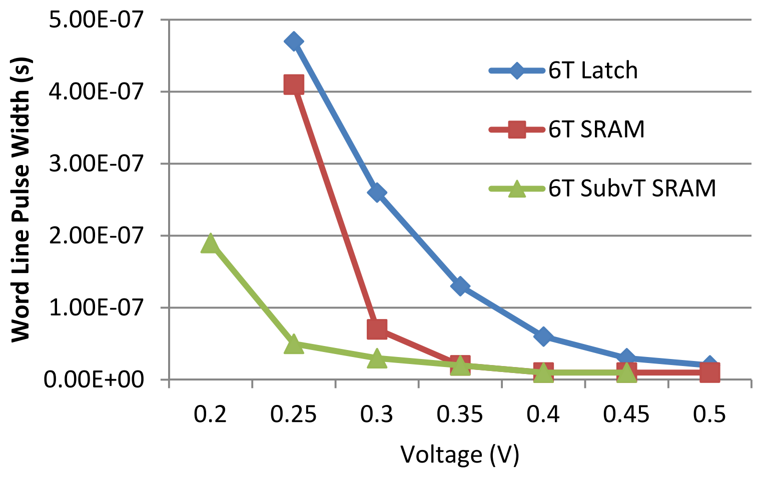
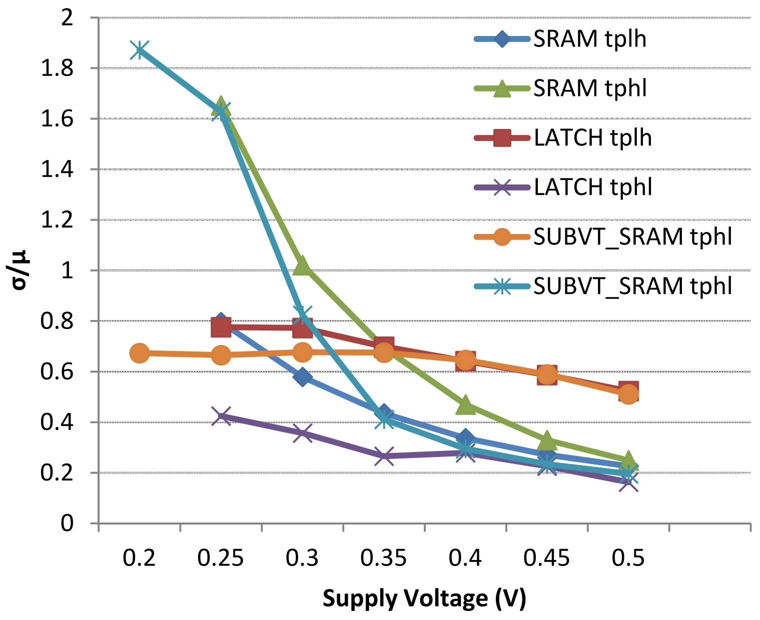


Figure 3.5. Minimum word line pulse width vs. supply voltage for the 6T conventional SRAM, 6T subthreshold SRAM, and 6T latch.



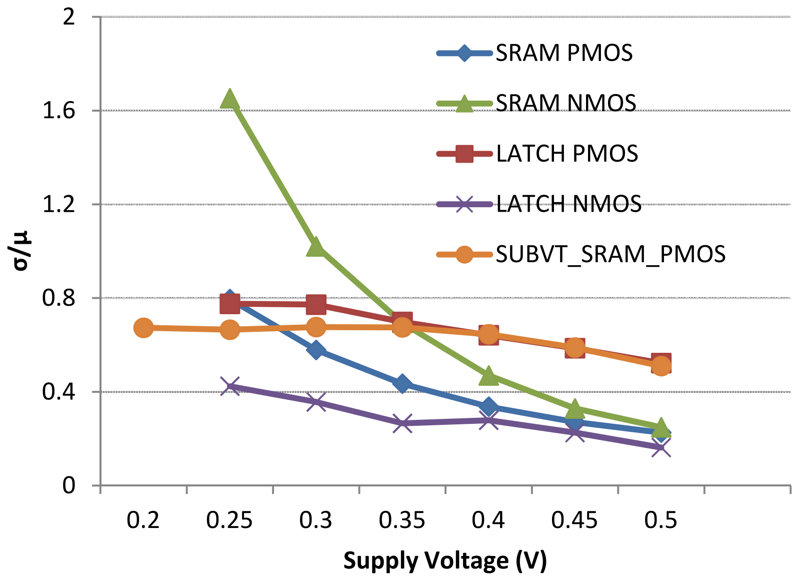


Figure 3.6. Standard deviation to mean ratio of propagation delay in the 6T SRAM bit cell, subthreshold 6T SRAM bit cell and 6T latch.

3.2 Test Chip Architecture

The test chip architecture is based upon the clustered, island-style class of FPGA architectures [29]. The array element in these architectures is called a tile and is comprised of a configurable logic block (CLB), a switch box (SB), and upper and lower connection boxes (CBs) that interface the CLBs to the SBs. Figure 3.7 shows a block diagram of the tile.

The tile in Figure 3.7 is replicated sixteen times on the test chip to form a 4x4 array. CLB and CB I/O are connected to test chip pads, which eliminates the need to design I/O blocks (IOBs) and facilitates verification of the tile functionality.

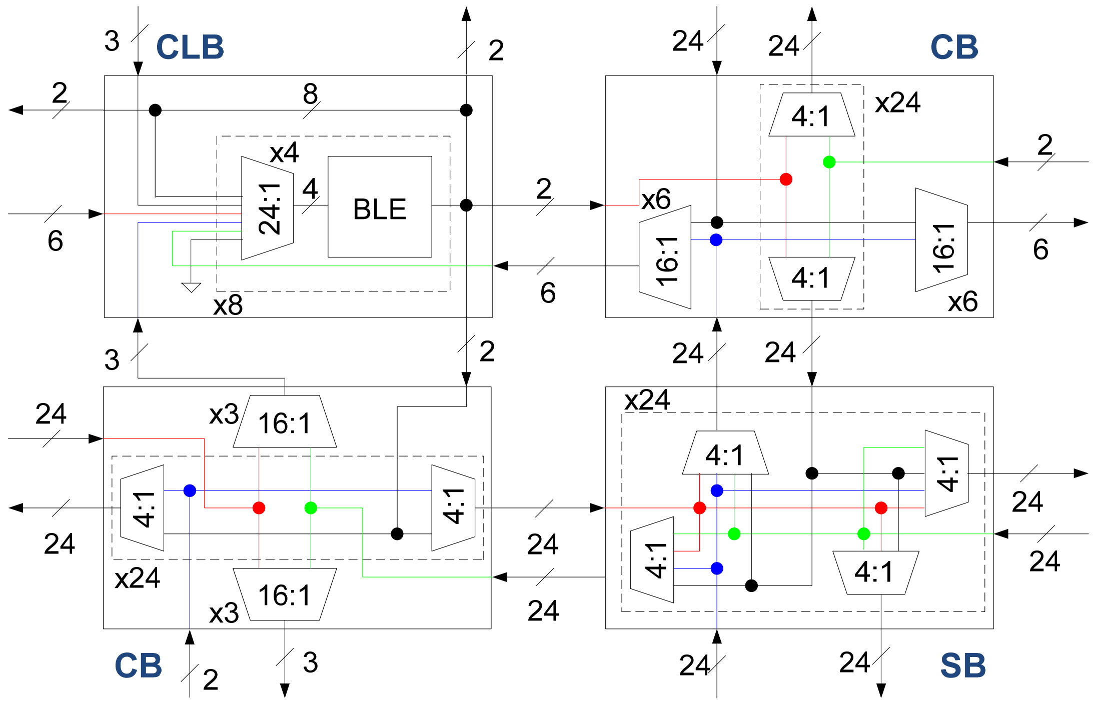


Figure 3.7. Test Chip Tile Block Diagram.

3.2.1 Tile Design

The CLBs in this work are eighteen-input, eight output circuits containing eight four-input basic logic elements (BLEs). Figure 3.8 shows a block diagram for a BLE like the one used in this work. A four-input lookup table (LUT) is built by connecting configuration bit outputs, usually implemented with SRAM, to a 16:1 multiplexer. The four BLE inputs are connected to the select bits of the multiplexer. A flip-flop is connected along with the LUT output to an output multiplexer that drives the BLE output. Eight BLEs per CLB has been shown to be a good number per CLB for low power design [59]. The eighteen inputs can be routed to any input of each of the eight LUTs using a nearly full crossbar [60] built with 24:1 multiplexers. This 24:1 muxes represent a significant amount of hardware overhead but the resulting input flexibility simplifies synthesis, placement, and routing to the FPGA.

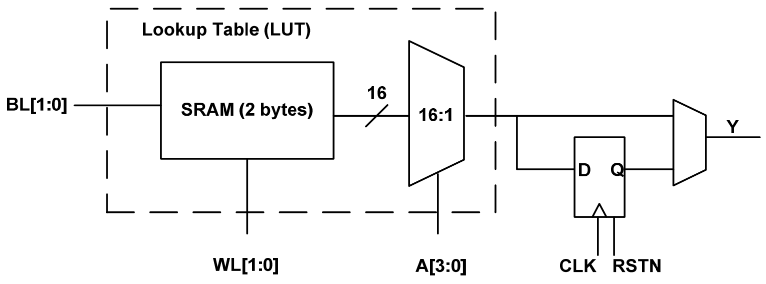


Figure 3.8. Test Chip Basic Logic Element (BLE).

The CBs and SBs are arrays of multiplexers, buffered at the inputs and outputs by inverters. It was shown in [30] that single-driver, unidirectional routing resources such as these are more efficient than bidirectional routing resources, reducing capacitance on each routing track while offering comparable routability per wire. The switch box is a subset switch box providing 48 total routing tracks in both vertical and horizontal directions. For unidirectional routing tracks, this requires 24 copies of a circuit containing four 4:1 multiplexers. Each 4:1 mux selects which of the four incoming sides is driven on the output and generates the output for one of the four sides.

The connection boxes contain arrays of multiplexers generating two types of signals: inputs to adjacent CLBs and inputs to adjacent SBs. Each CB interfaces to the CLB and SB in its own tile and to one CLB and one SB in adjacent tiles. To generate each CLB input, the CB selects from one of sixteen SB outputs. Eight of these come from within the tile and eight come from an adjacent tile. The switch box routing tracks are divided evenly amongst the 16:1 multiplexers driving the CLB inputs. In the lower CB, each routing track goes to one 16:1 mux; in the upper CB, each goes to two. The CLB inputs are divided unevenly between the CBs in order to prevent multiplexer inputs from being left unused. To generate each SB input, 4:1 multiplexers are provided so that each CLB output can be routed to any SB routing track. This flexibility comes at the expense of requiring buffering on the CLB outputs to cope with the high fanout of these nets.

3.2.2 Programming Interface Design

Random access programming is implemented to mitigate subthreshold supply variation and support testability. Configuration bits are organized into 8-bit words. Each group of eight latches forming a word is clocked by a word line generated from a seven-bit address decoder provided in each tile. The address lines are gated at the chip level by one-hot encoded ROW and COLUMN signals used to select which tile is being programmed. This results in separate gated address lines being distributed to each tile. Address zero is reserved so that when all address bits are low, no latches are transparent. This scheme is summarized in the block diagram in Figure 3.9. Each tile requires 98 configuration bytes: 41 words are required for the CLB, 33 for the connection boxes, and 24 for the switch box.

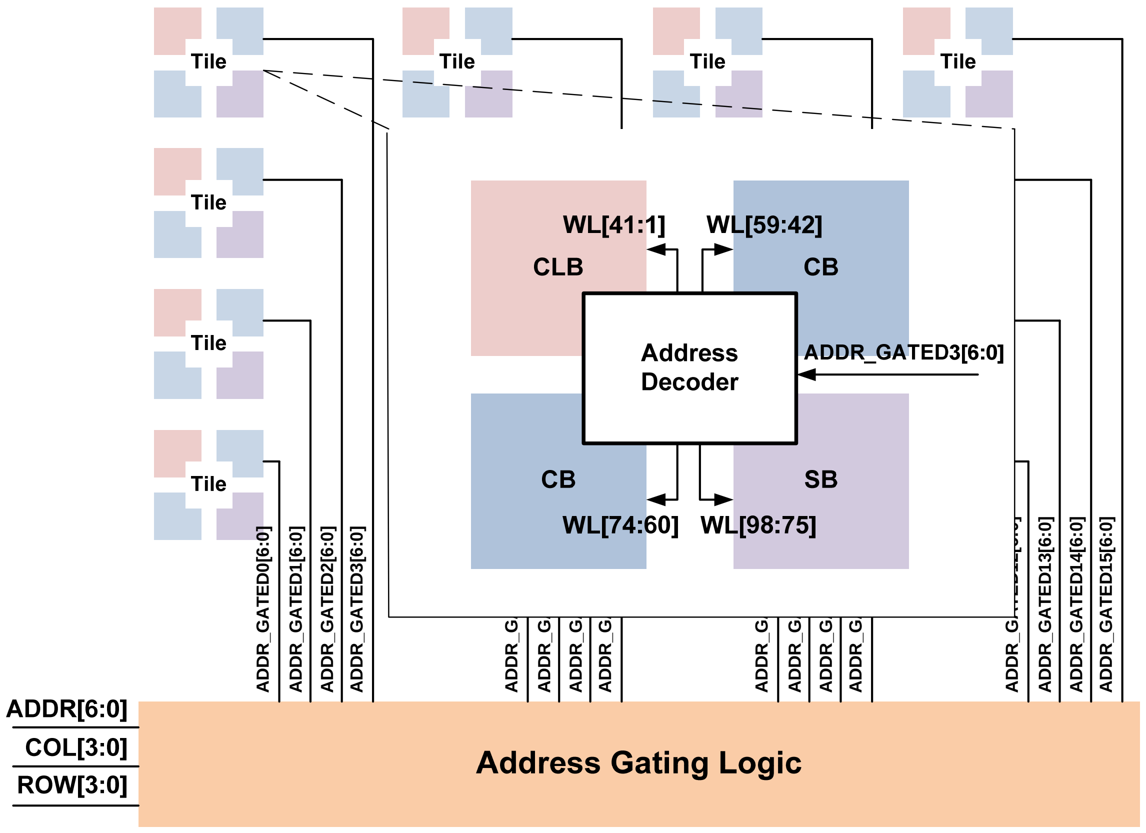


Figure 3.9. Programming circuit block diagram

3.2.3 Comparison to Previous Work

Table 3.2 provides a summary comparison between the architecture used in this work and that in [8]. The key differences between the two chips are the configuration bit storage used and the implementation and type of routing resources offered. Instead of replacing SRAM with latches, the FPGA in [8] used SRAM bit cells with a secondary,supply voltage. This enabled programming the FPGA with the secondary voltage as low as 0.4 V. The FPGA logic resources operated with the primary supply voltage as low as 0.2 V. Whereas the FPGA in [8] used novel circuit design to mitigate variation for pass gate-based bidirectional routing resources, this work achieves subthreshold operation with conventional, unidirectional multiplexer-based routing switches. The choice to use unidirectional routing resources was primarily motivated by their common use in commercial FPGAs and the studies showing that unidirectional routing fabrics are more area efficient, resulting in lower capacitance routing nets [61]. An architectural study of FPGAs compared bidirectional and unidirectional routing architectures for power and speed, showing that in nearly all cases the unidirectional routing resources were more energy-efficient [61]. Bidirectional resources were more energy-efficient only at low frequencies (< 10 MHz). Although the low frequency case is the relevant case for subthreshold operation, the study's results are for nominal supply voltages. It remains an open question for future work to determine whether the area penalty of bidirectional routing resources can be justified by an energy efficiency benefit for subthreshold FPGAs.

Table 3.2. Comparison of Subthreshold FPGA Architectures

	[8]	This Work
Configuration Bit Storage	SRAM with elevated V_{DD}	Latches
Routing Type	Bidirectional	Unidirectional
Routing Switch Implementation	Pass transistors + sense amplifiers	Multiplexers
Routing Tracks	36	48
BLEs per CLB	9	8
Local Routing	6 local tracks	CLB Input Multiplexing

3.3 Test Chip Measurement Results

Two test chips, one with conventional multiplexers and one with DTMOS multiplexers, were both fabricated on the same wafer run in the IBM 0.18 μm SOI process. A die photograph of the test chip with conventional multiplexers is shown in

Figure 3.10. The chips are pin compatible and used identical 4.1 × 4.4 mm pad frames. The conventional array core measured roughly 1.8 × 2.3 mm, while the DTMOS array core measured roughly 2.9 × 3.3 mm.

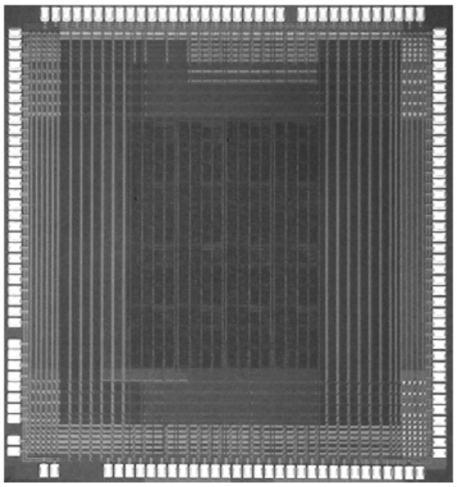


Figure 3.10. Die photograph of the subthreshold FPGA test chip.

Test chips were wafer probed on an Agilent SoC 93000 test system. For the chip with conventional multiplexers, an average minimum operating voltage of 0.30 V was obtained across eleven dice, with three operating at 0.26 V. Die-by-die results are reported in Table 3.3. For the test chip with DTMOS multiplexers, of four chips tested three operated at 0.26 V and the fourth at 0.27 V. When compared to previous work, the minimum operating voltage obtained is greater than the 0.2 V reported for the primary (logic) supply, but less than 0.4 V reported for programming [8].

Table 3.3. Minimum Operating Voltage for 11 Subthreshold FPGA Test Dies with Conventional Multiplexers

Chip #	Min V_{DD} (V)	Chip #	Min V_{DD} (V)
1	0.265	7	0.325
2	0.260	8	0.280

3	0.370	9	0.260
4	0.310	10	0.260
5	0.280	11	0.270
6	0.405	**AVG**	**0.30**

Functional testing was conducted with a range of configuration bitstreams and included programming with random bitstreams. A representative Shmoo plot showing pass/fail results for different clock periods as a function of voltage for a random bitstream is given in Figure 3.11. Performance testing was conducted by programming the test chip to function as an array of sixteen four-bit counters. A sample chip consumed 34.6 µW at 0.26 V and 322 kHz when programmed with the counter array, versus 76.5 mW at 16.7 MHz and 1.5 V. Minimum voltage operation thus led to a 43X reduction in power-delay product (PDP) compared against nominal voltage operation. For the same test, a sample DTMOS chip consumed 23.7 µW at 0.26 V and 202 kHz; the corresponding PDP is 1.1X that of the sample with conventional multiplexers. The DTMOS configuration does not permit nominal voltage operation and for this reason was not characterized at 1.5 V.

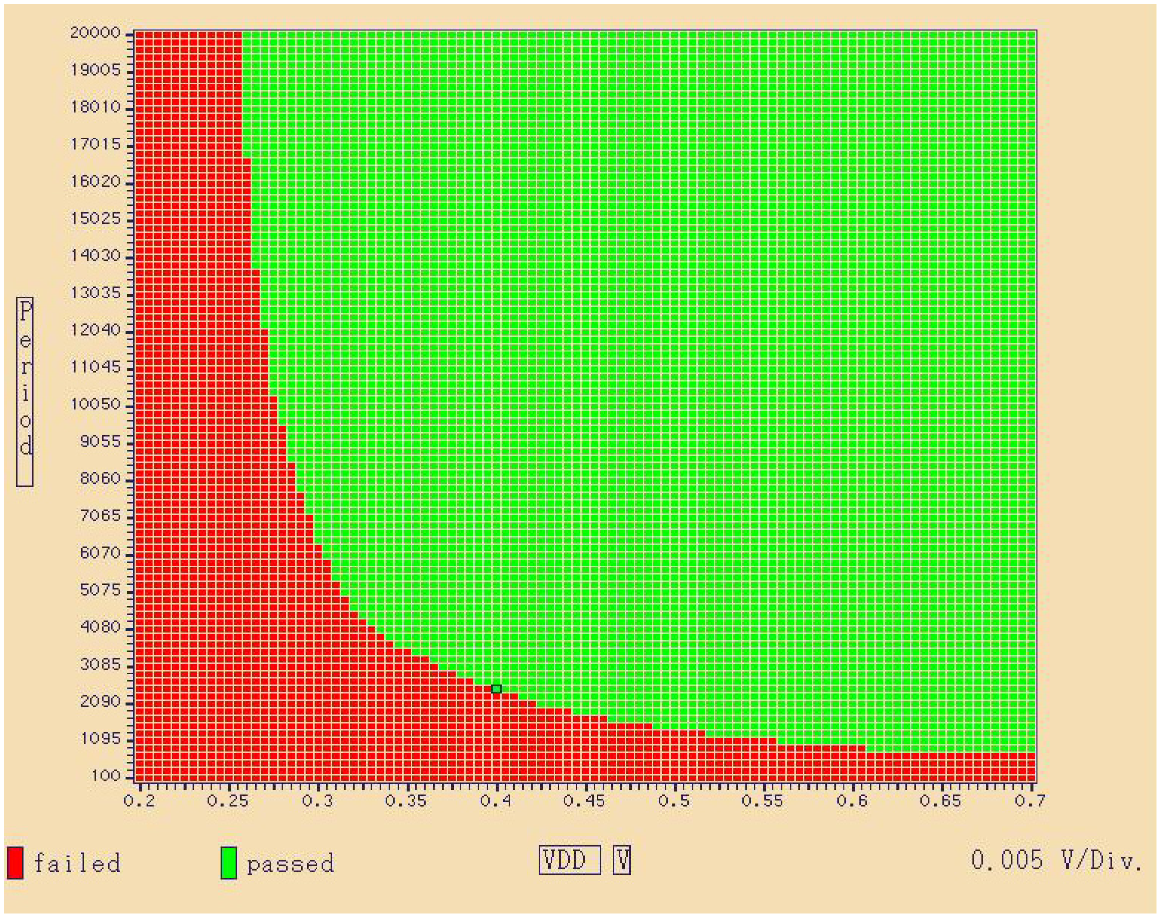


Figure 3.11. Representative Shmoo plot for subthreshold FPGA test chip with conventional multiplexers.

Figure 3.12 shows the power-performance tradeoff within the subthreshold region for a chip with conventional multiplexers. At 0.4 V, below the NMOS threshold voltage and slightly above the PMOS threshold voltage, the maximum frequency increases to 1.6 MHz.

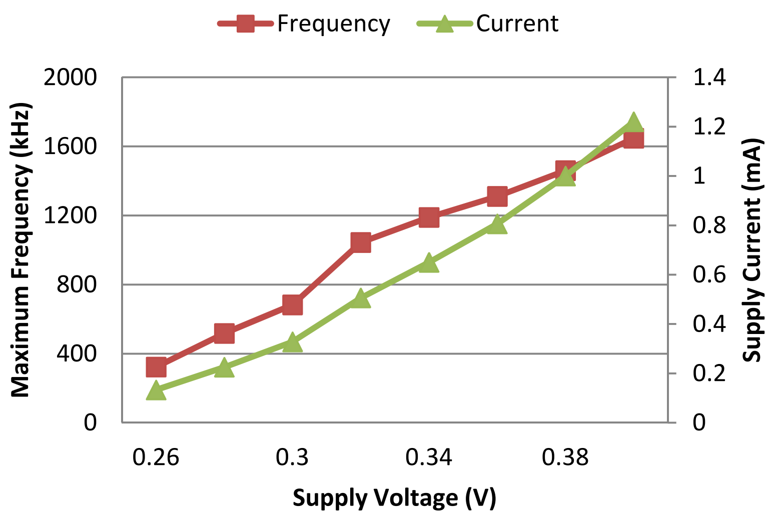


Figure 3.12. Maximum operating frequency and supply current at maximum frequency vs. voltage for subthreshold region operation of the FPGA.

Minimum energy analyses of the four DTMOS chips and four sample chips with conventional multiplexers are shown in Figure 3.13. The PDP of each chip is plotted as a function of voltage. All plots begin at the lowest voltage at which the chips function correctly. The PDP of the DTMOS chips is slightly higher at a given voltage than that of the chips with conventional multiplexers. In both cases the PDP steadily decreases as the supply voltage decreases, indicating a minimum energy point well below threshold.

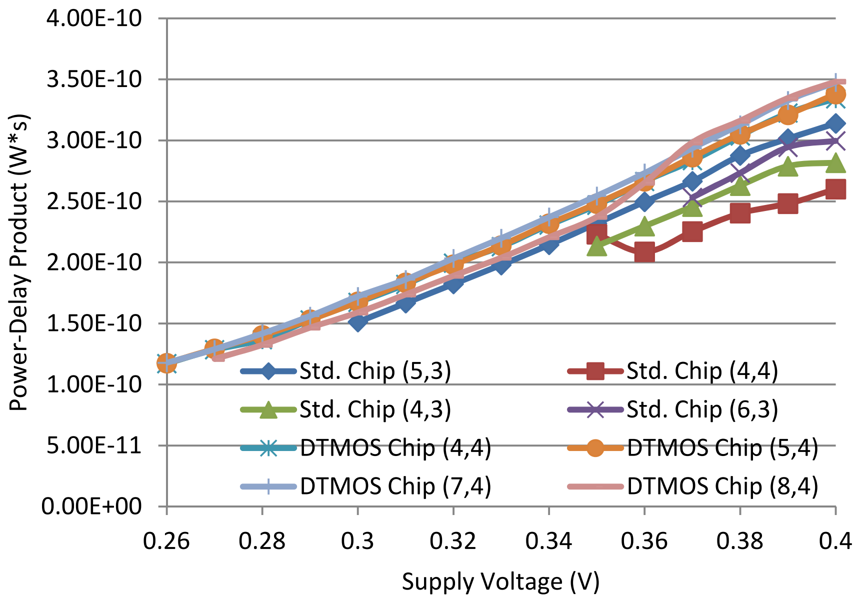


Figure 3.13. Power-delay product vs. supply voltage for four conventional and four DTMOS FPGA test chip dice programmed as arrays of 4-bit counters.

3.4 Summary

A pair of clustered, island-style subthreshold FPGA test chip with unidirectional routing fabric was fabricated in 0.18 µm SOI. A study of different multiplexer circuit topologies showed that ganged tri-state multiplexers have the best overall balance of area, delay, power consumption, and robustness for use in constructing subthreshold FPGA routing fabrics. The use of the DTMOS transistor configuration offers an improvement in robustness at the expense of area with similar energy efficiency to conventional multiplexers.

By replacing SRAM with interruptible latches, an average minimum operating voltage of 0.30 V was obtained across 11 dies with conventional multiplexers and an average minimum operating voltage of 0.26 V across four dies with DTMOS multiplexers. This voltage was achieved without the use of write assist circuitry or an elevated second supply voltage to load FPGA configuration bits. The latch achieves this low voltage operation by eliminating contention from writes. Investigation of the minimum energy point of the FPGA for a high-activity test case showed that the minimum energy point of the test chip can occur well below the threshold voltage.

4 MINIMUM ENERGY ANALYSIS OF FPGAS

Performing minimum energy analysis on FPGAs is inherently a more complex task than doing so for ASICs. Whereas an ASIC may require multiple input activity files to cover various operating modes or expected levels of on-chip activity, an FPGA bears this requirement across a representative range of ways that the FPGA can be programmed. If the circuit mapped to an FPGA occupies only a handful of BLEs, the corresponding on-chip switching activity will likely be dramatically less than if most of the BLEs are used. On the other hand, two circuits using a comparable number of BLEs might have wildly different placement and routing solutions, resulting in very different switching activity within the routing fabric.

The implications of these observations for the minimum energy point of FPGAs are not immediately clear. Intuition suggests that the minimum energy point will vary according to how fully the FPGA is utilized, but by how much? And how much will it vary for different circuits with comparable utilization? If the minimum energy point variation is significant enough, and if the energy savings obtained from operating precisely at the minimum energy point is large enough, ultra-low power FPGA platforms would require programmable voltage regulators to tune the FPGA supply voltage to the minimum energy point value; otherwise, energy will be wasted simply by running the FPGA at a suboptimal voltage. To choose the optimal supply voltage, FPGA CAD support for ultra-low power systems would require minimum energy point calculation as part of the tool flow.

This chapter presents a first step to take observations such as the ones made above and quantify them. To do this, the minimum energy point is estimated on a set of 21 benchmarks from the ISCAS '85 benchmark suite widely used in the ASIC community. Due to the lack of CAD tool support for the test chip presented in Chapter 3, the analysis is performed as part of the IC verification phase for an architecture compatible with the popular Versatile Place and Route (VPR) FPGA placement and routing software [37]. Doing this also enables exploration of a pre-tapeout approach for full-chip minimum energy point estimation of FPGAs. With minor adjustments, the flow can also be applied to ASICs.

Section 4.1 provides an overview of the approach, noting what existing software was used and what software and scripts needed to be developed for this analysis. Section 4.2 describes the FPGA architecture used for this study. Section 4.3 indicates the process technology used and reports choices for the transistor-level design of the FPGA building blocks. Section 4.4 describes the characterization of the leaf cells at near- and sub-threshold supply voltages. Section 4.5 provides details on the Verilog simulation flow used to generate switching activity information. Section 4.6 explains how VPR was used to estimate the maximum clock frequency of each benchmark circuit. Section 4.7 shows how the FPGA netlist, characterized cell library, switching activity from simulations, and maximum clock frequency estimate combine to calculate power delay product for each benchmark as a function of supply voltage, thereby obtaining a minimum energy point. Section 4.8 analyzes the results of the minimum energy point estimation for all benchmarks. Section 4.9 contains final observations and remarks.

4.1 Overview of Approach

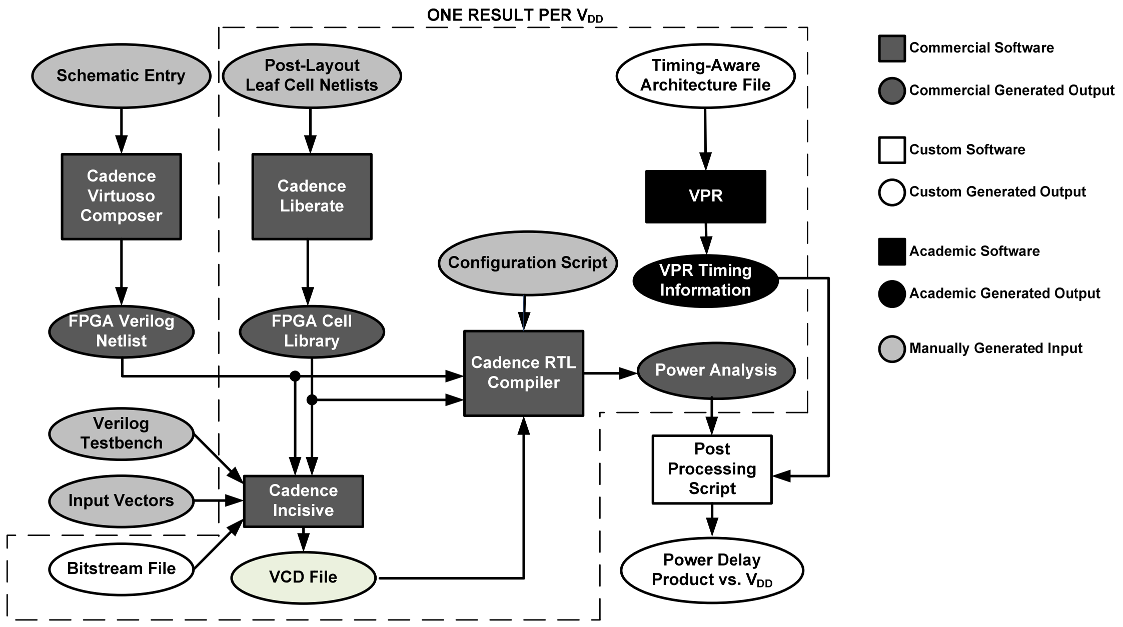


Figure 4.1 shows a flow diagram summarizing the approach taken. The upper left corner shows that for these experiments, a Verilog netlist is obtained from schematics generated with Cadence Virtuoso Composer. In practice, any technique for producing a valid gate-level Verilog netlist may be substituted. Section 4.2.2 will describe a technique developed as part of this work for using VPR routing graphs to provide information for schematic generation to ensure that VPR placed and routed to the FPGA correctly.

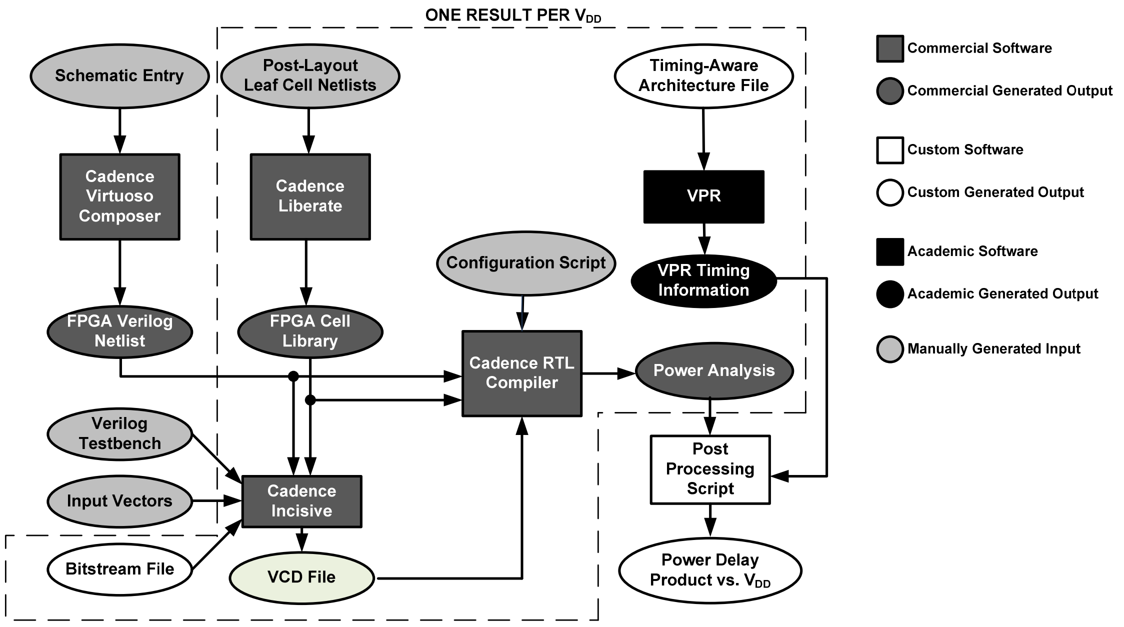


Figure 4.1. Flow diagram of steps used to perform FPGA minimum energy analysis.

For both simulation and power estimation in this flow, a characterized leaf cell library is required—the Synopsys Liberty format is required for power estimation, and Verilog is required for simulation. Both library formats were prepared by Liberate, a standard cell library characterization tool offered by Cadence. The Liberty file format stores voltage-specific characterization data, meaning that one Liberty file must be produced for each voltage of interest. Subthreshold cell library characterization has been reported previously [62], but use of a particular industry-standard characterization tools at subthreshold supply voltages is generally not documented.. Section 4.4 describes the tuning of Liberate for subthreshold operation performed in this work.

Simulation also requires several other input files: an input vector file, a Verilog testbench, and a bitstream file containing the configuration bits needed to map a particular benchmark to the FPGA. Generation of the bitstream file was not supported by available CAD tools. For this reason, a bitstream generator was developed as part of this work.. The output of the simulation is a value change dump (VCD) file that contains switching activity information for all nets in the FPGA. Because the simulation timing information is voltage-specific, the circuit must be resimulated at each voltage of interest, as suggested by Meinerzhagen et al. for ASICs [63]. The bitstream generator and other simulation setup details will be discussed further in section 4.5.

With the netlist, cell library, and VCD file, a short configuration script enables Cadence RTL Compiler, a commercial synthesis tool, to perform a power estimation specific to the switching activity contained in the VCD file at the voltage specified in the cell library This power estimate multiplied by the clock period of the benchmark gives a power-delay product. The clock period used in this analysis is the minimum period at which the benchmark will run; this is obtained from the critical path delay estimated by VPR during placement and routing of the benchmark circuit. VPR obtains timing information for this delay estimate from a voltage-specific input architecture file. The format of the architecture file is documented in the VPR User Manual [64]. It is necessary to maintain consistency between the timing information in the cell library and the architecture file. As part of this work, a technique was developed to ensure consistency and automate the generation of the input architecture file at each voltage of interest. This technique is discussed in section 4.5.

By performing power estimation and critical path delay estimation across an appropriate range of supply voltages, a power delay product versus supply voltage curve can be generated for any circuit that can be mapped to the target FPGA. The minimum of this curve is the minimum energy point.

4.2 FPGA Architecture Used

The following subsections present the FPGA architecture for which minimum energy analysis was performed. Because the architecture was intended to be used with a VPR-based CAD flow, it is specified in section 4.2.1 in terms of standard clustered island-style architecture parameters generally associated with VPR. Other top-level design parameters, such as array dimensions, are also specified in this section. Section 4.2.2 describes scripts that were used to process VPR output so that the FPGA netlist has exact correspondence to the architecture generated by VPR. The hardware implementation of the architecture described by these parameters is presented in section 4.2.3, including those aspects of the design that are not described by VPR parameters such as programming circuitry.

4.2.1 Architecture Parameters

The architecture used in these experiments is a clustered, island-style FPGA comprised of a 6x6 array of core (logic) tiles and six I/O tiles on each side. Each core tile contains eight four-input lookup tables (4-LUTs), and each I/O tile contains two I/O blocks (IOBs). The architecture therefore supports designs up to 288 4-LUTs and 48 total I/Os.

Table 4.1 shows the architecture parameters used to define the architecture within VPR. Setting the routing track width to 20 is done by passing a command-line argument to VPR. Further information regarding architecture description in VPR can be found in the VPR 6.0 User Manual [64].

The parameter value choices in Table 4.1 were made to match the CLB design and routing fabric as closely to the test chip presented in Chapter 3 as possible except for track count and I/O count per IOB. The I/O count per IOB is based upon coarse tile area estimates and a reasonable pad pitch suggesting that the tile would be two I/O pads wide. The track count was selected by determining the minimum number of routing tracks required to route all ISCAS ’85 benchmarks with 48 I/Os or less and adding 10%.

Table 4.1: VPR Architecture parameters used

VPR Parameter	Description	Value
K	number of inputs per LUT	4
N	cluster size (BLEs per CLB)	8
I	inputs per CLB	18
F_{cin}	input flexibility	0.333
F_{cout}	output flexibility	1
F_s	switch box flexibility	3
W	routing track width	20
L	segment length	1

4.2.2 Verification of CAD Tool Compatibility

In order to guide schematic generation toward a VPR-compatible FPGA netlist, a method was developed to derive the routing fabric design directly from the VPR routing graph. This method translates the contents of a VPR routing graph file into an easy-to-read text file that can be quickly and reliably, albeit manually, converted into schematics for simulation with industry standard tools. The resulting text file will be referred to as a routing template file.

The routing graph translator developed for this work generates the routing template file with the following steps:

1. Translate the routing graph into a comprehensive list of every routing multiplexer in the FPGA, with inputs and outputs defined by the nodes in the routing graph.
2. Prune the multiplexer list to contain only those multiplexers located within the tiles at (*x*,*y*) coordinates (0,2), (X,2), (2,0), (2,Y), and (2,2). X and Y are the determined by the coordinates for I/O tiles on the right side and top of the FPGA, respectively, based on VPR's tile coordinate system. The pruning step populates the template with a single tile for the interior of the FPGA and a single tile for the I/O blocks on each side of the FPGA. Avoidance of (*x*,*y*) coordinates where either coordinate is one ensures that boundary conditions are avoided when populating the template.
3. Apply a set of rules for naming the multiplexer inputs and outputs such that the names directly imply how the tiles are connected to each other.
4. Write the multiplexers with renamed I/O to a text file.

A suite of Perl scripts implements these steps.

4.2.3 Architecture Implementation

The implementation of the architecture described above is similar to that of the test chip described in Chapter 3. The CLB in particular is nearly identical. Figure 4.2 shows a block diagram of the CLB. The BLE shown in Figure 4.2 has inputs generated by a 32:1 mux instead of the 24:1 mux used in the test chip. This is because the input multiplexing scheme from [60] is not easily translated into VPR architecture file syntax. The 32:1 multiplexer provides support for a full crossbar, in which all inputs can be selected by all BLE inputs.

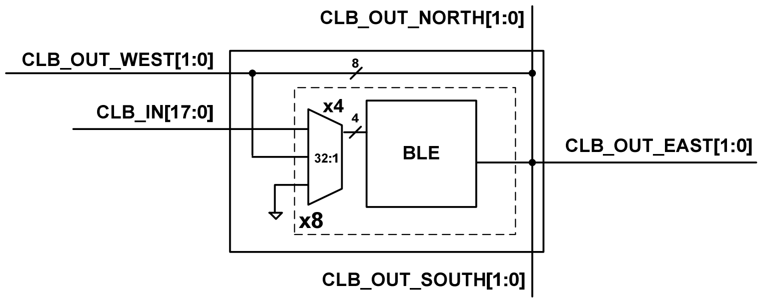


Figure 4.2. CLB Block Diagram

Figure 4.3 shows schematics of the switch box unit cells for horizontal and vertical routing channels. These are implemented as 8:1 multiplexers. Because the architecture calls for fully connected outputs, the CLB outputs connect to each track in the switch box. Since the switch box flexibility is three, three routing tracks connect to each multiplexer. The eighth multiplexer input is tied to ground. Use of this connection enables a reduction in chip activity by ensuring that unused routing tracks are held constant.

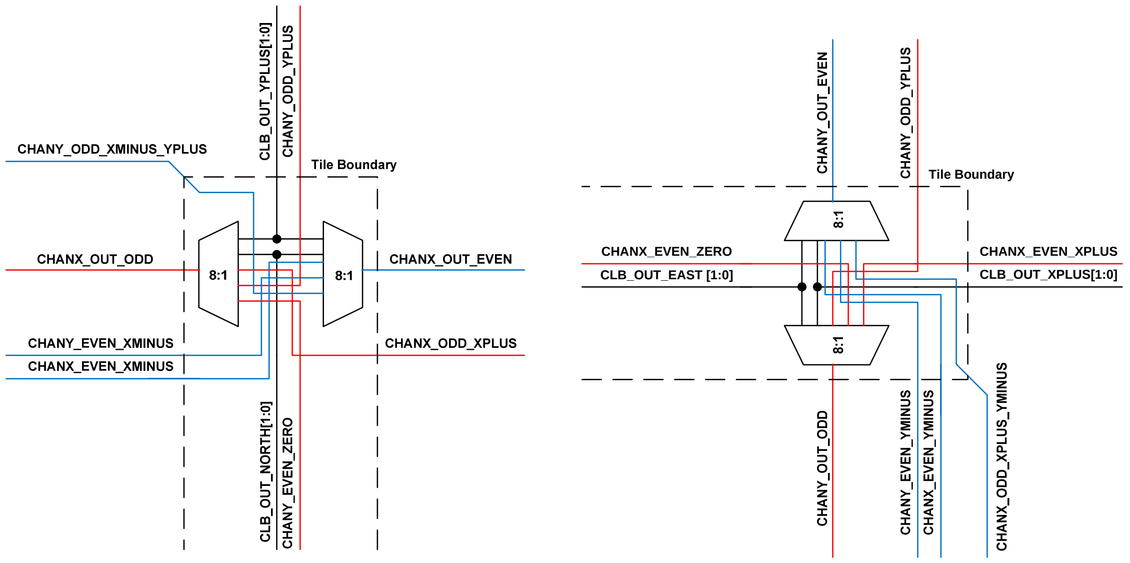


Figure 4.3. Switch box horizontal (left) and vertical (right) unit cells

Figure 4.4 provides a graphical representation of the complete switch box and its interface to the CLB. Because the CLB is positioned in the lower left corner of the tile, each switch box receives CLB inputs both from within the tile and from neighboring tiles to the north and east. Since the routing tracks are unidirectional and there are twenty total, each side of the switch box has ten routing tracks per data direction. Because the switch box flexibility is three, the same routing tracks are distributed to multiple tiles. For example, each tile receives even vertical channels from adjacent tiles both to the west and south.

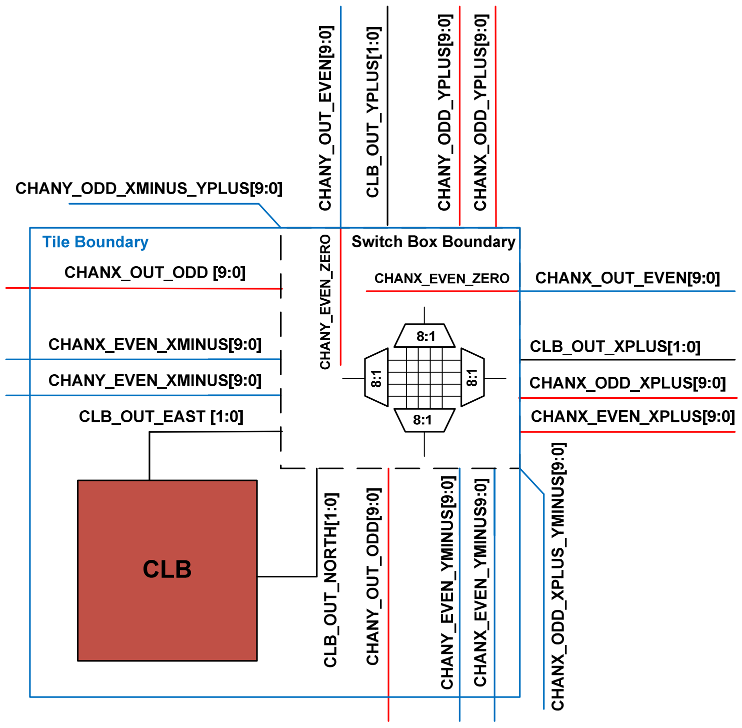


Figure 4.4. Switch box block diagram

Figure 4.5 shows a block diagram of the connection box. The connection box is built from 8:1 multiplexers. Since the input flexibility is 0.333 and there are twenty routing tracks, each CLB input may select from six different routing tracks. As with the switch box, an option to select ground is provided so that unused inputs can be held constant. The connection box is thus implemented with 8:1 multiplexers.

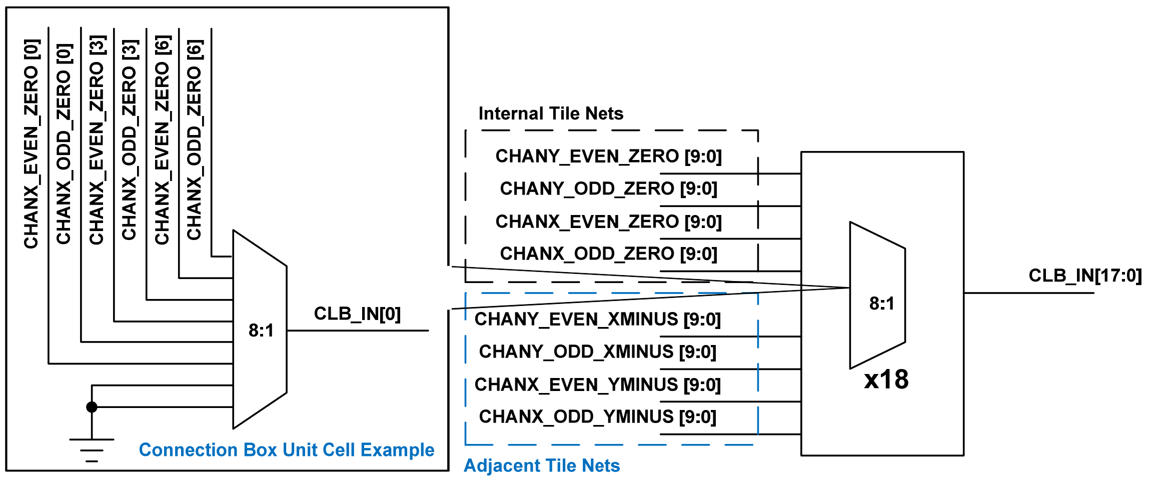


Figure 4.5. Connection box block diagram

A minimal I/O tile is provided to support VPR's architecture definition of I/O tiles. The tile coordinate system built in to VPR places portions of the routing fabric in I/O tiles on the left and bottom sides of the array. Instead of a CLB, each I/O tile contains an I/O block (IOB) per I/O pad connected to the I/O tile. In this implementation, there are two bidirectional I/Os per I/O tile. The I/O block consists of a tri-state buffer, D flip-flops for reregistering I/O, and multiplexing to select whether the flip-flops are used. The tri-state buffer is enabled when the I/O is configured as an output, and disabled when the I/O is an input. The implementation is shown in Figure 4.6. Each I/O tile also requires a connection box to select which signals will be routed to the IOBs. Partial switch boxes containing vertical channels are implemented on the left side IOBs and partial switch boxes with horizontal channels are implemented on bottom side IOBs. A high-level block diagram showing the connections between the IOB, connection box, and switch box for a left side IOB is shown in Figure 4.7. The other tiles are similar.

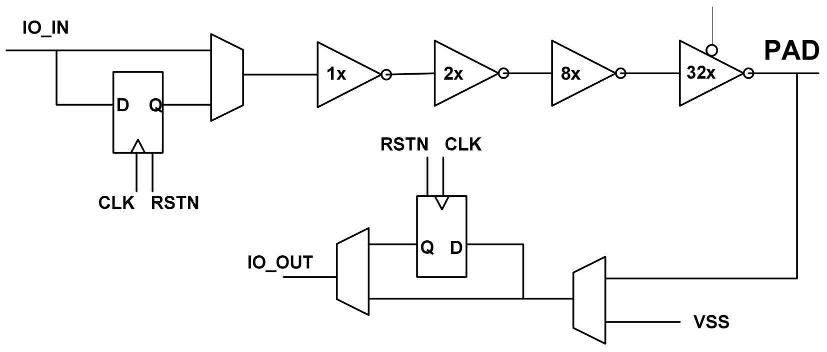


Figure 4.6. I/O Block (IOB) schematic

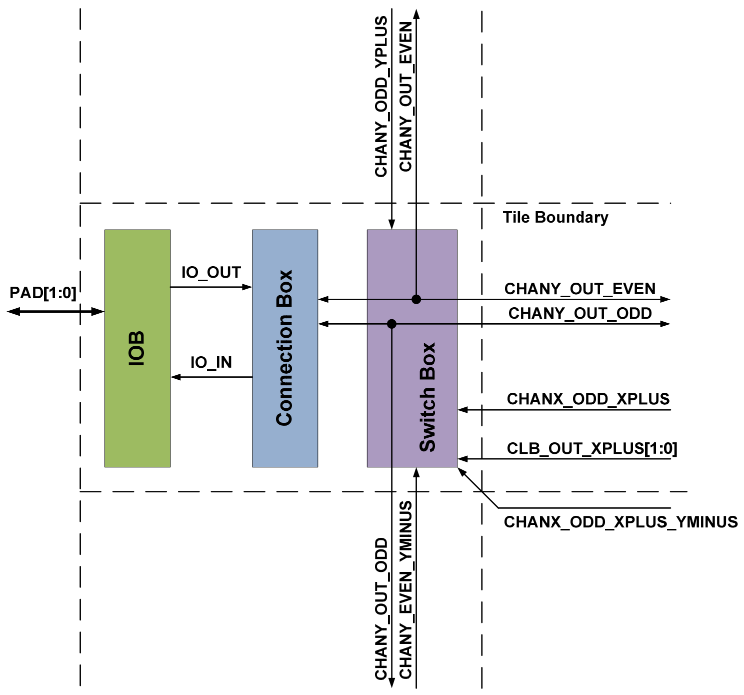


Figure 4.7. Left side I/O tile block diagram

The programming circuitry used is a revised version of the circuitry in Chapter 3, augmented to offer support for configuration byte readback. Since the configuration bits are stored in latches rather than SRAM, there are no common output nets to share among configuration bytes. Instead, additional latches are added to permit serial shifting of data through the tiles to a readout port. Serial shifting was avoided for programming, but supporting random access reads presents the unattractive options of sacrificing noise margin to support multi-driver shared bus lines or implementing large multiplexer trees to support single-driver shared bus lines. The programming circuit for each configuration word is shown in Figure 4.8. An enable signal, when high, enables a programming clock PHI2 to assert and make the latches transparent. For configuration byte readout, two-phase clocking is used. When readout is enabled by asserting the SHIFT_MODE signal, the PHI1 clock loads configuration bytes into a temporary storage latch. The shift operation is completed when PHI2 is asserted. Although this scheme is expensive in terms of required hardware, it offers a great deal of robustness. The two-phase clocking approach provides great flexibility in coping with the high uncertainty in clock skew, setup times, and hold times present in sequential circuits at subthreshold supply voltages. In addition to reducing clock frequency to cope with setup time issues, off-chip clock generation schemes can insert time as needed between the PHI1 trailing edge and PHI2 rising edge to cope with clock skew and hold time constraints.

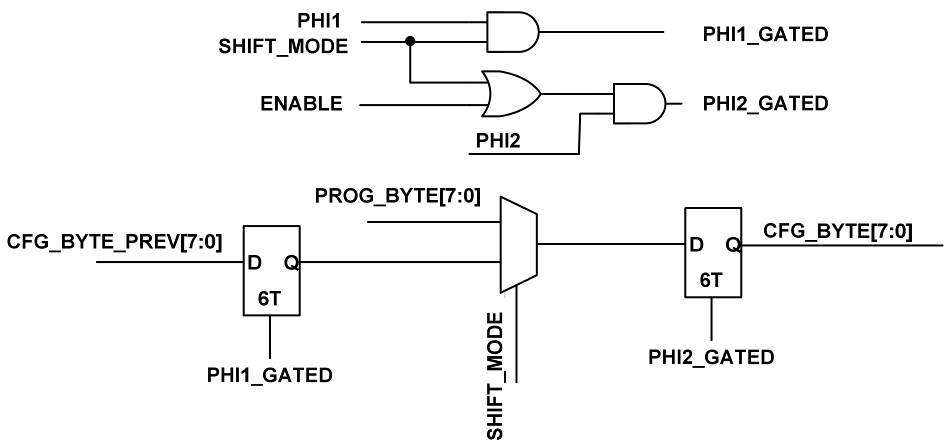


Figure 4.8 Programming circuit supporting readback of configuration bits

Each core tile contains seventy configuration bytes. To manage the programming of these bytes, each tile contains an address decoder. The enable signal of each byte is connected to a decoder output, or word line (WL). The eight-bit data bus for loading configuration bytes is shared among all tiles, so tile decoding logic determines whether a given tile is being programmed. Figure 4.9 shows a block diagram of the tile that includes this logic. 41 words are required for the CLB, 9 for the connection box, and 20 for the switch box.

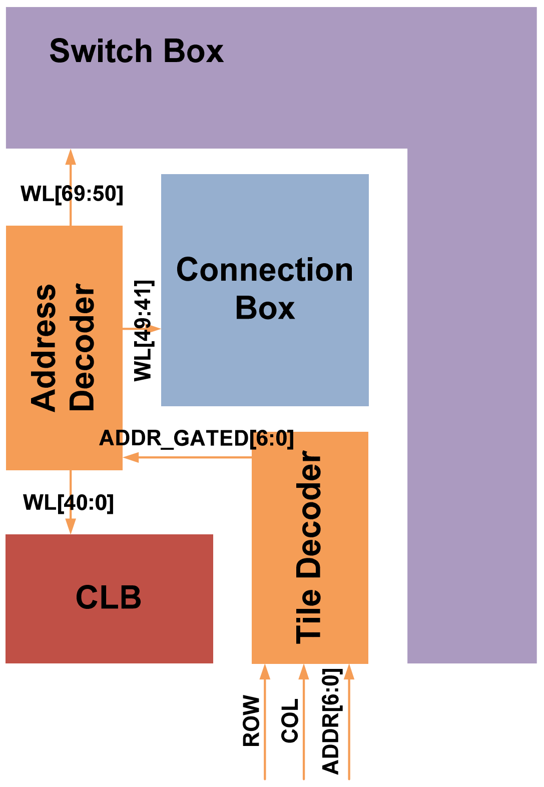


Figure 4.9. Tile block diagram showing programming byte address decoding

The FPGA used here is a basic and representative sample of clustered island-style FPGAs. The implementation details, however, have been tailored for low voltage operation. At the leaf cell level, the convention of using pass transistors or transmission gates for multiplexers has been abandoned in favor of ganged tri-state multiplexers based on results in Chapter 3. SRAM has been replaced with latches for configuration bit storage. Beyond the leaf cell level, the programming and configuration readback schemes both make choices designed to reduce the number of timing hazards present. The programming scheme makes use of random access addressing so that data storage is not dependent on the integrity of long serial shift register chains. The readback strategy reduces the risk of timing hazards on its serial shift register chains by implementing two-phase clocking. These choices are all made with the high-latency, high-variation conditions of subthreshold operation in mind.

4.3 Process Technology and Transistor Information

The architecture was implemented in the IBM 65 nm low power bulk process. Standard-V_{TH} transistors were used throughout. The length of all transistors was set to 100 nm to mitigate area-dependent process variation.

4.4 Library Characterization

One of the foundations of ASIC design and verification is the organization of the leaf cells comprising the lowest level of hierarchy into libraries—cells designed specifically for reuse and carefully characterized to ensure predictable performance. When designing FPGAs, this strategy is less common, but was employed for this work so that IC verification and power estimation software used by ASIC vendors, which assumes this library organization, could be exploited. Fortunately, the library development requirement for FPGAs is greatly reduced compared to that of ASICs—whereas an ASIC library typically contains hundreds of cells, the library used in this work contains just 15. A list of the 15 cells used is provided in Appendix B.

Library characterization is a well-established ASIC design method of systematically and automatically performing a set of SPICE simulations on a library of digital logic gates in order to obtain timing and power information for those cells across a wide range of operating conditions, and then summarizing that information in a single text file. The Synopsys Liberty file format [65] is the de facto standard format for this file and is used throughout academia and industry. While the Liberty format permits varying a number of operating conditions within a single file, such as load capacitance, input slew rate, and circuit state, it permits defining only a single supply voltage, process corner, and temperature. For this reason, Liberty-based IC verification typically requires multiple Liberty files per library. In this work, unique libraries were generated for each supply voltage simulated at each of three process corners (slow-slow, typical-typical, and fast-fast). Temperature was held constant at 25 °C.

There are a number of EDA vendors offering library characterization software to streamline the characterization process and generate the Liberty file. In order for this software to be able to characterize libraries reliably, the automatic generation of the SPICE testbench for the circuits must be flexible enough to adapt to the extremely long delays associated with subthreshold operation. Both library characterization tools considered for this work, Cadence Liberate and Silvaco Accucell, offered commands to help tailor the simulation runtimes to the slow speeds of subthreshold circuits. However, for some input states of sequential cells Accucell leakage power characterization reported erroneously large leakage currents below 0.5 V. Liberate did not exhibit this inaccuracy, and thus was used throughout this work.

Two input scripts were provided to Liberate for each characterization run: a template file describing the contents of the cell library and general characterization conditions, and a run-specific file tailored to the voltage and process corner of interest. Several commands and internal variables (set with the `set_var` command) were of particular importance. The internal variable `sim_duration` required tuning to the approximate range of transient simulation times required for SPICE to perform its delay and power calculations within Liberate. In general, setting this value approximately an order of magnitude above the largest unit inverter delay obtained across the range of operating conditions was a good choice. Other variables influencing simulation delay or slew rate (`max_transition`, `min_transition`, and the index values for slew range specified in the `define_template` command) required similar care and ballpark knowledge in advance of what the timing characterization results would be. Liberate's `auto_index` feature, in which the software attempts to choose a set of input slew rate values within user-defined bounds, required knowledge of what input slew bounds (set with the `max_transition` and `min_transistion` variables) exercised the circuit in a way that balanced intrinsic and parasitic delay. This ensured that the output rise/fall times were consistent with the input slew rates and that the library data contained a realistic operating range for circuits built from the standard cells. The characterization of sequential cells was particularly sensitive to these delay values—underestimating the `sim_duration` and/or `max_transition` resulted in SPICE simulations not running long enough to characterize the cell, whereas overestimating them resulted in failing of measure commands used to calculate timing parameters (setup and hold calculations proved particularly sensitive). In this work, a different template file was created with new values for these variables every 0.1 V at near-threshold and subthreshold voltages.

To assess the accuracy of the cell library power information at low voltages, the switchbox unit cell from the test chip presented in Chapter 3 (an array of four buffered four-input multiplexers) was simulated with Spectre using a post-layout netlist. The power estimate taken from this was compared against a power estimate using RTL Compiler and the same input vectors as the Spectre simulation. The primary goal of this experiment was to validate consistency between Liberty file power lookup tables produced by Liberate and Spectre-generated power estimates at subthreshold supply voltages. Because the circuit is small enough that switched capacitance power estimates have little influence on the overall power estimate, the experiment minimizes the influence of RTL compiler on the results; instead, the dominant term is the interal switching power of the cells, which is obtained by table lookup from the Liberty file. Table 4.2 shows that the accuracy is between three and eight percent from 0.3V to 0.5 V, indicating that Liberate's ability to produce good power lookup tables extends to subthreshold and near-threshold supply voltages.

Table 4.2. Verification of low voltage power estimation accuracy

Supply Voltage (V)	RTL Compiler Power Estimate (W)	Spectre Power Estimate (W)	Percent Difference in RTL Compiler Estimate
0.3	4.21E-08	4.56E-08	7.7%
0.35	6.06E-08	6.52E-08	7.1%
0.4	8.31E-08	9.01E-08	7.8%
0.45	1.118E-07	1.15E-07	3.2%
0.5	1.413E-07	1.48E-07	4.5%

With sufficient care in accounting for the slow operation of low voltage circuits and a good understanding of cell characterization, reliable library generation is feasible even for subthreshold circuits using existing commercial software. By exploiting commercial software, the need to manually develop power and timing models for the leaf cells used in this work was avoided.

4.5 Activity Data Generation

Generation of the activity file was performed by running gate-level Verilog simulations with SDF back-annotated timing to produce value change dump (VCD) files for use by RTL Compiler. In order to support the Verilog simulations, it was necessary to develop a scheme for programming the FPGA in simulation. This in turn required FPGA CAD support. The Verilog-to-routing flow (VTR) outlined in Chapter 2 covers all required elements of the CAD flow except for bitstream generation. A custom bitstream generator was written for this work and is described next, followed by a summary of the Verilog testbench used to simulate the FPGA and a summary of the benchmark circuits simulated.

4.5.1 Bitstream generation

For bitstream generation, this work aggregates required information for programming from VPR output files and manually created files. The following values are required:

- Each bit in each LUT occupied by the user circuit
- I/O-related configuration bits in IOBs occupied by the user circuit
- Each select bit in each multiplexer in the routing fabric (SB and CB)
- Each select bit in each multiplexer in the CLB input crossbar
- The select bits that choose whether or not the BLE output is registered with the D flip-flop

In addition, the bitstream generator must order the programming bits so that they are loaded into the correct locations in the FPGA. For programming scheme presented in section 4.2.3, programming bits are organized into eight-bit words. Each eight-bit word within a tile is assigned a seven-bit address. One-hot encoded row and column values select which tile is being programmed. Each programming bit in the FPGA must thus be mapped to a particular tile row number, column number, address, and bit position.

The bitstream generator targets the FPGA presented in section 4.2.3. It presumes the programming scheme described above, although the word widths, address bus width, and array dimensions are parameterized. The bitstream generator is also capable of adapting to different values for the VPR parameters in Table 4.1, but testing it across a wide range of architectures and programming methods is beyond the scope of this work. For each of the programming bit types in the list above, hand-generated text files define the address and bit number associated with each programming bit. Information from VPR output is then used to determine the value of each programming except for the IOB flip-flop selection, which must be determined manually. In all the experiments performed in this work, I/Os are not reregistered. Figure 4.10 shows a flow diagram for a prototype software bundle that couples architecture template generation and bitstream generation called vpr2bitstream. The bundle is organized into three major components: a VPR output processor, a routing graph translator, and a bitstream generator. The VPR output processor reads the BLIF netlist, packed netlist, placement file, and routing file and produces two output files: a logic data file summarizing information required for programming logic resources, and a mux select file summarizing information required for programming routing resources. The routing graph translator presented in section 4.2.2 reads the VPR routing graph file and produces three files used by the bitstream generator. The mux map file defines the mapping of routing multiplexer select bits to address numbers and bit positions within a tile. The mux list file defines a list of multiplexers available in the FPGA that map directly to VPR routing graph nodes, used in the routing file to define how signals are routed. This file also defines which routing nodes are equivalent to the multiplexer inputs. The mux list map file addresses connections between IOBs and non-IOBs. The matching of these connections in the VPR routing file to multiplexers in the mux list file must be treated as a special case. In addition to files used by the bitstream generator, the routing graph translator produces the routing template file. The bitstream generator takes output files from VPR and the routing graph translator hand-generated map files defining the programming word address mappings for the I/O blocks, CLB LUTs, flip-flop muxes, and input crossbars, along with a pin definition file defining the programming of the IOBs, and produces a formatted bitstream file defining the programming word values for each word in each tile in the array.

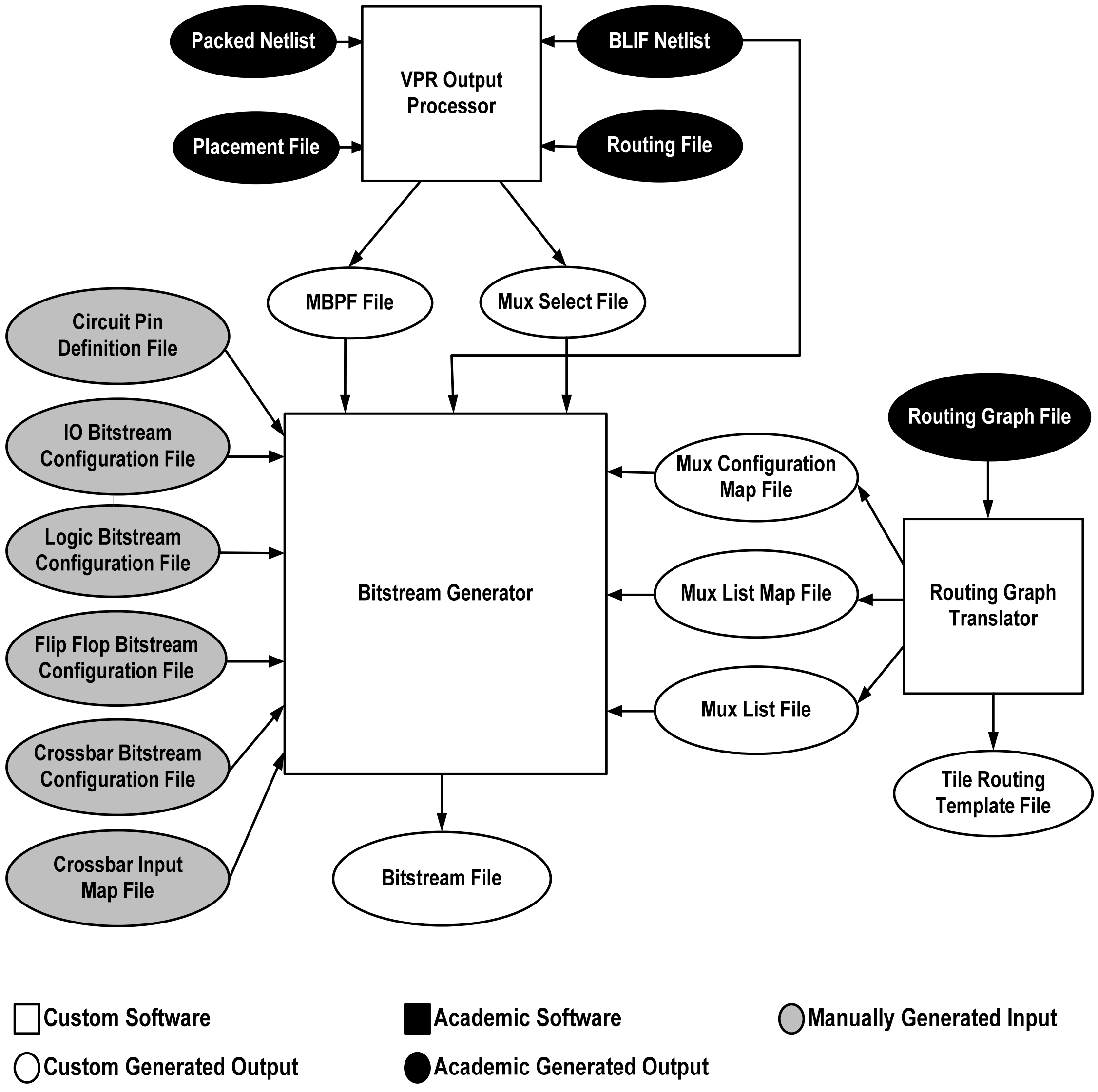


Figure 4.10. Flow diagram for routing graph translation with bitstream generation.

4.5.2 Verilog Testbench Design

A single, simple, flexible testbench was designed to permit the same Verilog testbench to be reused not only across all benchmarks, but on FPGAs of arbitrary array dimensions. This was done to support minimum energy analysis on a 30x30 array as well as a 6x6 array. In addition to making heavy use of command line `+define` arguments to set FPGA size and I/O counts, it also uses `` `include `` statements to import two components:

- A file must be provided containing the instantiation of the FPGA
- A file with a set of task calls to the program_byte task defined in the testbench. In this work, this file was auto-generated using a Perl script from the output of the bitstream generator described in section 4.5.1.

The testbench also requires a standard delay format (SDF) file, an industry-standard text file for back-annotated timing. Because complete FPGA programming is simulated, a nonzero-delay timing model is required for FPGA simulation. This is needed to avoid infinite loops in simulation event scheduling caused by intermediate FPGA programming states that may contain combinational loops. Because leakage power is state-dependent, SDF back-annotation has the added benefit of improving power estimation accuracy.

Finally, the testbench also requires a file to load a set of input stimului applied after the FPGA has been programmed. Random input vectors were generated on a per benchmark basis to ensure that the the same input vectors were applied to the same benchmark inputs regardless of the place and route solution generated. Since the file read by the testbench assigns vectors to FPGA inputs, not benchmark inputs, a Perl script was responsible for translating the vectors for each benchmark circuit to a place-and-route solution-specific vector file to be read by the testbench. FPGA I/Os not used by the benchmark circuit were held constant at zero, meaning that input activity factor was a strong function of benchmark input count. The number of input vectors provided is parameterizable and was set to 1000 for these experiments.

The testbench contains standard function calls used to generate the VCD file to be read by RTL Compiler. Since the goal of power estimation is to report power during benchmark circuit operation, and not during programming, the calls are organized so that activity information is not generated during programming.

4.5.3 Benchmarks Used in Simulation

The ISCAS '85 benchmarks used in this work were obtained as part of the IWLS 2005 benchmark suite [66]. A complete list of the benchmarks used is provided as Appendix C. . Each of the 21 circuits used contains both combinational logic and flip-flops. The number of 4-LUTs required to map the benchmarks ranges from 7 to 262 of 288 LUTs available, providing a full range of utilization cases. The I/O count ranges from five to 30 of 48. The utilization statistics for each benchmark are provided in section 4.8. Each benchmark was simulated separately at each voltage of interest.

4.6 Critical Path Delay Estimation

Figure 4.11 displays a flow diagram showing how this work appends voltage-specific timing information from standard cell library characterization results to VPR architecture files. For each timing path defined in the architecture file, a timing report is generated using the static timing analysis engine in Cadence RTL Compiler. The information from all timing reports at a particular voltage is post-processed and summarized into a single file. This file is then read by an annotation script that takes a template VPR architecture file as input and outputs a timing-annotated VPR architecture file. By performing static timing analysis on the relevant timing paths of the FPGA netlist, the timing information in the VPR architecture is matched to the timing information used in the gate-level Verilog simulations, ensuring a consistent approach between activity information generation and maximum clock frequency estimation.

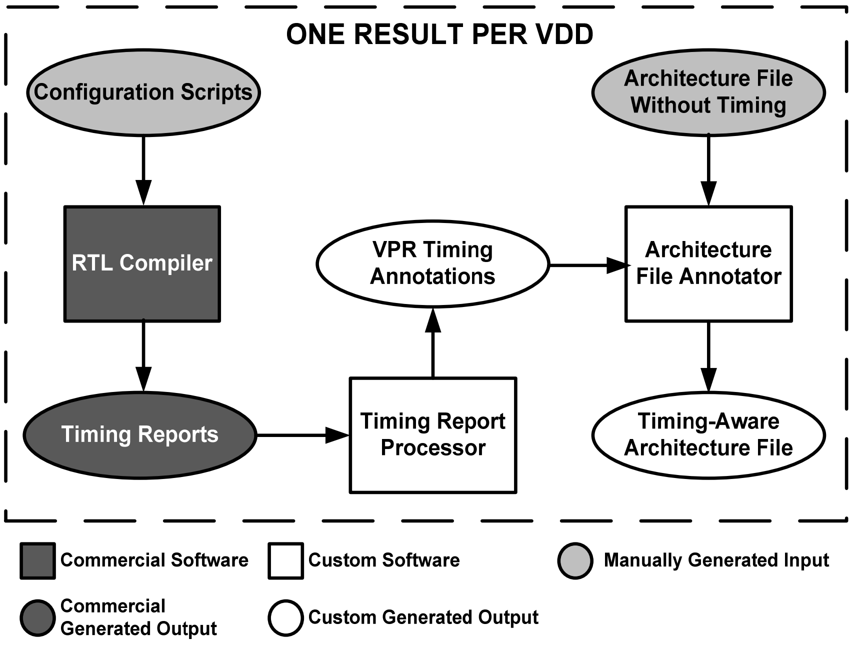


Figure 4.11. Flow diagram for annotating VPR architecture files with RTL Compiler-based timing information

4.7 Power Consumption Estimation

With the setup described in previous sections, the power estimation step becomes straightforward. Figure 4.12 shows the RTL Compiler script used to estimate power consumption for a single benchmark at a single supply voltage:

```
set_attribute library FPGA_STDLIB_L100_subvt_range_tt_VDD_04.lib
read_netlist ../../simulation/sdf/fpga_6x6_4_8_20_2_loop_break.v
read_vcd -static -vcd_module
dut ../../simulation/benchmark_analysis_6x6_20/s1196/alpha_0.25/pg8_6x6_t
t_VDD04/dump.vcd.gz
report power -verbose -depth 1 >
s1196_alpha_0.25_pg8_6x6_tt_VDD04_.gz.pwr
report power -flat -verbose >
s1196_alpha_0.25_pg8_6x6_tt_VDD04_.gz.pwr.test
```

```
exit
```

Figure 4.12. Sample RTL Compiler power analysis script

For ASICs, only four commands are required. First, the cell library for the voltage of interest must be read in. Then the Verilog netlist must be read, followed by the VCD file. A report power command with any desired options and redirection of the output to a text file enables the power information to be aggregated by post-processing scripts following power analysis at multiple supply voltages.

The setup is slightly more complicated for FPGAs because FPGAs contain combinational loops—combinational circuits with feedback. These occur in the both the CLB, where it is possible to program the FPGA such that a LUT output feeds back to an input of the same LUT without using a flip-flop; and in the routing fabric, where the network of signal choices contains paths through chains of multiplexers that are loops. Combinational loops are not permitted in ASICs using synthesis tools such as RTL Compiler to generate hardware. When RTL Compiler encounters combinational loops, it inserts buffers, referred to as loop breakers, to mark stop points for performing static timing analysis and propagating activity during power analysis. Through experimentation, it was determined that for consistency between SDF back-annotation and activity propagation, it was necessary to perform simulation and power analysis on a netlist that included the loop breakers, and then subtract the power consumption of the loop breakers for final minimum energy analysis. Because only buffers are inserted, the functionality of the circuit is unaltered. The loop breakers did not appear in FPGA critical paths used to annotate the VPR architecture file, indicating that they had no impact on critical path estimation.

An additional `report power` command is required with the `-flat` option to obtain power estimates for the loop breakers. A consistent naming convention used by RTL Compiler for the loop breaker instance names makes aggregating the loop breaker power contribution straightforward.

4.8 Minimum Energy Point Analysis Results

To pipeclean the analysis flow and to gain a first impression of how the power delay product and minimum energy point might shift based on FPGA usage, minimum energy analysis was performed on a single tile across a wide range of input activity factors. The tile was selected randomly from a place and route solution for the s526 benchmark, a medium-sized benchmark amongst those considered. The tile was programmed exactly as it would have been had it been used in a full benchmark simulation. Random input activity was added to all routing channel inputs of the tile as well as the CLB I/Os to emulate the switching activity of nets routed through the tile to other CLBs, and the input activity of both the CLB and routing channel inputs were set to equal values. The clock period of the tile was set to the critical path delay found during place and route of the full s526 benchmark.

Figure 4.13 shows the tile power delay product as a function of input activity factor. Throughout these results, activity factor is defined such that the activity factor of a clock signal is 1. The expected linear trend is clearly observable. The PDP increases by a factor of 3.5 over the range of activity factors explored.

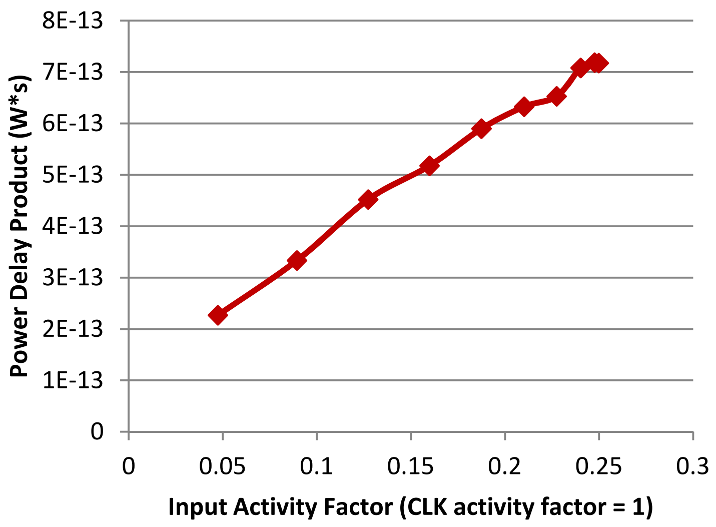


Figure 4.13. Single-tile power delay product vs. activity factor, VDD=0.4V

Figure 4.14 shows the power delay product versus supply voltage, with constant input activity factor of 0.19. This choice is based on setting the probability of an input vector being zero to 0.25. The PDP minimum is observable at 0.36V. A second local minimum occurs at 0.41V. The existence of multiple minima in a PDP vs. supply voltage plot has not previously been reported for ASIC circuits. Based upon the consistency of the power and delay trends present in the Liberty files at each voltage tried, it appears that the source of the multiple minima is variation in the energy efficiency of the place and route solutions obtained by VPR. Variation in routing solution quality in VPR has been observed previously, primarily in the context of critical path delay [67]. This makes variation in energy efficiency for a given benchmark circuit less unexpected.

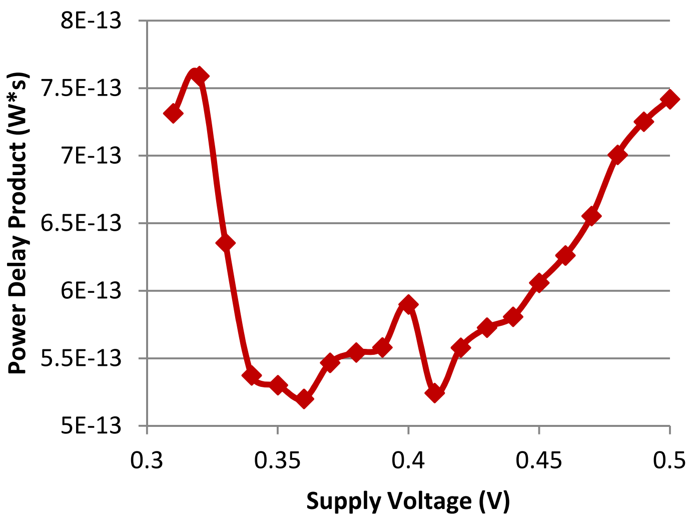


Figure 4.14. Single-tile power delay product vs. supply voltage, input activity factor = 0.19.

Figure 4.15 shows the minimum energy point for the tile as a function of the input activity factor. Across the range of activity factors tried, the minimum energy point varies from 0.34V to 0.43V. As expected, the minimum energy point increases as activity decreases. At an input activity factor of 0.05, the minimum energy point is near or slightly above the threshold voltage of the transistors used.

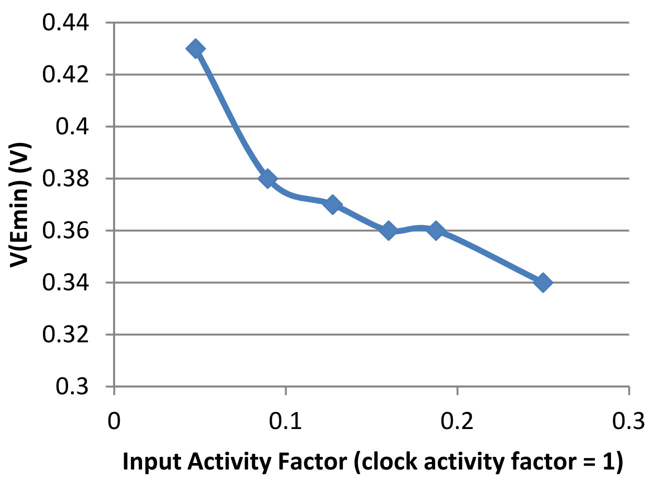


Figure 4.15. Minimum energy supply voltage vs. activity factor, single tile

Table 4.3 summarizes the minimum energy points for all benchmarks simulated. To provide a basis by which to compare the benchmark circuits, the number of 4-LUTs they occupied, the number of inputs, and the maximum clock frequency obtained at 0.5 V are provided. The utilization ranges from 7 to 262 of a possible 288 4-LUTs. The average minimum energy voltage across all benchmarks was 0.49 V. The lowest minimum energy supply was 0.42 V, while the highest was 0.54 V. The benchmarks thus all attained peak energy efficiency at voltages slightly above the threshold of the transistors used.

Table 4.3. Minimum energy point estimates for 21 ISCAS ’85 Benchmark Circuits

Benchmark	# LUTs used	#Inputs	Max. Clock Frequency at 0.5V (kHz)	PDP at V(Emin) (W*s)	Minimum Energy Point (V)
s1196	228	16	43.7	2.38E-11	0.42
s1238	247	16	42.3	2.45E-11	0.42
s1423	183	19	39.4	1.10E-11	0.44
s1488	261	10	56.7	7.22E-12	0.48
s1494	262	10	54.9	7.11E-12	0.54
s208_1	25	12	122.6	3.64E-12	0.44
s27	7	6	230.8	2.21E-12	0.48

Benchmark	# LUTs used	#Inputs	Max. Clock Frequency at 0.5V (kHz)	PDP at V(Emin) (W*s)	Minimum Energy Point (V)
s298	44	5	123.4	3.24E-12	0.54
s344	46	11	143.7	3.51E-12	0.48
s349	46	11	137.3	4.05E-12	0.52
s382	55	5	107.8	3.12E-12	0.5
s386	62	9	102.7	3.96E-12	0.5
s400	54	5	116.7	2.93E-12	0.5
s420_1	60	20	79.0	5.43E-12	0.5
s444	54	5	107.6	3.40E-12	0.48
s510	104	21	87.1	4.60E-12	0.5
s526	73	5	111.0	3.47E-12	0.5
s526n	73	5	104.7	3.24E-12	0.52
s820	132	20	74.5	6.51E-12	0.52
s832	130	20	72.7	6.52E-12	0.48
s838_1	124	36	57.0	8.35E-12	0.54

Figure 4.16 plots normalized power delay product versus supply voltage for five of the 21 benchmarks analyzed. Each benchmark's power delay product is normalized to the mnimum PDP obtained for that benchmark, so that all minimum energy points appear at $y = 1$. These five have been chosen as a representative sample of the full set, containing both larger and smaller circuits, and a full range of shapes of the curves. The energy efficiency of all benchmarks below 0.4 V degrades quickly. Above 0.4 V, a range of results are obtained. For most benchmarks, including those in Figure 4.16 other than s349, the PDP remains close to the energy minimum over a small voltage range, typically +/- 0.02-0.04 V. However, a few benchmarks, such as s349, exhibit a clearer minimum energy point. The smoothness of the PDP plots also varies, with s1238 being a particularly smooth example and s1494 being a particularly bumpy example. The implications of this are less clear, other than to observe that FPGAs exhibit a much more complicated energy efficiency vs. voltage relationship than ASICs due to not only to circuit properties, but to the influence of CAD tools on their performance.

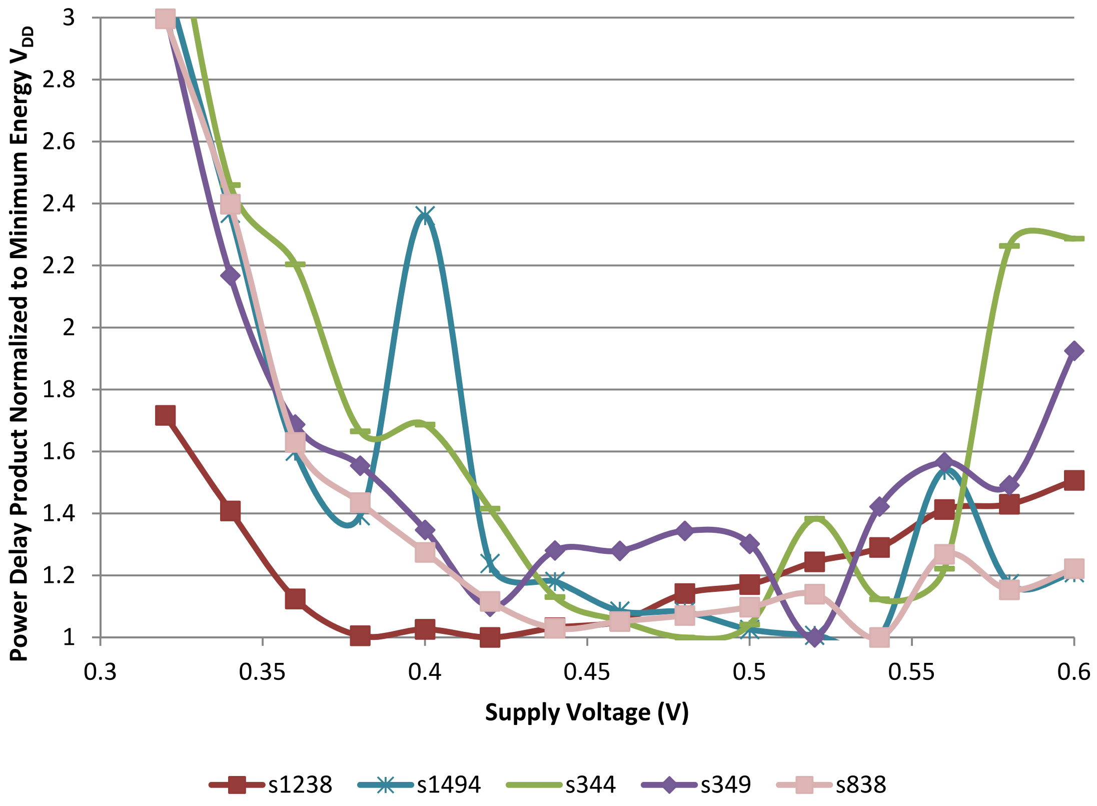


Figure 4.16. Normalized power delay product vs. supply voltage for five ISCAS '85 benchmarks.

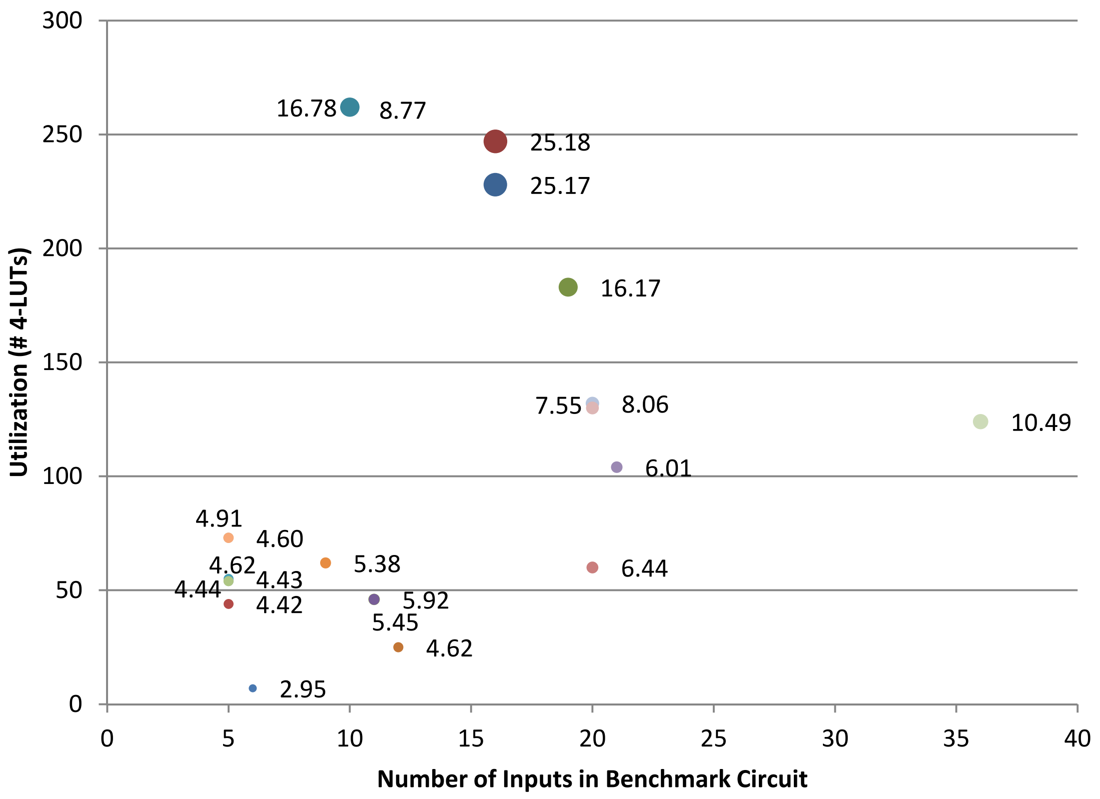


Figure 4.17. Power delay product (in pJ) at V_{DD} = 0.4 V as a function of benchmark input count and 4-LUT utilization for 21 ISCAS '85 benchmarks.

Figure 4.17 shows the power delay product at 0.4 V for each benchmark circuit as a function of both the number of inputs in the benchmark and the number of 4-LUTs that the benchmark requires. A number of benchmarks with simular input and LUT counts, clustered in the lower left corner, show PDPs with similar values, trending generally upward with both input count and utilization. This matches intuition—the FPGA should use more energy per clock cycle as the number of LUTs (and presumably the number of muxes in the routing fabric) are used, and as the number of toggling inputs increases. The trends continue away from this cluster as well, with one somewhat surprising result that two benchmarks with identical LUT counts and input counts have PDPs that differ by a factor of two. This outlying data point and the lack of a clear gradient to this plot offer reminders that factors other than just utilization and input count can have a significant impact on FPGA energy consumption.

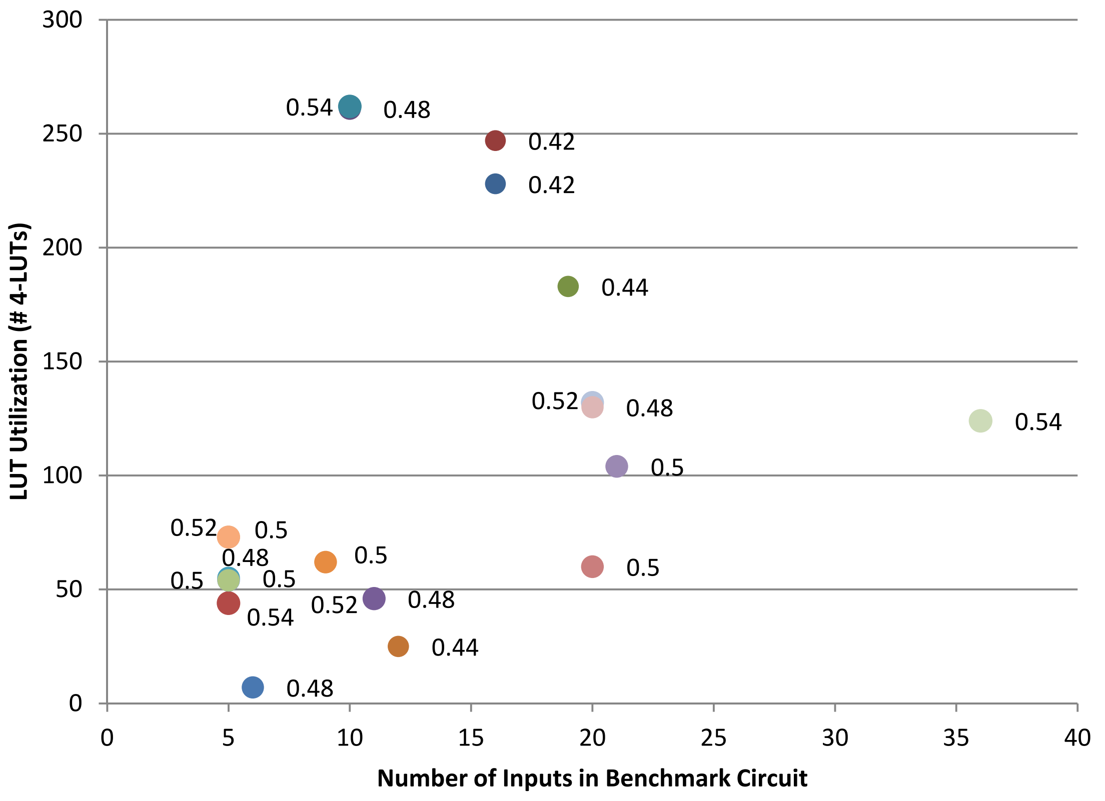


Figure 4.18. Minimum energy point (in V) as a function of benchmark input count and 4-LUT utilization for 21 ISCAS '85 benchmarks.

Figure 4.18 shows a plot of the minimum energy points listed in Table 4.3 as a function of benchmark input count and LUT utilization, as with the PDP plot in Figure 4.17. Unlike the PDP at 0.4 V, the minimum energy point exhibits little or no relationship to these two parameters. While the 21 benchmarks exhibit a range of 0.12 V in minimum energy supply voltage, the benchmarks that share a common minimum energy point are sprinkled across the range of input counts and 4-LUT counts. The implication of this result is that properties of the benchmark circuit itself, as much as benchmark LUT or input counts, are influencing the energy efficiency of the FPGA.

Minimum energy points were obtained for the 21 benchmark circuits at three process corners: typical-typical (TT), shown above, fast-fast (FF), and slow-slow (SS). To obtain a sense of the power/delay range obtained across the three corners, Figure 4.19 plots the average power consumption vs. supply voltage of the s526 benchmark as a function of supply voltage for each corner, while Figure 4.20 plots the maximum clock frequency vs. supply voltage for each corner. For both plots the y-axis is given on a logrithmic scale. As expected, the spread across corners increases as the voltage decreases for both delay and power. However, as shown in Figure 4.21, the power delay product remains within a tight range across the corners. While the operating point of the FPGA is a strong function of process corner, the energy efficiency at the minimum clock period remains similar.

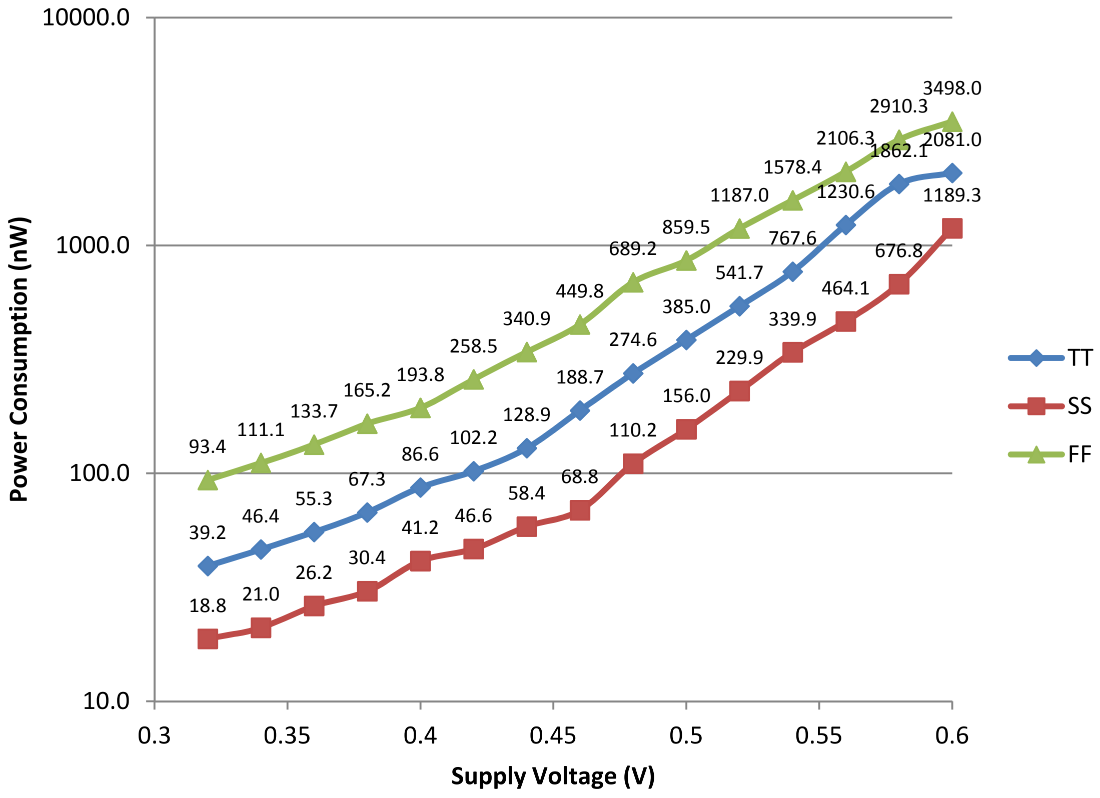


Figure 4.19. Average power consumption vs. supply voltage for s526 benchmark

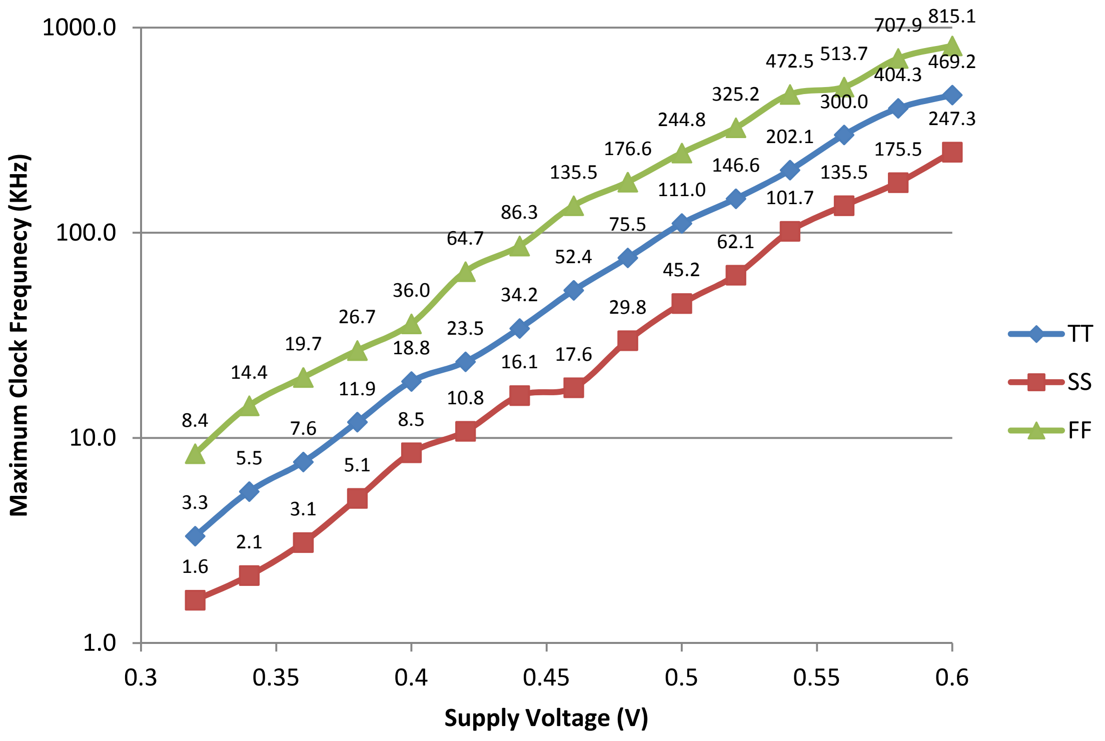


Figure 4.20. Maximum clock frequency vs. supply voltage for s526 benchmark

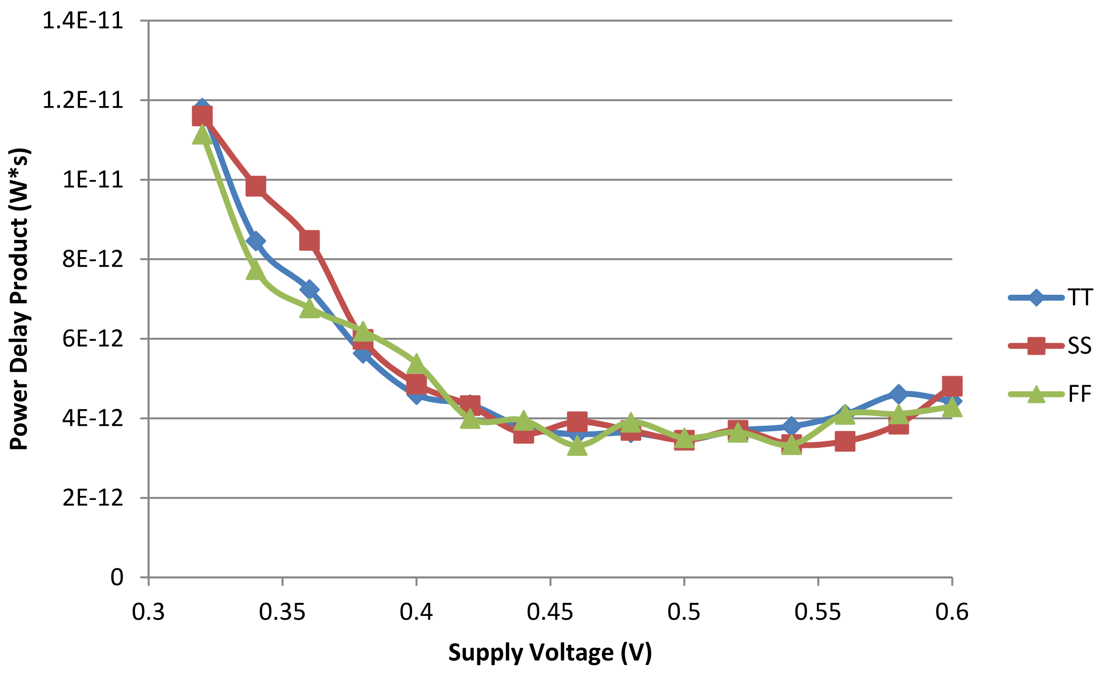


Figure 4.21. Power delay product vs. supply voltage at process corners, s526 benchmark

Table 4.4 compares the minimum energy point of each benchmark circuit across the three process corners simulated. Some benchmarks exhibit the same minimum energy voltage at all three, while others vary by as much as 0.08V. Variation in routing solution quality is almost certainly contributing to these results. There are no strong trends in the spread, and the average minimum energy point across all benchmarks is within 0.02V for all corners. Operation in the fast-fast process corner thus offers increased performance, but not increased energy efficiency. Similarly, operation in the slow-slow corner offers power savings, but not decreased energy efficiency. Although process corner analysis is critical for subthreshold circuit design due to the large variation in power and maximum clock frequency shown in Figure 4.19 and Figure 4.20, it does not appear to strongly influence FPGA energy efficiency.

Table 4.4. Minimum energy point comparison across process corners, 21 ISCAS '85 benchmarks

			Minimum Energy Point (V)		
Benchmark	# LUTs	#Inputs	TT Corner	FF Corner	SS Corner
s1196	228	16	0.42	0.4	0.42
s1238	247	16	0.42	0.42	0.42
s1423	183	19	0.44	0.46	0.52
s1488	261	10	0.48	0.46	0.5
s1494	262	10	0.54	0.48	0.5
s208_1	25	12	0.44	0.46	0.44
s27	7	6	0.48	0.44	0.48
s298	44	5	0.54	0.46	0.48
s344	46	11	0.48	0.5	0.54
s349	46	11	0.52	0.48	0.5
s382	55	5	0.5	0.48	0.5
s386	62	9	0.5	0.5	0.52
s400	54	5	0.5	0.5	0.54
s420_1	60	20	0.5	0.46	0.52
s444	54	5	0.48	0.44	0.5
s510	104	21	0.5	0.52	0.5
s526	73	5	0.5	0.46	0.54
s526n	73	5	0.52	0.48	0.48
s820	132	20	0.52	0.5	0.44
s832	130	20	0.48	0.48	0.54
s838_1	124	36	0.54	0.52	0.46
Average Minimum Energy Point			**0.49**	**0.47**	**0.49**

4.9 Summary

This chapter presented a first step toward quantifying how the minimum energy point of an FPGA can vary across a range of possible configurations. This was done with an eye toward design and fabrication of a future subthreshold FPGA, so power modeling within FPGA CAD tools was avoided in favor of performing power analysis using an industry standard ASIC tool flow augmented by a VPR-based FPGA CAD flow to support simulations and to perform timing analysis.

Several aspects of the experimental setup required special attention. Cell library characterization required parameter tuning in order to obtain good results across a full range of subthreshold and near-threshold supply voltages. The FPGA CAD flow required development of a bitstream generator to post-process VPR output and generate configuration data for programming the FPGA in simulations. To assist in the implementation of schematics for an FPGA that was compatible with VPR, a template generator for VPR routing fabrics was developed. The simulation testbench required adaptability to simulate an FPGA programmed with different benchmark circuits. Both simulation and power analysis required accounting for the presence of combinational loops in the FPGA netlist. Supporting the timing analysis engine within VPR across a range of supply voltages motivated automating the translation of RTL Compiler timing report information into timing information with voltage-specific VPR architecure files. Combining timing analysis results from VPR with power estimates from RTL Compiler enabled generation of plots of power delay product (PDP) versus supply voltage for each of 21 ISCAS '85 benchmarks.

The results obtained from these PDP vs. voltage plots quantified the expected trends in PDP as a function of utilization-based metrics of FPGA activity that vary from benchmark to benchmark—specifically, the number of 4-LUTs and the number of inputs required for each benchmark. Across the 21 benchmarks tried, the PDP at 0.4 V spanned nearly an order of magnitude. The minimum energy point varied from 0.42-0.54 V amongst the 21 benchmarks. However, the minimum energy point values did not strongly correlate with circuit utilization, indicating that the energy efficiency of the benchmark circuit itself, in addition to FPGA architecture parameters, can significantly impact the FPGA minimum energy point. This, in turn, implies that for an FPGA platform to properly serve a wide range of applications, programmable voltage regulators should be investigated to determine whether the energy efficiency improvement obtained from voltage supply tuning exceeds the energy penalty of the regulator.

Analysis of the minimum energy point of each benchmark at fast-fast and slow-slow process corners revealed that while power and maximum clock frequency estimates for a benchmark could vary by 3-5X in the operating range of 0.3-0.6 V depending on the process corner, the power delay product varied only slightly. As a result, most benchmarks exhibit similar minimum energy points at both corners to their typical-typical minimum energy point.

5 CONCLUSIONS AND FUTURE WORK

This research showed that a conventional, academic FPGA architecture can be made to function at subthreshold supply voltages by following known rules of thumb for subthreshold CMOS circuit design and replacing SRAM with latches for configuration bit storage. With these techniques, a pair of FPGA test chips were fabricated in the IBM 0.18 µm SOI process—one with conventional multiplexers in the routing fabric, and one with DTMOS multiplexers. Correct functionality was obtained at voltages as low as 0.26 V for both test chips, and a ~40X power delay product reduction was observed for each test chip at 0.26 V vs. conventional multiplexers at 1.5 V. Eleven conventional multiplexer test chips had an average minimum operating voltage of 0.3 V, while four DTMOS multiplexer test chips had an average minimum operating voltage of 0.26 V. The lower average for the DTMOS chips is consistent with post-layout simulations indicating that the DTMOS multiplexers are less sensitive to process variation. When configured as arrays of four-bit counters to maintain a high chip activity factor, the minimum energy point of both test chips is well below threshold.

This research implements a technique for pre-fabrication minimum energy estimation of FPGAs at the full-chip level. The technique consists of combining FPGA programming software with integrated circuit verification software and additional custom scripts. By providing complete CAD support for both mapping of benchmark circuits to the FPGA and gate-level Verilog simulation of the FPGA across a range of supply voltages, the power delay product of the FPGA for a specific use case can be plotted as a function of voltage. This enables estimation of the minimum energy point. For this research, power delay product vs. voltage was plotted from 0.3-0.6 V for 21 ISCAS '85 benchmarks on an FPGA consisting of 288 4-LUTs and 48 I/Os. The benchmark circuits had minimum energy points ranging from 0.42-0.54 V. The minimum energy point was a property of the benchmark circuit rather than a function of its FPGA utilization. The range of values obtained indicate the potential benefit of a programmable supply voltage on FPGA platforms seeking to optimize energy consumption across a range of use cases.

5.1 Future Work

Numerous opportunities exist to extend this work. These include:

- *Incorporate a power model into VPR 6.0 and compare results*: The power model developed by Poon et al. for VPR 4.3 [68] and incorporated into VPR 5.0 by Jamieson et al. [61] provides a basis by which minimum energy analysis can be directly incorporated into an FPGA CAD flow. Incorporation of a power model into VPR 6.0 presents new challenges not faced in previous versions due to the new support in VPR 6.0 for architectures containing memories and other hard IP blocks. Such challenges were well beyond the scope of this work. Even adapting the power model for minimum energy analysis on older VPR versions requires a number of enhancements to the modeling. Foremost, a strategy must be devised for integrating power estimation at multiple supply voltages into the CAD software. Supporting multiple alternative circuit implementations for a given FPGA architecture, which has not been considered previously, is also a likely requirement. VPR supports not just a single FPGA, but an entire class of FPGAs that in general might operate at superthreshold or subthreshold supply voltages. As demonstrated in Chapter 3, a given architecture is likely to be implemented with different circuitry when targeting near- or sub-threshold supply voltages instead of nominal supplies. This in turn will impact how the power model estimates net capacitance in the absence of implementation-specific details. While not required for many FPGA architecture studies, power modeling support across a range of process technologies is highly desirable for subthreshold FPGA study; at very low supply voltages, the best choice for process technology is much less clear than at nominal supplies. Such considerations may be beyond the scope of VPR's use as an architecture evaluation tool. In developing a minimum energy analysis capability for VPR, a good starting point would be to simply pick a process technology node.

- *Explore minimum energy point sensitivity to FPGA architecture parameters*. The low power FPGA studies presented in Chapter 2 analyzed a wide variety of cluster sizes, LUT sizes, and varied other architecture parameters as well. An architecture study of this type is needed for FPGAs to consider how architecture parameters can be tuned to reduce minimum energy supply voltage, thereby reducing static power for FPGAs running at optimal energy efficiency.

- *Study impact of I/O level shifters on FPGA energy efficiency.* In the interest of managing the scope of this work, the I/O blocks were radically simplified compared to a commercial FPGA. Of most signficance to power analysis, higher-voltage circuitry typically present in commercial I/O blocks to support various commercial standard I/O levels was deferred to future work. In general, low voltage FPGAs will not always have the luxury of interfacing to other low voltage parts, and support for I/O levels such as 1.2 V, 1.5 V, 1.8 V, 2.5 V, and perhaps even 3.3 V may be required. For an FPGA running at 0.5 V or below, the level shifter circuits and output drivers required to these logic levels will consume a significant amount of power relative to the on-chip logic. Future work should attempt to quantify the impact of this requirement.
- *Study the energy efficiency tradeoff between providing programmable voltage regulators to tune FPGA supply voltage vs. a fixed supply.* The results obtained in Chapter 4 indicated a range of 0.12 V for the minimum energy point across all benchmarks, but also a smaller range, V_{DD} = 0.45-0.5 V, for which most benchmarks remained within 20% of their minimum. Future work could investigate whether the energy penalty for providing a programmable voltage regulator provides a net energy benefit over choosing a fixed supply voltage for which most benchmarks are near optimal. To do this, additional circuit-level research in accurately programmable low voltage, low power voltage regulators is also needed.
- *Investigate benefits of aggressive pipelining for improving FPGA energy efficiency.* It has been shown for microprocessors that subthreshold circuits gain

an energy efficiency benefit from being pipelined more aggressively than their superthreshold counterparts [69]. For FPGAs, there are at least two major implications for this: first, that circuits mapped to FPGAs are also likely to benefit from more aggressive pipelining; second, that subthresold FPGAs themselves may benefit from more aggressive pipelining. Pipelined routing fabrics have been evaluated for superthreshold FPGAs [70], but have not been considered for minimum energy operation.

References

[1] X. Liu, Y. Zheng, M. W. Phyu, F. N. Endru, V. Navaneethan, and B. Zhao, “An Ultra-Low Power ECG Acquisition and Monitoring ASIC System for WBAN Applications,” *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 2, no. 1, pp. 60 –70, Mar. 2012.

[2] S. Consul-Pacareu, J. Albo-Canals, X. Vilasis-Cardona, and J. Riera-Babures, “High performance DT-CNN camera device design on ACTEL IGLOO low power FPGA,” in *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, 2011, pp. 37 –40.

[3] IGLOO Handbook, *Actel Corporation*. Mountain View, CA: , 2008.

[4] Texas Instruments, “MSP430G22x0 Mixed Signal Microcontroller Datasheet.” [Online]. Available: http://www.ti.com/general/docs/lit/getliterature.tsp?literatureNumber=slas753e&fileType=pdf. [Accessed: 26-Feb-2013].

[5] T. F. Al-Somani and H. Houssain, “Implementation of GF(2^m) Elliptic Curve cryptoprocessor on a Nano FPGA,” in *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, 2011, pp. 7 –12.

[6] D. Blaauw and B. Zhai, “Energy efficient design for subthreshold supply voltage operation,” in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, 2006, p. 4 pp.–32.

[7] A. Wang, *Sub-threshold design for ultra low-power systems*. New York: Springer, 2006.

[8] J. F. Ryan and B. H. Calhoun, “A sub-threshold FPGA with low-swing dual-VDD interconnect in 90nm CMOS,” in *Custom Integrated Circuits Conference (CICC), 2010 IEEE*, 2010, pp. 1–4.

[9] B. H. Calhoun, D. C. Daly, N. Verma, D. F. Finchelstein, D. D. Wentzloff, A. Wang, S.-H. Cho, and A. P. Chandrakasan, “Design considerations for ultra-low energy wireless microsensor nodes,” *Computers, IEEE Transactions on*, vol. 54, no. 6, pp. 727–740, Jun. 2005.

[10] Bo Zhai, S. Pant, L. Nazhandali, S. Hanson, J. Olson, A. Reeves, M. Minuth, R. Helfand, T. Austin, D. Sylvester, and D. Blaauw, “Energy-Efficient Subthreshold Processor Design,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 8, pp. 1127–1137, 2009.

[11] A. Wang and A. Chandrakasan, “A 180-mV subthreshold FFT processor using a minimum energy design methodology,” *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 310–319, 2005.

[12] B. Devlin, M. Ikeda, and K. Asada, “Energy minimum operation in a reconfigurable gate-level pipelined and power-gated self synchronous FPGA,” in *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, 2011, pp. 3 –8.

[13] S. Jain, S. Khare, S. Yada, V. Ambili, P. Salihundam, S. Ramani, S. Muthukumar, M. Srinivasan, A. Kumar, S. K. Gb, R. Ramanarayanan, V. Erraguntla, J. Howard, S. Vangal, S. Dighe, G. Ruhl, P. Aseron, H. Wilson, N. Borkar, V. De, and S. Borkar, “A 280mV-to-1.2V wide-operating-range IA-32 processor in 32nm CMOS,” in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, 2012, pp. 66 –68.

[14] J. Kwong, Y. K. Ramadass, N. Verma, and A. P. Chandrakasan, “A 65 nm Sub-V_{t} Microcontroller With Integrated SRAM and Switched Capacitor DC-DC Converter,” *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 1, pp. 115–126, 2009.

[15] J. T. Kao, M. Miyazaki, and A. R. Chandrakasan, “A 175-MV multiply-accumulate unit using an adaptive supply voltage and body bias architecture,” *Solid-State Circuits, IEEE Journal of*, vol. 37, no. 11, pp. 1545–1554, 2002.

[16] Bo Zhai, S. Hanson, D. Blaauw, and D. Sylvester, “A Variation-Tolerant Sub-200 mV 6-T Subthreshold SRAM,” *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 10, pp. 2338–2348, 2008.

[17] N. Verma and A. P. Chandrakasan, “A 256 kb 65 nm 8T Subthreshold SRAM Employing Sense-Amplifier Redundancy,” *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 141–149, 2008.

[18] Ik Joon Chang, Jae-Joon Kim, S. P. Park, and K. Roy, “A 32 kb 10T Sub-Threshold SRAM Array With Bit-Interleaving and Differential Read Scheme in 90 nm CMOS,” *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 2, pp. 650–658, 2009.

[19] S. Okumura, Y. Iguchi, S. Yoshimoto, H. Fujiwara, H. Noguchi, K. Nii, H. Kawaguchi, and M. Yoshimoto, “A 0.56-V 128kb 10T SRAM using column line assist (CLA) scheme,” in *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, 2009, pp. 659–663.

[20] M. J. Deen, S. Naseh, O. Marinov, and M. Kazemeini, “Very low-voltage operation capability of complementary metal-oxide-semiconductor ring oscillators and logic gates.” [Online]. Available: http://scitation.aip.org/journals/doc/JVTAD6-ft/vol_24/iss_3/763_1.html. [Accessed: 18-Feb-2010].

[21] B. Graniello, “Subthreshold SOI Logic for Digital Integrated Circuits,” M.S. Thesis, University of Texas, El Paso, 2005.

[22] H. P. Alstad and S. Aunet, “Three Subthreshold Flip-Flop Cells Characterized in 90 nm and 65 nm CMOS Technology,” in *Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on*, 2008, pp. 1–4.

[23] S. Fisher, A. Teman, D. Vaysman, A. Gertsman, O. Yadid-Pecht, and A. Fish, “Ultra-low power subthreshold flip-flop design,” in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, 2009, pp. 1573–1576.

[24] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer, “High-performance CMOS variability in the 65-nm regime and beyond,” *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 433 –449, Jul. 2006.

[25] S. K. Saha, “Modeling Process Variability in Scaled CMOS Technology,” *IEEE Design Test of Computers*, vol. 27, no. 2, pp. 8 –16, Apr. 2010.

[26] B. H. Calhoun, S. Khanna, R. Mann, and Jiajing Wang, “Sub-threshold circuit design with shrinking CMOS devices,” in *IEEE International Symposium on Circuits and Systems, 2009. ISCAS 2009*, 2009, pp. 2541–2544.

[27] D. Bol, R. Ambroise, D. Flandre, and J.-D. Legat, “Impact of Technology Scaling on Digital Subthreshold Circuits,” in *Symposium on VLSI, 2008. ISVLSI ’08. IEEE Computer Society Annual*, 2008, pp. 179–184.

[28] S. Trimberger, *Field-programmable gate array technology*. Boston: Kluwer Academic Publishers, 1994.

[29] W. M. Fang and J. Rose, “Modeling routing demand for early-stage FPGA architecture development,” in *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays - FPGA ’08*, Monterey, California, USA, 2008, p. 139.

[30] G. Lemieux, E. Lee, M. Tom, and A. Yu, “Directional and single-driver wires in FPGA interconnect,” in *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*, 2004, pp. 41–48.

[31] A. DeHon, “Balancing interconnect and computation in a reconfigurable computing array (or, why you don’t really want 100% LUT utilization),” in *Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays*, Monterey, California, United States, 1999, pp. 69–78.

[32] G. Borriello, C. Ebeling, S. A. Hauck, and S. Burns, “The Triptych FPGA architecture,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 3, no. 4, pp. 491–501, 1995.

[33] D. C. Cronquist, C. Fisher, M. Figueroa, P. Franklin, and C. Ebeling, “Architecture design of reconfigurable pipelined datapaths,” in *Advanced Research in VLSI, 1999. Proceedings. 20th Anniversary Conference on*, 1999, pp. 23–40.

[34] M. Leeser, W. M. Meleis, M. M. Vai, S. Chiricescu, Weidong Xu, and P. M. Zavracky, “Rothko: a three-dimensional FPGA,” *Design & Test of Computers, IEEE*, vol. 15, no. 1, pp. 16–23, 1998.

[35] P. Jamieson, K. B. Kent, F. Gharibian, and L. Shannon, “Odin II-an open-source verilog HDL synthesis tool for CAD research,” in *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*, 2010, pp. 149–156.

[36] R. Brayton and A. Mishchenko, “ABC: an academic industrial-strength verification tool,” in *Proceedings of the 22nd international conference on Computer Aided Verification*, Berlin, Heidelberg, 2010, pp. 24–40.

[37] J. Rose, J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson, “The VTR project: architecture and CAD for FPGAs from verilog to routing,” in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, New York, NY, USA, 2012, pp. 77–86.

[38] J. Lamoureux and W. Luk, “An Overview of Low-Power Techniques for Field-Programmable Gate Arrays,” in *Adaptive Hardware and Systems, 2008. AHS ’08. NASA/ESA Conference on*, 2008, pp. 338–345.

[39] IGLOO nano Handbook, *Actel Corporation*. Mountain View, CA: , 2008.

[40] Lattice Semiconductor, “iCE40 LP Series Ultra Low-Power FPGA Family.” [Online]. Available: http://www.latticesemi.com/documents/iCE40LPdatasheet121003.pdf. [Accessed: 08-Feb-2013].

[41] V. George, *Low-energy FPGAs□: architecture and design*. Boston: Kluwer Academic, 2001.

[42] B. Devlin, J. MyeongGyu, T. Nakura, M. Ikeda, and K. Asada, “647 MHz, 0.642pJ/block/cycle 65nm self synchronous FPGA,” in *ESSCIRC, 2009. ESSCIRC ’09. Proceedings of*, 2009, pp. 156 –159.

[43] Bo Zhai, L. Nazhandali, J. Olson, A. Reeves, M. Minuth, R. Helfand, Sanjay Pant, D. Blaauw, and T. Austin, "A 2.60pJ/Inst Subthreshold Sensor Processor for Optimal Energy Efficiency," in *VLSI Circuits, 2006. Digest of Technical Papers. 2006 Symposium on*, 2006, pp. 154–155.

[44] B. H. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 9, pp. 1778 – 1786, Sep. 2005.

[45] D. Bol, R. Ambroise, D. Flandre, and J.-D. Legat, "Analysis and minimization of practical energy in 45nm subthreshold logic circuits," in *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, 2008, pp. 294–300.

[46] O. Andersson, S. M. Y. Sherazi, and J. N. Rodrigues, "Impact of switching activity on the energy minimum voltage for 65 nm sub-VT CMOS," in *NORCHIP, 2011*, 2011, pp. 1 –4.

[47] B. Mishra, C. Botteron, P. A. Farine, and B. M. Al-Hashimi, "Ultra Low Power Multi-Operand Adder architecture for subthreshold circuits," in *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2011, pp. 1 –4.

[48] K. Kim and V. D. Agrawal, "Minimum energy CMOS design with dual subthreshold supply and multiple logic-level gates," in *2011 12th International Symposium on Quality Electronic Design (ISQED)*, 2011, pp. 1 –6.

[49] S. Paul, R. Garg, S. P. Khatri, and S. Vaidya, "Design and implementation of a sub-threshold BFSK transmitter," in *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, 2009, pp. 664–672.

[50] A. K. Kureshi, N. Alam, M. Hasan, and T. Arslan, "Subthreshold deep submicron performance investigation of CMOS and DTCMOS biasing schemes for reconfigurable computing," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, 2009, pp. 2545–2548.

[51] Ameet Chavan, Gaurav Dukle, Ben Graniello, and Eric MacDonald, "Robust Ultra-Low Power Subthreshold Logic Flip-Flop Design for Reconfigurable Architectures," in *Reconfigurable Computing and FPGA's, 2006. ReConFig 2006. IEEE International Conference on*, 2006, pp. 1–7.

[52] A. Chavan, E. MacDonald, N. Liu, and J. Neff, "A novel floating gate circuit family with subthreshold voltage swing for ultra-low power operation," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, 2008, pp. 3354–3357.

[53] R. Sankaranarayanan and M. R. Guthaus, "A single-VDD ultra-low energy sub-threshold FPGA," in *2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC)*, 2012, pp. 219 –224.

[54] B. H. Calhoun, J. F. Ryan, S. Khanna, M. Putic, and J. Lach, "Flexible Circuits and Architectures for Ultralow Power," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 267–282, 2010.

[55] N. Mehta, R. Rubin, and A. DeHon, "Limit study of energy &#38; delay benefits of component-specific routing," in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, New York, NY, USA, 2012, pp. 97–106.

[56] K.-J. Lee, H. Park, J. Kong, and A. P. Chandrakasan, “Demonstration of a Subthreshold FPGA Using Monolithically Integrated Graphene Interconnects,” *IEEE Transactions on Electron Devices*, vol. 60, no. 1, pp. 383 –390, Jan. 2013.

[57] “Cadence Virtuoso Spectre Circuit Simulator.” [Online]. Available: http://www.cadence.com/products/cic/spectre_circuit/pages/default.aspx. [Accessed: 26-Feb-2013].

[58] J. Wang, S. Nalam, and B. H. Calhoun, “Analyzing static and dynamic write margin for nanometer SRAMs,” in *Low Power Electronics and Design (ISLPED), 2008 ACM/IEEE International Symposium on*, 2008, pp. 129 –134.

[59] F. Li, D. Chen, L. He, and J. Cong, “Architecture evaluation for power-efficient FPGAs,” in *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays*, Monterey, California, USA, 2003, pp. 175–184.

[60] A. G. Ye, “Using the Minimum Set of Input Combinations to Minimize the Area of Local Routing Networks in Logic Clusters Containing Logically Equivalent I/Os in FPGAs,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 1, pp. 95–107, 2010.

[61] P. Jamieson, W. Luk, S. J. E. Wilton, and G. A. Constantinides, “An energy and power consumption analysis of FPGA routing architectures,” in *International Conference on Field-Programmable Technology, 2009. FPT 2009*, 2009, pp. 324 –327.

[62] J. Zhou, S. Jayapal, B. Busze, L. Huang, and J. Stuyt, “A 40 nm Dual-Width Standard Cell Library for Near/Sub-Threshold Operation,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2569 –2577, Nov. 2012.

[63] P. Meinerzhagen, O. Andersson, Y. Sherazi, A. Burg, and J. Rodrigues, “Synthesis strategies for sub-VT systems,” in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, 2011, pp. 552 –555.

[64] J. Luu and J. Rose, “VPR 6.0 User Manual.” [Online]. Available: http://code.google.com/p/vtr-verilog-to-routing/downloads/detail?name=VPR_User_Manual_6.0.pdf. [Accessed: 10-Aug-2012].

[65] “OpenSource Liberty.” [Online]. Available: http://www.opensourceliberty.org/. [Accessed: 07-Feb-2013].

[66] “IWLS 2005 Benchmarks.” [Online]. Available: http://iwls.org/iwls2005/benchmarks.html. [Accessed: 26-Feb-2013].

[67] R. Y. Rubin and A. M. DeHon, “Timing-driven pathfinder pathology and remediation: quantifying and reducing delay noise in VPR-pathfinder,” in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, New York, NY, USA, 2011, pp. 173–176.

[68] K. K. W. Poon, S. J. E. Wilton, and A. Yan, “A detailed power model for field-programmable gate arrays,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, no. 2, pp. 279–302, 2005.

[69] D. Jeon, M. Seok, C. Chakrabarti, D. Blaauw, and D. Sylvester, “A Super-Pipelined Energy Efficient Subthreshold 240 MS/s FFT Core in 65 nm CMOS,” *Solid-State Circuits, IEEE Journal of*, vol. 47, no. 1, pp. 23 –34, Jan. 2012.

[70] A. Sharma, K. Compton, C. Ebeling, and S. Hauck, “Exploration of pipelined FPGA interconnect structures,” in *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, New York, NY, USA, 2004, pp. 13–22.

Appendix A Glossary of Abbreviations

BLE – Basic Logic Element
CB – Connection Box
CLB – Configurable Logic Block
CMOS – Complementary Metal-Oxide Semiconductor
DTMOS – Dynamic Threshold MOS
FPGA – Field Programmable Gate Array
IOB – I/O Block
SB – Switch Box
SoC – System on a Chip
SRAM – Static Random Access Memory
VPR – Versatile Place and Route

Appendix B List of Standard Cells Used in FPGA Minimum Energy Analysis

and2_1x	two-input AND gate, unit drive strength
dffr	static D flip-flop with gated feedback and active low asynchronous reset
ilatch_6t	interruptible latch
inv_1x	inverter, unit drive strength
inv_2x	inverter, 2X drive strength
inv_4x	inverter, 4X drive strength
inv_8x	inverter, 8X drive strength
inv_32x	inverter, 32X drive strength
mux2_1x	two-input ganged tri-state multiplexer, unit drive strength, with embedded inverters for select inversion and non-inverting output generation
nand2_1x	two-input NAND gate, unit drive strength
nmux2_1x	two-input ganged-tri-state multiplexer, unit drive strength, inverting output
nor2_1x	two-input NOR gate, unit drive strength
or2_1x	two-input OR gate, unit drive strength
tinv_1x	tri-state inverter, unit drive strength
tinv_32x	tri-state inverter, 32X drive strength

Appendix C List of ISCAS ’85 Benchmarks Used in FPGA Minimum Energy Analysis

s1196
s1238
s1423
s1488
s1494
s208_1
s27
s298
s344
s349
s382
s386
s400
s420_1
s444
s510
s526
s526n
s820
s832
s838_1