Design Methodologies and CAD Tools for Leakage Power Optimization in FPGAs

by

Hassan Hassan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2008

Abstract

The scaling of the CMOS technology has precipitated an exponential increase in both subthreshold and gate leakage currents in modern VLSI designs. Consequently, the contribution of leakage power to the total chip power dissipation for CMOS designs is increasing rapidly, which is estimated to be 40% for the current technology generations and is expected to exceed 50% by the 65nm CMOS technology. In FPGAs, the power dissipation problem is further aggravated when compared to ASIC designs because FPGA use more transistors per logic function when compared to ASIC designs. Consequently, solving the leakage power problem is pivotal to devising power-aware FPGAs in the nanometer regime. This thesis focuses on devising both architectural and CAD techniques for leakage mitigation in FPGAs. Several CAD and architectural modifications are proposed to reduce the impact of leakage power dissipation on modern FPGAs.

Firstly, multi-threshold CMOS (MTCMOS) techniques are introduced to FPGAs to permanently turn OFF the unused resources of the FPGA, FPGAs are characterized with low utilization percentages that can reach 60%. Moreover, such architecture enables the dynamic shutting down of the FPGA idle parts, thus reducing the standby leakage significantly. Employing the MTCMOS technique in FPGAs requires several changes to the FPGA architecture, including the placement and routing of the sleep signals and the MTCMOS granularity. On the CAD level, the packing and placement stages are modified to allow the possibility of dynamically turning OFF the idle parts of the FPGA. A new activity generation algorithm is proposed and implemented that aims to identify the logic blocks in a design that exhibit similar idleness periods. Several criteria for the activity generation algorithm are used, including connectivity and logic function. Several versions of the activity generation algorithm are implemented to trade power savings with runtime. A newly developed packing algorithm uses the resulting activities to minimize leakage power dissipation by packing the logic blocks with similar or close activities together. By proposing an FPGA architecture that supports MTCMOS and developing a CAD tool that supports the new architecture, an average power savings of 30% is achieved for a 90nm CMOS process while incurring a speed penalty of less than 5%. This technique is further extended to provide a timing-sensitive version of the CAD flow to vary the speed penalty according to the criticality of each logic block.

Secondly, a new technique for leakage power reduction in FPGAs based on the use of input dependency is developed. Both subthreshold and gate leakage power are heavily dependent on the input state. In FPGAs, the effect of input dependency is exacerbated due to the use of pass-transistor multiplexer logic, which can exhibit up to 50% variation in leakage power due to the input states. In this thesis, a new algorithm is proposed that uses bit permutation to reduce subthreshold and gate leakage power dissipation in FPGAs. The bit permutation algorithm provides an average leakage power reduction of 40% while having less than 2% impact on the performance and no penalty on the design area.

Thirdly, an accurate probabilistic power model for FPGAs is developed to quantify the savings from the proposed leakage power reduction techniques. The proposed power model accounts for dynamic, short circuit, and leakage power (including both subthreshold and gate leakage power) dissipation in FPGAs. Moreover, the power model accounts for power due to glitches, which accounts for almost 20% of the dynamic power dissipation in FPGAs. The use of probabilities in the power model makes it more computationally efficient than the other FPGA power models in the literature that rely on long input sequence simulations. One of the main advantages of the proposed power model is the incorporation of spatial correlation while estimating the signal probability. Other probabilistic FPGA power models assume spatial independence among the design signals, thus overestimating the power calculations. In the proposed model, a probabilistic model is proposed for spatial correlations among the design signals. Moreover, a different variation is proposed that manages to capture most of the spatial correlations with minimum impact on runtime. Furthermore, the proposed power model accounts for the input dependency of subthreshold and gate leakage power dissipation. By comparing the proposed power model to HSpice simulation, the estimated power is within 8% and is closer to HSpice simulations than other probabilistic FPGA power models by an average of 20%.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Prof. Mohamed Elmasry and Dr. Mohab Anis for their guidance and expertise. Without their knowledge and support, this work would not have been possible. I would also like to thank Prof. Farid Najm, Prof. Anthony Vannelli, Prof. Catherine Gebotys, and Prof. John Vanderkooy for reviewing this work as well as their valuable comments to make the thesis come out in this current form.

I am also grateful to Prof. Andrew Kennings who first introduced me to FPGA CAD design in the ECE647, which was one of the most interesting courses I attended during my study at Waterloo. I am thankful to Prof. James Barby, whose guidelines in teaching and course management will always be in mind if I ever get the chance to teach a course in the future.

I have been fortunate to work with many wonderful people in the VLSI research group in the University of Waterloo. I am thankful to all of them for their time and ideas that made this work more productive and my time spent at Waterloo memorable. I always remember the funny as well as informative discussions I used to have with some of my friends in the lab, including Ayman Hassan, Mohamed Abu Rahma, and Mohamed Elsaid. During my PhD studies in the University of Waterloo, I interacted with a wide group of interesting people that will always be my friends, including, Mohamed El-Abd, Shady Hassan, and Hatem Elbehiry.

I would like to thank Dr. Nizar Abdallah and Dr. Julien Dunoyer of Actel Corp. for their valuable discussion and support during my visit to Actel headquarters in 2007.

Moreover, I would like to express my deepest gratitude to my parents for going through so much in supporting my academic choices, even though it meant being away from them. Finally, I am deeply thankful to my wife who joined and supported me in the final and rough stages of this journey. I really appreciate her support during my defense exam.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Fueled by the increase in functionality and size of modern *Field Programmable Gate Arrays* (FPGAs) , the market for FPGAs has witnessed a notable expansion in the past few years. This increase in demand has pushed FPGA vendors to design modern FPGAs using state-of-the-art CMOS technologies. Consequently, modern FPGAs suffer greatly from the deep submicron issues that affected the ASIC industry earlier along the technology scaling road. The biggest deep submicron challenge that hurdles further expansions of the use of FPGAs in the VLSI industry is leakage power dissipation. In order for FPGAs to continue competing with the ASIC industry, FPGA vendors must tackle this issue using architectural and/or CAD techniques.

The continuous scaling of the CMOS process has attracted FPGA vendors to integrate more and more devices on the same chip to increase the chip functionality. As a result, the power dissipation of modern FPGAs increased significantly. Much of this increase in power dissipation is attributed to the increase in leakage power dissipation which is expected to exceed 50% of the FPGA power dissipation as modern FPGAs start using the 65nm CMOS process [1]. In addition, the excessive scaling of the MOS gate oxide thickness t_{ox} resulted in a significant increase in the gate oxide tunneling current, thus exacerbating the leakage problem. In recent experiments, it was found that both the subthreshold and gate leakage power dissipation increase by about 5X and 30X, respectively, across successive technology generations [2].

Traditionally, FPGA vendors were concerned with performance and area optimization to reduce the gap between FPGAs and ASIC designs. In the past few years, a shift in the industry towards power optimization has been witnessed as new power efficient FPGAs are introduced to the market. However, these power efficient FPGAs are mainly concerned with dynamic power reduction. With the expected dominance of leakage power dissipation, FPGA vendors are starting to move towards developing low-leakage FPGA devices.

1.2 Thesis Contributions

This thesis will provide several CAD and architectural modifications to FPGA designs to reduce the impact of leakage power dissipation on modern FPGAs.

Leakage reduction techniques have been applied in ASIC designs for the past few years. One of the most successful techniques is the use of Multi-threshold CMOS (MTCMOS). MTCMOS employs a high threshold voltage V_{th} (HVT) device, called sleep transistor (ST), in a circuit that is made of low-V_{th} devices (LVT). Hence, the design benefits from the high performance of the LVT devices and the reduced leakage power dissipation of the HVT device. In this thesis, MTCMOS techniques will be employed in FPGAs for leakage power reduction. Moreover, this architecture will enable shutting down the unused or idle parts of the FPGA, thus reducing their standby leakage significantly. Employing this technique in FPGAs requires several changes to the FPGA architecture, including the placement of the sleep transistors; location and connections to the sleep transistors, as well as the CAD design flow. The packing and placement stages of the CAD flow will be changed to reflect the addition of the sleep transistor and allow the possibility of turning OFF the idle parts of the FPGA. Furthermore, a new step is added in the CAD flow to identify the logic blocks that share common idleness periods so they can be collectively turned OFF during their idle periods.

Leakage power is characterized by being significantly state dependent. Depending on the input vector, the leakage power dissipation can vary by about 50%. In this thesis, a new methodology for leakage power optimization in FPGAs is proposed that depends on the state dependency of leakage power. By varying the order of the inputs to the logic blocks, the logic block leakage power is noticeably reduced. Moreover, the algorithm changes the deign configuration to put the FPGA components in the lowest leakage power mode.

In order to measure the leakage power savings from the proposed power optimization techniques, a newly proposed accurate power model for FPGAs is explained in this thesis. The power model takes into consideration the spatial correlation among the design signals. In addition, the power model developed is very flexible in terms of being able to estimate power dissipation in different FPGA architectures.

1.3 Thesis Organization

This thesis is organized as follows: an overview of the architecture; logic and routing architecture, and CAD flow of FPGAs is presented in Chapter 2. Chapter 3 discusses the current status of leakage power dissipation in modern FPGAs as well as some of the methods proposed in the literature to address power dissipation in FPGAs. Chapter 4 presents the newly proposed accurate power modeling technique for FPGAs. The proposed MTCMOS implementation of FPGAs is discussed in Chapter 5. The input reordering leakage power minimization methodology is proposed in Chapter 6. Finally, this thesis is concluded in Chapter 7 and possible future work is discussed.

Chapter 2

FPGA Overview: Architecture and CAD

2.1 Introduction

Fueled by the increase in the time-to-market pressures, the rise in ASIC mask and development costs, and increase in the Field Programmable Gate Arrays (FPGAs) performance and system-level features, more and more traditionally ASIC designers are migrating their designs to programmable logic devices. Moreover, programmable logic devices progressed both in terms of resources and performance. The latest FPGAs have come to provide platform solutions that are easily customizable for system connectivity, *Digital Signal Processing* (DSP), and/or data processing applications. These platform building tools accelerate the time-to-market by automating the system definition and integration phases of the *System on Programmable Chip* (SoPC) development.

The market of programmable logic devices is dominated by two main products; FPGAs and Complex Programmable Logic Devices (CPLDs). FPGAs mostly employ a look-up table approach to implement logic functions, while CPLDs use sum-of-products for logic implementation. Recently, FPGA vendors provided a comprehensive alternative to FPGAs for large volume demands called *structured ASICs* [3,4]. Structured ASICs offer a complete solution from prototype to high-volume production, and maintain the powerful features and high-performance architecture of their equivalent FPGAs with the programmability removed. Structured ASIC solutions not only provide performance improvement, but also result in significant high-volume cost reduction over FPGAs.

FPGAs consist of programmable logic resources embedded in a sea of programmable interconnects. The programmable logic resources can be configured to implement any logic function, while the interconnects provide the flexibility to connect any signal in the design to any logic resource. The programming technology for the logic and interconnect resources can be Static Random Access Memory (SRAM), flash memory [5], or antifuse [6, 7]. SRAM-based FPGAs offer in-circuit reconfigurability at the expense of being volatile, while antifuse are write-once devices. Flash-based FPGAs provide an intermediate alternative by providing reconfigurability as well as non-volatility. The most popular programming technology in state-of-the-art FPGAs is SRAM.

Traditionally, FPGAs consist of input/output pads, logic resources, and routing resources. However, state-of-the-art FPGAs usually include embedded memory, DSP blocks, *Phase-Locked Loops* (PLLs), embedded processors, and other special feature blocks, as shown in Figure 2.1. These features allowed FPGAs to be an attractive alternative for some SoPC designs. The next Sections shed light on some of the available commercial FPGA architectures and FPGA CAD flow.

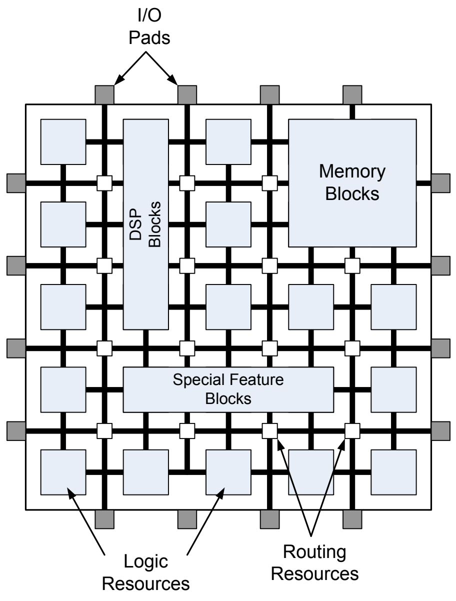


Figure 2.1: Modern FPGA fabric.

2.2 FPGA Logic Resources Architecture

The logic blocks in FPGAs are responsible for implementing the functionality needed by each application. Increasing the functional capability of the logic blocks increases the number of logic functions that can be packed into it. Moreover, increasing the size of logic blocks, *i.e.*, increasing the number of inputs to each logic block, increases the number of logic functions performed by each logic block as well as improving the area/delay performance of the logic block [8]. However, this comes on the expense of wasted resources because not all of the blocks will have all of their inputs fully utilized.

Most commercial FPGAs employ look-up tables (LUTs) to implement the logic blocks. A k-input LUT consists of 2^k configuration bits in which the required truth table is programmed during the configuration stage. The almost standard number of inputs for LUTs is four, which was proven optimum for area and delay objectives [9]. However, this number can vary depending on the targeted application by the vendor. Moreover, modern FPGAs utilize a hierarchial architecture where every group of basic logic blocks are grouped together into a bigger logic structure, logic cluster. The remaining of this Section describes the programmable logic resources in three of the most popular commercial FPGAs.

2.2.1 Altera Stratix III Logic Resources

The logic blocks in Altera's Stratix III are called *Adaptive Logic Modules* (ALMs). An 8-input ALM contains a variety of LUT-based resources that can be divided between two adaptive LUTs [10]. Being adaptive, ALMs can perform the conventional 4-input LUT operations as well as implementing any function of up to 6-inputs and some 7-input functions. Besides the adaptive LUTs, ALMs contain two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. Using these components, ALMs can efficiently perform arithmetic and shift operations. A detailed view of an ALM is shown in Figure 2.2. Every eight ALMs are grouped together to form a *Logic Array Block* (LAB).

2.2.2 Xilinx Virtex-5 Logic Resources

The slice is the basic logic resource in Xilinx Virtex-5 FPGAs. Slices consist of four LUTs, wide function multiplexers, and carry logic [11]. Figure 2.3 shows the architecture of a typical Virtex-5 slice. The slices employ four 6-LUTs that are capable of performing any 6-input logic function. Functions with up to 8-inputs can be implemented using multiplexers to combine the output of two LUTs. Every two interconnected slices are grouped together in a *Configurable Logic Block* (CLB) [11].

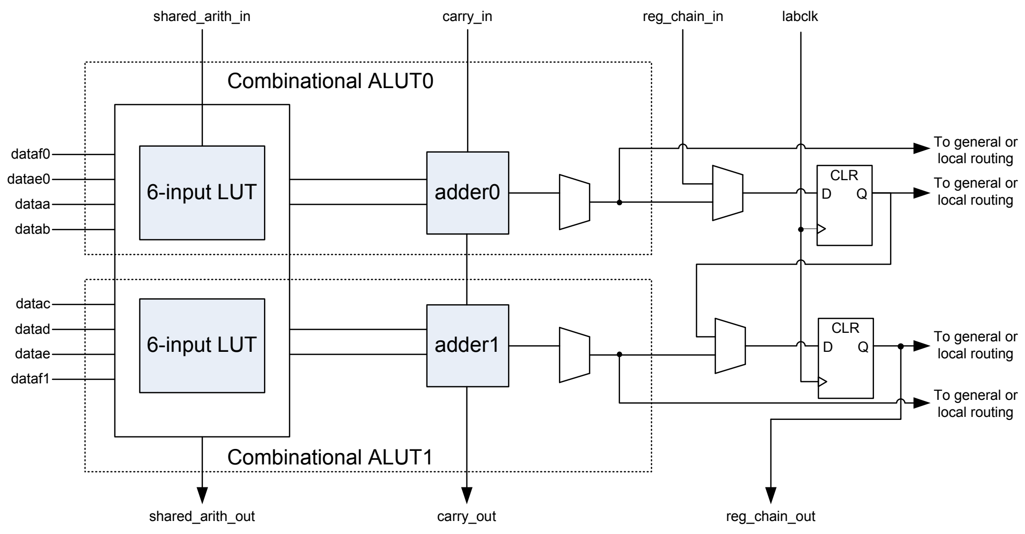


Figure 2.2: Altera's Stratix III ALM architecture [10].

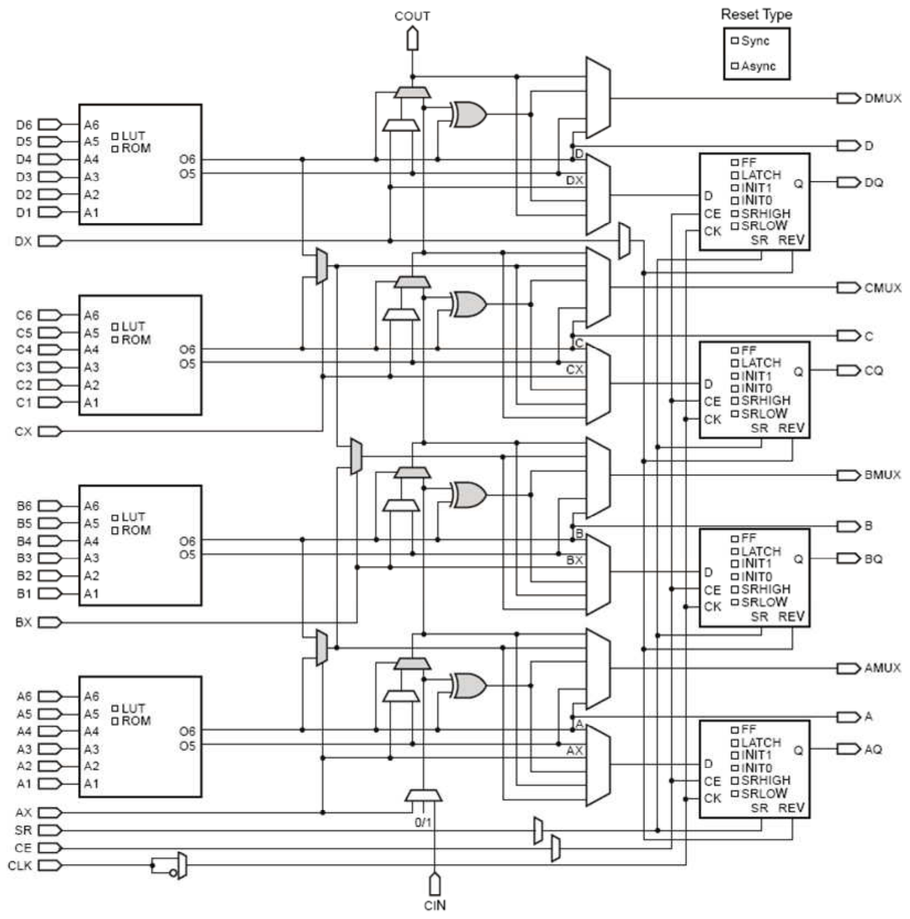


Figure 2.3: Xilinx's Vertex-5 slice architecture [11].

2.2.3 Actel ProASIC3/IGLOO Logic Resources

Actel ProASIC3/IGLOO FPGAs employ a flash-based architecture, instead of the conventional SRAM-based FPGAs used by both Altera and Xilinx, to store the configuration bits. The flash architecture provides the FPGAs with both reconfigurability and non-volatility. The ProASIC3/IGLOO FPGAs employ the VersaTile 3-input logic block that can implement any 3-input logic function as well as sequential functionality, as shown in Figure 2.4 [12]. Furthermore, the hierarchal architecture is not employed in the ProASIC3/IGLOO FPGAs and the output of each VersaTile can be directly routed to either the fast local lines or long routing resources. Another interesting characteristic of the VersaTile is that it does not adopt the conventional LUT architecture in FPGAs, as shown in Figure 2.4.

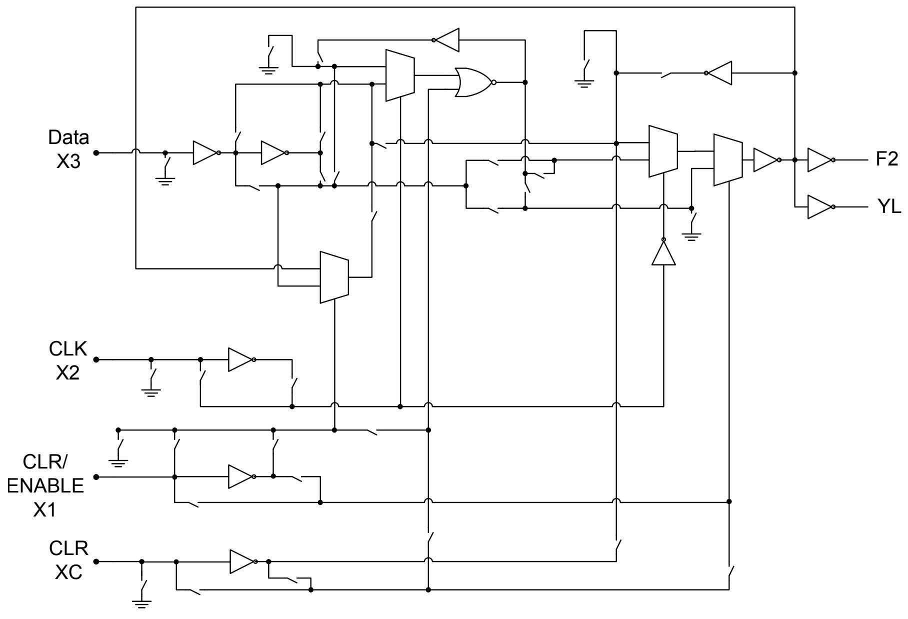


Figure 2.4: Actel's ProASIC3/IGLOO VersaTile architecture [12].

2.3 FPGA Routing Resources Architecture

Routing resources in FPGAs can be divided into two components; *segmented local routing* and *dedicated routing*. Segmented local routing is used to provide connection among the logic blocks. As depicted in Figure 2.5, the segmented wires are prefabricated in channels to provide programmable connections between switch blocks, connection blocks, and logic blocks. The number of wires in one channel is usually denoted by W [13].

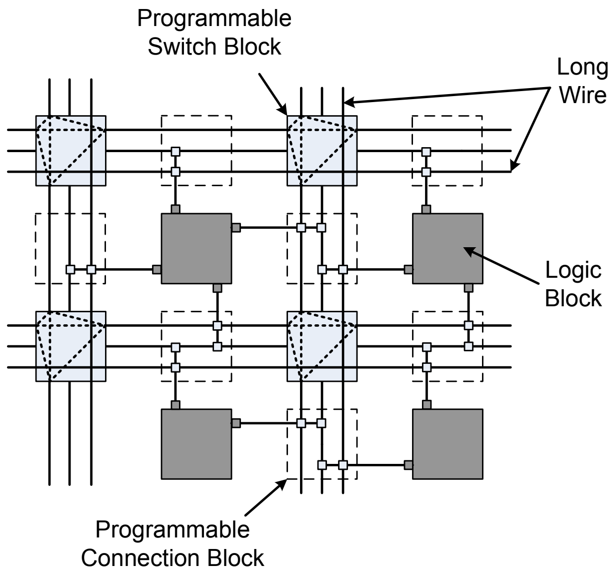


Figure 2.5: Routing resources in island-style FPGAs.

The I/O of the logic blocks are dynamically connected to the segmented routing channels on all four sides using *connection blocks*. The number of wires in each channel to which a logic block pin can connect to is called the *connection block flexibility* F_c. In addition, the *switch blocks* provide programmable connectivity between the horizontal and vertical wires. The *switch block flexibility* F_s is defined as the number of wires to which each incoming wire can connect to in a switch block. The *segment length* of a certain wire segment is defined as the number of logic blocks spanned by the routing wire. Modern FPGAs use a combination of wires of different segment lengths to achieve the optimum performance in terms of routability, delay, or both.

Dedicated routing is used for global signals that fan out to a large number of logic blocks, *e.g.*, clock and reset, thus providing low-skew. Moreover, some commercial FPGAs employ PLLs and *Delay-Locked Loops* (DLLs) for further skew reduction. Modern FPGAs have the flexibility to provide different clock domains inside the FPGA to enable asynchronous designs.

2.4 CAD for FPGAs

FPGAs are implemented using a huge number of programmable switches that are used to implement a certain logic function. The CAD tools of FPGAs transform the design, entered either as a schematic or using a hardware description language, to a stream of ‘1’s and ‘0’s that program the FPGA during the configuration time. The flow chart in Figure 2.6 shows the different steps involved in the CAD flow for a typical FPGA design.

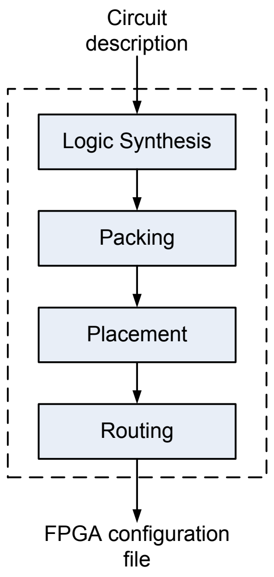


Figure 2.6: A typical FPGA CAD flow.

2.4.1 Logic Synthesis

In the *synthesis* phase, the circuit description is converted to a netlist of basic logic gates. This phase is usually divided into two different stages; *logic optimization* and *technology mapping* [14–17].

Logic optimization is a technology-independent stage that involves simplifying the logic function of the design without the use of any technology information. Any redundant logic is removed at this stage. The optimized user circuit is then mapped into LUTs and flip-flops in the technology mapping stage, where each k-bounded logic function in the circuit is mapped into a k-LUT. This step resolves to finding a set of k-feasible cuts that include all the nodes in the circuit in such a way to minimize the delay, area, and/or power dissipation of the final implementation. The process of technology mapping is often treated as a covering problem.

2.4.2 Packing

The *packing* phase converts the netlist of LUTs and flip-flops into a netlist of logic blocks, as shown in Figure 2.7. The input netlist is converted into clusters of logic blocks that can be mapped into the physical logic blocks of the FPGA. Most packing algorithms minimize the number of resulting logic blocks, the number of connections between them, and/or the delay along the critical path. The packing algorithm has to consider the physical limitations of the actual logic blocks of the FPGA in terms of the maximum number of LUTs in a logic block and the number of distinct input signals and clocks a logic block has.

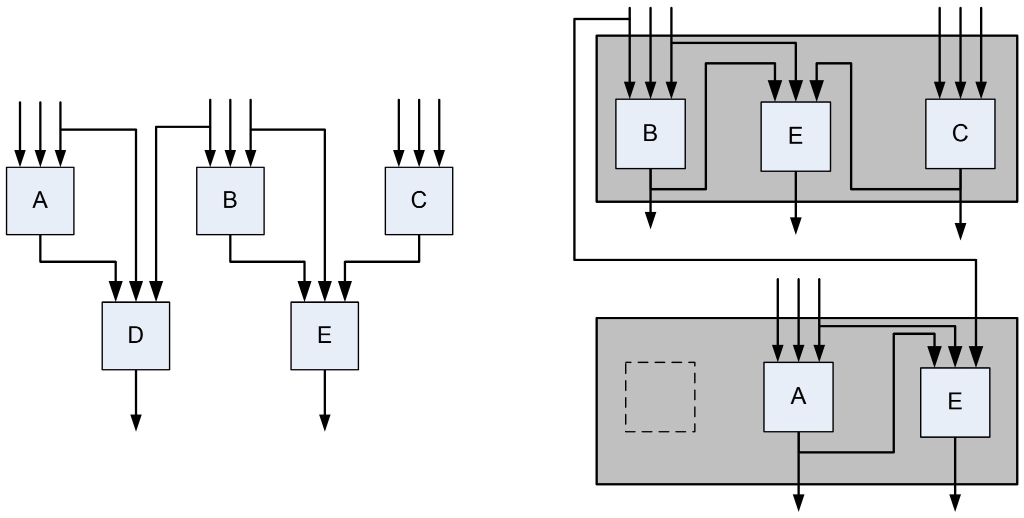


Figure 2.7: An example of packing.

Packing algorithms can be categorized as either bottom-up [14, 18–20] or top-down [21, 22]. Bottom-up packing algorithms build each cluster individually around a seed LUT until the cluster is full. However, top-down packing approaches partition the LUTs into clusters by successive circuit subdivision. Bottom-up algorithms are much faster and simpler than top-down approaches because they only consider local connections. However, this comes at the expense of solution quality.

2.4.3 Placement

In the placement phase, the packed logic blocks are distributed among the physical logic blocks in the FPGA fabric. Placement algorithms try to minimize the delay along the critical path and enhance the resulting circuit routability. Available placement algorithms can be classified into three categories; min-cut [23, 24], analytic [25, 26], and simulated annealing [27–29] based algorithms. Most of the commercial placement tools for FPGAs employ simulated annealing-based algorithms because of their flexibility to adapt to a wide variety of optimization goals.

Simulated annealing (SA) placement tools depend on the SA algorithm, which is derived from the annealing process used to cool molten metals [30]. Initially, a random initial placement for all the logic blocks is generated. Afterwards, pairs of logic blocks are selected at random as candidates for swapping to improve the cost function. If the swap results in a decrease in the cost function, it is directly allowed, otherwise, it is only allowed with a probability that decreases as the algorithm progresses, thus allowing less worsening swaps after every iteration. A pseudocode for the SA placer is listed in Algorithm 2.1.

Algorithm 2.1 SA generic placer pseudocode.

```
S = RandomPlacement()
T = InitialTemperature()
while ExitCriterion() == False do
  /* Outer loop */
  while InnerLoopCriterion () == False do
    /* Inner loop */
    $S_{new}$ = GenerateViaMove(S)
    $\Delta C$ = Cost($S_{new}$) - Cost(S)
    r = random(0,1)
    if $r < e^{-\Delta C/T}$ then
      S = $S_{new}$
    end if
  end while
  T = UpdateTemp()
end while
```

2.4.4 Timing Analysis

Timing analysis [31] is used to guide placement and routing CAD tools in FPGAs to: (1) determine the speed of the placed and routed circuit and (2) estimate the slack of each source-sink connection during routing to identify the critical paths. Timing analysis is usually performed on a directed graph representing the circuit, where the nodes represent LUTs or registers and the edges represent connections.

The minimum required clock period of the circuit can be determined by a breadth first search through the graph, starting from the primary inputs, to find the arrival time at node i using the following relation

$$T_{arrival}(i) = \max_{\forall j \in fanin(i)} \{T_{arrival}(j) + delay(j,i)\} \ , \tag{2.1}$$

where $delay(j,i)$ is the delay on the edge between j and i. The required time at node i is calculated by a breadth-first search of the graph, starting from the primary outputs, and using the following relation

$$T_{required}(i) = \min_{\forall j \in fanout(i)} \{T_{required}(j) + delay(i,j)\} \; . \tag{2.2}$$

Afterwards, the slack on the connection between node i and j is calculated as

$$slack(i,j) = T_{required}(j) - T_{arrival}(i) - delay(i,j) \; . \tag{2.3}$$

Connections with a zero *slack* are critical connections, while those with a positive *slack* are non-critical ones that can be routed using longer routes.

2.4.5 Routing

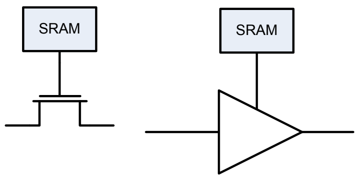
The routing phase assigns the available routing resources in the FPGA to the different connections between the logic blocks in the placed design [28]. The objective of a typical routing algorithm is to minimize the delay along the critical path and avoid congestions in the FPGA routing resources. Generally, routing algorithms can be classified into *global routers* and *detailed routers*. Global routers consider only the circuit architecture without paying attention to the number and type of wires available, while detailed routers assign the connections to specific wires in the FPGA.

2.5 Versatile Place and Route (VPR) CAD Tool

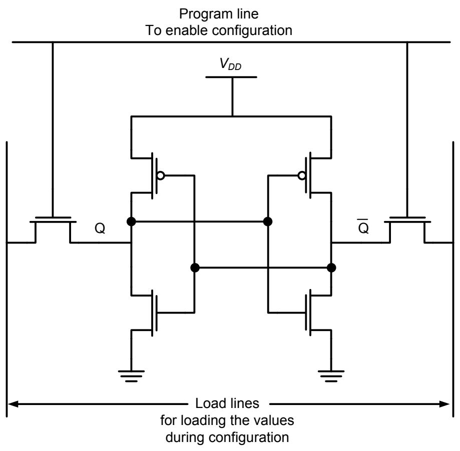
The CAD flow used in this thesis is based on the *Versatile Place and Route* (VPR) CAD tool. VPR is a popular academic placement and routing tool for FPGAs [28]. Moreover, VPR is the core for Altera's Modeling Toolkit (FMT) CAD tool [32,33]. VPR is usually used in conjunction with T-VPack [18, 27], a timing-driven logic block packing algorithm. VPR consists of two main parts; a placer and router and an area and delay model. These two components interact together to find out the optimum placement and routing that satisfies a set of conditions. This Section describes the FPGA architecture supported by VPR as well as giving a quick overview about the tool flow.

2.5.1 VPR Architectural Assumptions

VPR assumes an SRAM-based architecture, where SRAM cells hold the configuration bits for all the pass-transistors multiplexers and tri-state buffers in the FPGA in both logic and routing resources, as shown in Figure 2.8(a). The SRAMs used are the six-transistor SRAM cell made of minimum size transistors, as shown in Figure 2.8(b). Moreover, an island-style FPGA is assumed by VPR, where the logic clusters are surrounded by routing tracks from all sides. VPR uses an *architecture file* to describe the underlying FPGA architecture used. The architecture file contains information about the logic block size, wire segment length, connection topologies, and other information used by VPR. The use of the architecture file allows VPR to work on a wide range of FPGA architectures. However, there are some general architectural assumptions made by VPR which are discussed in this Section.



(a) Programmable switches and tri-state buffers in VPR FPGA architecture.



(b) Six-transistor SRAM circuit.

Figure 2.8: Building blocks of SRAM programmable FPGAs used by VPR.

2.5.1.1 VPR Logic Architecture

VPR targets the hierarchal or cluster-based logic architecture, where every N of the smallest logic element, called *Basic Logic Element* (BLE), are grouped together to a form a *Cluster of Logic Blocks* . Each BLE consists of a k-LUT, a D flip-flop (DFF), and a 2:1 multiplexer (MUX), as shown in Figure 2.9. Such configuration allows both the registered and unregistered versions of the output to be readily available. Local routing resources are used to connect the BLEs inside each logic cluster to each other and to the inputs/outputs of the logic cluster, as shown in Figure 2.10. As noticed in Figure 2.10, not all of the BLE inputs are accessible from outside. However, any of the BLEs inputs can be connected to any of the BLEs outputs or any of the external inputs.

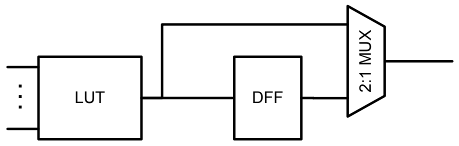


Figure 2.9: VPR BLE architecture.

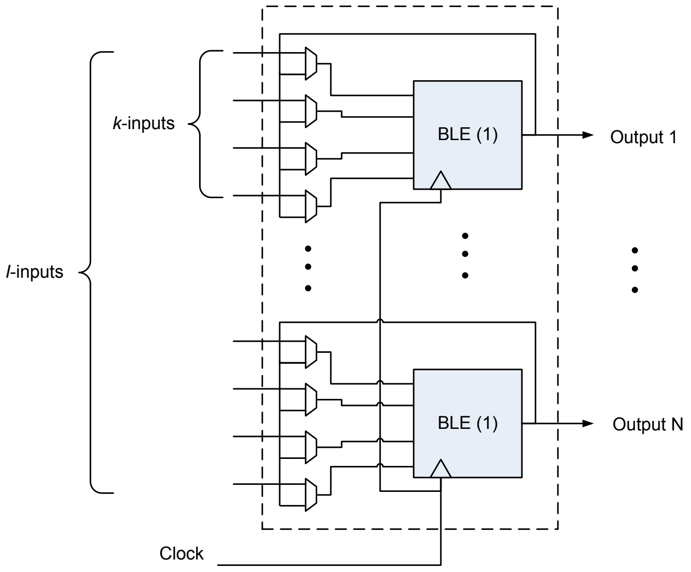


Figure 2.10: VPR logic cluster architecture.

A logic cluster is defined in the architecture file by four main parameters: (1) the size of its LUTs k, (2) the number of BLEs in the cluster N, (3) the number of external inputs to the cluster I, and (4) the number of external clock inputs M_{clk}. It should be noted that VPR assumes minimum-sized transistors are used to implement the LUTs, as a result, the capacitances of these transistors are ignored. However, VPR accounts for the capacitance of the internal routing tracks within the logic cluster.

2.5.1.2 VPR Routing Resources Architecture

VPR divides the routing resources characterization into three categories: channel, switch block, and wire parameters. The channel information specifies the channel width and the connections between between the IO pads and logic blocks from one side and the routing tracks from the other side. The channel width parameters include the width of horizontal ($chan\_width\_x$) and vertical ($chan\_width\_y$) routing channels, the width of the IO channel ($chan\_width\_io$), and the number of IO pads that fit in one row or column of logic clusters (io_rat). The connections between the routing tracks and either the logic blocks or IO pads are defined by the number of tracks connected to each logic block input ($F_{c,input}$) and output ($F_{c,output}$) and the number of tracks connected to each IO pad ($F_{c,pad}$). As an example, Figure 2.11 shows a high-level view of a sample VPR FPGA routing model and the values of the parameter used to describe the channel.

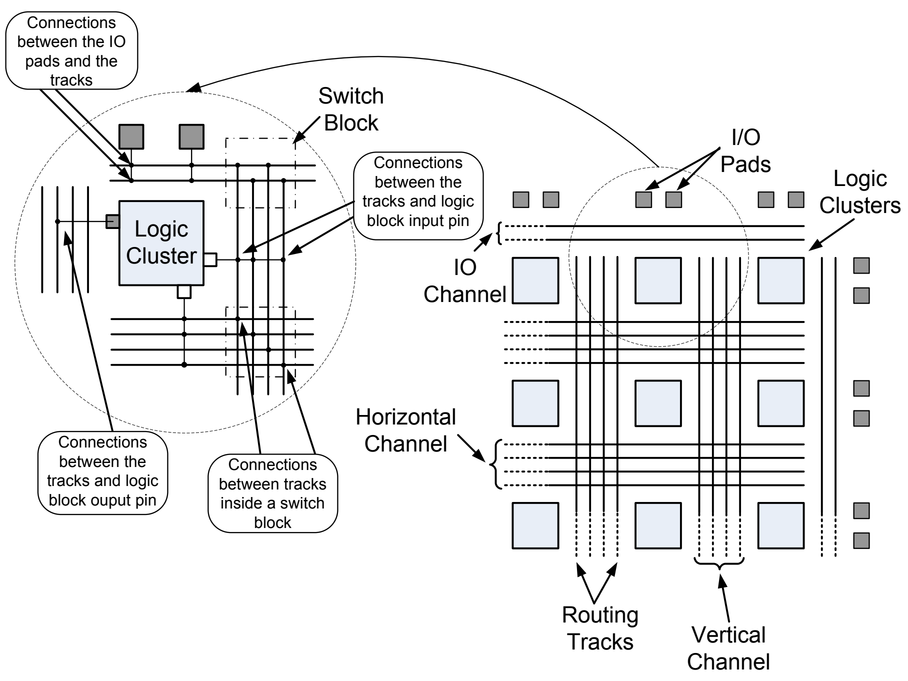
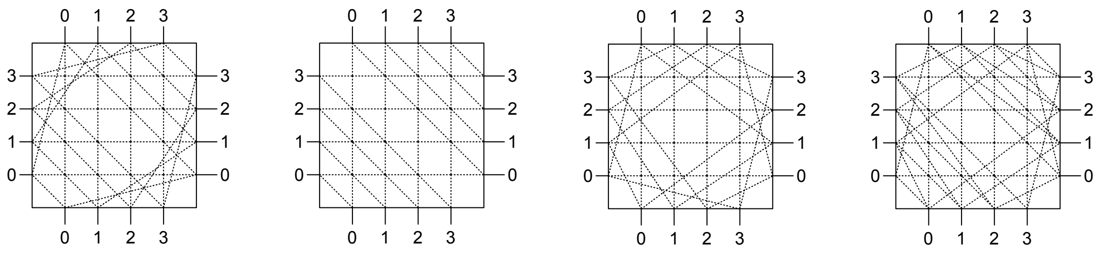

Connections between the IO pads and the tracks
Switch Block
Connections between the tracks and logic block input pin
Logic Cluster
I/O Pads
Logic Clusters
IO Channel
Horizontal Channel
Connections between the tracks and logic block ouput pin
Connections between tracks inside a switch block
Routing Tracks
Vertical Channel

Figure 2.11: VPR FPGA routing architecture. $chan\_width\_x = 1$, $chan\_width\_y = 1$, $chan\_width\_io = 0.5$, $io\_rat = 2$, $F_{c,input} = 3$, $F_{c,output} = 1$, and $F_{c,pad} = 2$.

Switch blocks are used to provide programmable connectivity between the horizontal and vertical routing tracks, as shown in Figure 2.11. VPR characterizes switch blocks by their resistance (R), input capacitance (C_{in}), output capacitance (C_{out}), intrinsic delay (T_{del}), connection flexibility (F_s), switch type (whether buffered or not), and the switch block topology. The connection flexibility of a switch is defined as the number of connections available for each pin to other pins on the other sides of the switch. Figure 2.8(a) shows the unbuffered and buffered versions of the switch blocks supported by VPR. Four different topologies of switches can be used within VPR: Disjoint [34], Universal [35], Wilton [36], and Imran [37], as shown in Figure 2.12.



(a) Disjoint switch block. (b) Universal switch block. (c) Wilton switch block. (d) Imran switch block.

Figure 2.12: Switch topologies supported by VPR.

Finally, VPR describes wire segments by: the usage frequency of the segment in the FPGA (*segment_frequency*), the number of logic clusters spanned by the wire (*segment_length*), the resistance (R_{metal}) and capacitance (C_{metal}) per unit length, the switch type that connects the wire and logic clusters (*opin_switch*), the switch type that connects the wire with other wires (*wire_switch*).

2.5.2 Basic Logic Packing Algorithm: VPack

VPack is a logic packing algorithm that converts an input netlist of LUTs and registers into a netlist of logic clusters. The packing is done in a hierarchial manner in two stages: packing LUTs and registers into BLEs and packing a group of N, or less, BLEs into logic clusters. The pseudocode for VPack is listed in Algorithm 2.2.

The first stage of VPack is a pattern matching algorithm that packs a register and a LUT into one BLE when the output of the LUT fans out to *only* one register, as shown in Figure 2.13. The second phase packs the BLEs into logic blocks to achieve two objectives: (1) fill the logic clusters to their full capacity N and (2) minimize the number of inputs to each cluster. These two objectives originate from the two main goals of packing: area reduction and improving routability. Packing starts by putting BLEs into the current cluster sequentially in a greedy manner while satisfying the following hard constrains:

1. the number of BLEs must be less than or equal to the cluster size N,

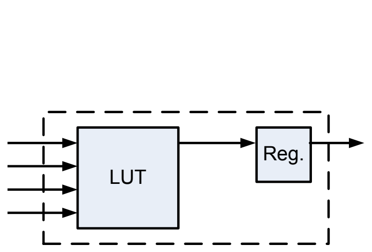
2. the number of externally generated signals, and used inside the cluster, must be less than or equal to the number of inputs to the cluster I,

Algorithm 2.2 VPack pseudocode [28].

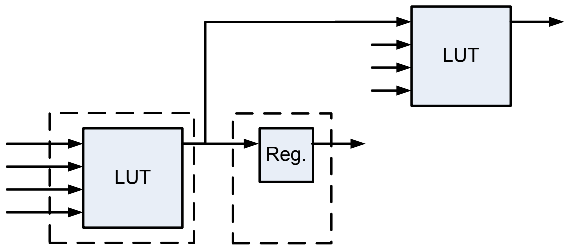
```
Let: **UnclusteredBLEs** be the set of BLEs not contained in any cluster
- **C** be the set of BLEs in the current cluster
- **LogicClusters** be the set of clusters (where each cluster is a set of BLEs)
UnclusteredBLEs = PatternMatchToBLEs (LUTs, Registers)
LogicClusters = NULL
while UnclusteredBLEs != NULL do
   /* More BLEs to cluster */
   C = GetBLEwithMostUsedInputs (UnclusteredBLEs)
   while |C| < $N$ do
      /* Cluster is not full */
      BestBLE = MaxAttractionLegalBLE (C, UnclusteredBLEs)
      if BestBLE == NULL then
         /* No BLE can be added to cluster */
         break
      end if
      UnclusteredBLEs = UnclusteredBLEs - BestBLE
      C = C $\cup$ BestBLE
   end while
   if |C| < $N$ then
      /* Cluster not full — try hill-climbing */
      while |C| < $N$ do
         BestBLE = MINClusterInputIncreaseBLE (C, UnclusteredBLEs)
         C = C $\cup$ BestBLE
         UnclusteredBLEs = UnclusteredBLEs - BestBLE
      end while
      if ClusterIsIllegal (C) then
         RestoreToLastLegalState (C, UnclusteredBLEs)
      end if
   end if
   LogicClusters = LogicClusters $\cup$ C
end while
```

3. the number of distinct clock signals needed by the cluster must be less than or equal to the number of clock inputs M_{clk}.

A seed BLE is selected for each cluster such that it has the maximum number of inputs among the unclustered BLEs. Other unclustered BLEs B are attracted to



(a) LUT and register packing into one BLE



(b) LUT and register packing into two BLEs

Figure 2.13: Packing LUTs and registers into BLEs [28].

the cluster C in such a way as to maximize the *attraction*() objective function

$$Attraction(B) = |Nets(B) \cap NetsC| \, , \tag{2.4}$$

where $Nets(x)$ are the nets connected to BLE (or cluster) x. This process continues until the cluster is filled to its maximum capacity N.

If the cluster does not reach its maximum capacity, but the number of inputs used by the BLEs inside it reaches I, a hill-climbing stage is invoked. In this stage, unclustered BLEs are added to the cluster in such a way to minimize the increase in the number of inputs to the cluster, an example of that is depicted in Figure 2.14. This is achieved by minimizing the following cost function

$$\Delta_{\text{cluster inputs}}(B) = |Fanin(B)| - |Nets(B) \cap Nets(C)| \, . \tag{2.5}$$

It is worth mentioning that the hill-climbing stage allows violating the number of inputs constraint while executing, but does not permit violating the clock inputs constraint. The hill-climbing phase terminates when the cluster size reaches N. If the cluster is infeasible, *i.e.*, its inputs are more than I, the algorithm retracts to the last feasible cluster. Afterwards, VPack selects a new seed BLE and constructs a new cluster.

2.5.3 Timing-Driven Logic Block Packing: T-VPack

T-Vpack [18, 27] is a modified version of the VPack algorithm that attempts to minimize the number of inter-cluster connections along the critical path, besides packing the clusters to their maximum capacity. This achieves speed up along the critical path as local interconnects (intra-cluster connections) are faster than inter-cluster interconnects. T-VPack employs a timing analyzer to calculate the *slack*

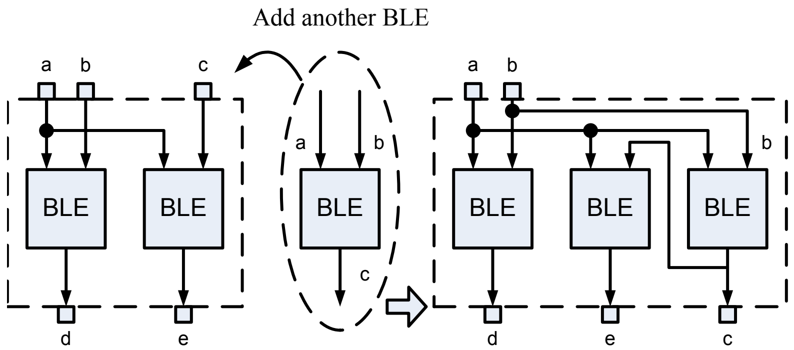


Figure 2.14: Adding a BLE to a cluster can decrease the number of used cluster inputs [28].

along the connections in the design and identify the critical path(s). The *criticality* measure of a connection is calculated as

$$ConnectionCriticality(i) = 1 - \frac{slack(i)}{MaxSlack}\ , \tag{2.6}$$

where $MaxSlack$ is the largest *slack* in the circuit.

In T-VPack, the BLE with the highest *criticality*, *i.e.*, the BLE connected to the nets with the highest *ConnectionCriticality*, is selected as the seed BLE for the cluster. Afterwards, BLEs are attracted to the cluster to maximize a modified version of the *Attraction*() function in Eq. (2.4), given by

$$Attraction(B) = \lambda \times Criticality(B) + (1 - \lambda) \times \frac{Nets(B) \cap Nets(C)}{MaxNets}\ , \tag{2.7}$$

where $MaxNets$ is and λ is $Criticality(B)$

2.5.4 Placement: VPR

VPR models that the FPGA as a block array of logic clusters bounded by routing tracks, as shown in Figure 2.15. Simulated Annealing (SA) is used as the optimization algorithm for placement in VPR using an adaptive annealing schedule to adapt to the current placement at any time instant.

The initial temperature is selected from the basic features of the circuit. Assume that the total number of logic clusters in the design is N_{clusters}. After the initial random placement is evaluated, N_{clusters} pairwise swaps are performed and the initial temperature is calculated as 20 times the standard deviation of the cost of the different N_{clusters} combinations evaluated. Moreover, the number of inner moves

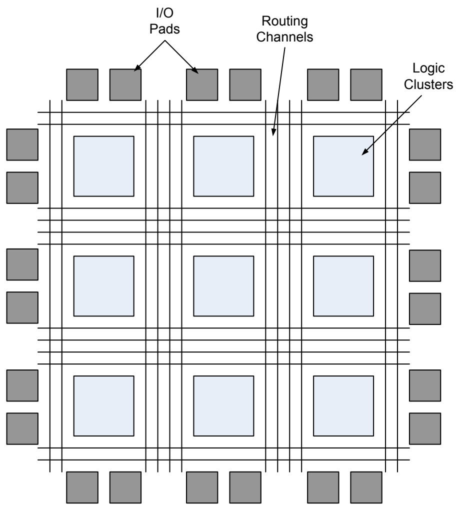


Figure 2.15: FPGA model assumed by the VPR placer.

performed at each temperature is evaluated as

$$MovesPerTemperature = InnerNum \times N_{\text{clusters}}^{4/3} \ , \tag{2.8}$$

where $InnerNum$ is a constant and usually set to 10.

Another feature of the adaptive SA algorithm used in VPR is the way the temperature is updated. In conventional SA, almost all of the moves are accepted at high temperatures, while at low temperatures, only improving moves are accepted. In the adaptive SA [28], the cooling scheme tries to prolong the time spent in these cost improving temperatures (medium and low temperatures) at the expense of possibly cost worsening temperatures (high temperatures) using the following temperature update relationship

$$T_{new} = \gamma \times T_{old} \ , \tag{2.9}$$

where γ is evaluated with respect to the percentage of moves accepted (α), according to Table 2.1.

Table 2.1: VPR temperature update schedule [28].

α	γ
$0.96 < \alpha$	0.5
$0.8 < \alpha \leq 0.96$	0.9
$0.15 < \alpha \leq 0.8$	0.95
$\alpha \leq 0.15$	0.8

2.5.5 Routing: VPR

VPR incorporates two different routing algorithms; a *routability-driven router* and a *timing-driven router*.

2.5.5.1 Routability-Driven Router

The VPR routability-driven router is based on the *Pathfinder* algorithm [38]. The Pathfinder algorithm repeatedly rips-up and re-routes every net in the circuit during each routing iteration until all congestions are removed. Initially, all nets are routed to minimize the delay, even if this results in congestion. Afterwards, routing iterations are applied to overused routing resources to resolve such congestions. In VPR, the cost of using a routing resource n when it is reached by connecting it to routing resource m is given by

$$Cost(n) = b(n) \times h(n) \times p(n) + BendCost(n, m) \ , \tag{2.10}$$

where $b(n)$, $h(n)$, and $p(n)$ are the base cost, historical congestion, and present congestion, respectively. $b(n)$ is set to the delay of n, $delay(n)$. $h(n)$ is incremented after each routing iteration in which n is overused. $p(n)$ is set to '1' if routing the current net through n will not result in congestion and increases with the amount of overuse of n. The $BendCost(n, m)$ is used to penalize bends in global routing to improve the detailed routability.

2.5.5.1.1 Timing-Driven Router

The timing-driven router in VPR is based on the Pathfinder, but timing information is considered during every routing iteration. Elmore delay models are used to calculate the delays, and hence, timing information in the circuit. In order to include timing information, the cost of including a node n in a net's routing is given by

$$Cost(n) = Crit(i, j) \times delay(n, topology) + [1 - Crit(i, j)] \times b(n) \times h(n) \times p(n) \ , \tag{2.11}$$

where a connection criticality $Crit(i, j)$ is given by

$$Crit(i, j) = \max\left\{ \left[MaxCrit - \frac{slack(i, j)}{D_{max}} \right]^{\eta}, 0 \right\} \ , \tag{2.12}$$

where D_{max} is the critical path delay and η and $MaxCrit$ are parameters that control how a connection's slack impacts the congestion-delay tradeoff in the cost function.

Chapter 3

Leakage Power in Modern FPGAs

The tremendous growth of the semiconductor industry in the past few decades is fueled by the aggressive scaling of the semiconductor technology following Moore's Law. As a result, the industry witnessed an exponential increase in the chip speed and functional density with a significant decrease in power dissipation and cost per function [39]. However, as the CMOS devices enter the nanometer regime, leakage current is becoming one of the main hurdling blocks to Moore's law. According to Moore himself, the key challenge for continuing process scaling in the nanometer era is leakage power reduction [40]. Thus, circuit designers and CAD engineers have to work hand in hand with device designers to deliver high-performance and low-power systems for future CMOS devices.

In this Chapter, the leakage power problem is discussed in the VLSI industry in general and in FPGAs in particular. Moreover, the proposed techniques in the literature to combat both dynamic and leakage power dissipation in FPGAs are presented and their advantages and disadvantages are discussed.

3.1 Dynamic Power Reduction Techniques in FPGAs

Most of the low-power design techniques proposed for FPGAs targeted dynamic power dissipation. These works were motivated by the fact that dynamic power dissipation constituted most of the power dissipation in old CMOS technologies, 180nm and before. In [41], a hierarchal interconnect architecture is proposed to reduce dynamic power dissipation in the interconnect multiplexers. The basic idea behind this work was to reduce the capacitance of the interconnects by reducing the number of wire segments connected to each switch box. Moreover, the authors proposed the use of lower supply voltage to reduce the total power dissipation.

A low swing signalling scheme was proposed in [42] for FPGAs for total power reduction. The authors used level converters at both sides of long interconnects that reduces the voltage of the transmitted signal. Moreover, the authors proposed a symmetric mesh architecture for the FPGA interconnects that reduces the total wire capacitance.

An efficient power-aware CAD flow was implemented in [43]. The authors introduced power figures of merits to the cost function at every level of the FPGA CAD design flow. The main idea of the work is to minimize the capacitance of signals with high switching activity. In the technology mapping phase, the authors include the switching activity of the feasible cuts evaluated during mapping. The clustering phase tries to cluster the logic blocks connected to a high activity net together to reduce the net capacitance. Similarly, during the placement stage, clusters connected to high activity nets are placed close to each other.

A power-aware technology mapping methodology for FPGAs was proposed in [44] that aims to keep nets with high switching activity out of the FPGA routing network and takes an activity-conscious approach to logic replication. As result, the wire capacitance of high activity nets is reduced significantly.

A methodology to reduce glitching power in FPGAs was proposed in [45] that relies on adding extra programmable delays to balance path delays. By adding extra delays to nets with positive slacks, the authors managed to equalize the delays at most of the inputs to each logic block in the design. As a result, glitching power is reduced to almost zero.

3.2 CMOS Technology Scaling Trends and Leakage Power in VLSI Circuits

The main driving forces that govern the CMOS technology scaling trend are the overall circuit requirements; the maximum power dissipation, the required chip speed, and the needed functional density. The overall device requirements such as the maximum MOSFET leakage current, minimum MOSFET drive current, and desired transistor size are determined to meet the overall circuit requirements. Similarly, the choices for MOSFET scaling and design, including the choice of physical gate length L_g and equivalent oxide thickness of the gate dielectric t_{ox}, etc., are made to meet the overall device requirements. Figure 3.1 depicts the scaling trend for the CMOS feature size across several technology generations as well as some future predictions according to the semiconductor roadmap published by the International Technology Roadmap for Semiconductors (ITRS) [1].

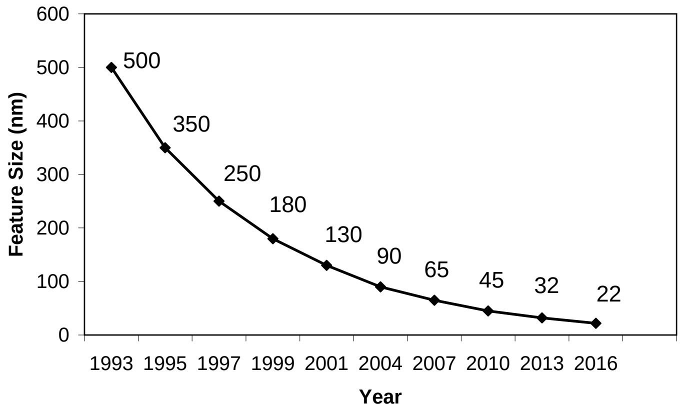


Figure 3.1: Gate length scaling of CMOS technologies [1].

There are two common types of scaling trends in the CMOS process: constant field scaling and constant voltage scaling. Constant field scaling yields the largest reduction in the power-delay product of a single transistor. However, it requires a reduction in the power supply voltage as the minimum feature size is decreased. Constant voltage scaling does not suffer from this problem, therefore, it provides voltage compatibility with older circuit technologies. The disadvantage of constant voltage scaling is that the electric field increases as the minimum feature length is reduced, resulting in velocity saturation, mobility degradation, increased leakage currents, and lower breakdown voltages. Hence, the constant field scaling is the most widely used scaling approach in the CMOS industry. Table 3.1 summarizes the constant field scaling in the CMOS process.

In order to maintain the switching speed improvement of the scaled CMOS devices, the threshold voltage V_{TH} of the devices is also scaled down to maintain a constant device overdrive. However, decreasing V_{TH} results in an exponential increase in the subthreshold leakage current,

$$I_D \propto 10^{\frac{V_{GS}-V_{TH}+\eta V_{DS}}{S}}, \tag{3.1}$$

where $S = \frac{nkT}{q} ln10$. Moreover, as the technology is scaled down, the oxide thickness t_{ox} is also scaled down, as shown in Table 3.1. The scaling down of t_{ox} results in an exponential increase in the gate oxide leakage current.

Table 3.1: Constant field scaling of the CMOS process.

Parameter	Symbol	Constant Field Scaling
Gate Length	L	$1/\alpha$
Gate Width	W	$1/\alpha$
Field	ε	1
Oxide Thickness	t_{ox}	$1/\alpha$
Substrate Doping	N_a	α
Gate Capacitance	C_G	$1/\alpha$
Oxide Capacitance	C_{ox}	α
Circuit Delay	t_d	$1/\alpha$
Power Dissipation	P_d	$1/\alpha^2$
Area	A	1
Power Density	P/A	1

As a result of the continuous scaling of V_{TH} and t_{ox}, the contribution of the total leakage power to the total chip power dissipation is increasing notably. The contribution of leakage power is expected to exceed 50% of the total chip power by the 65nm CMOS process [1], as shown in Figure 3.2.

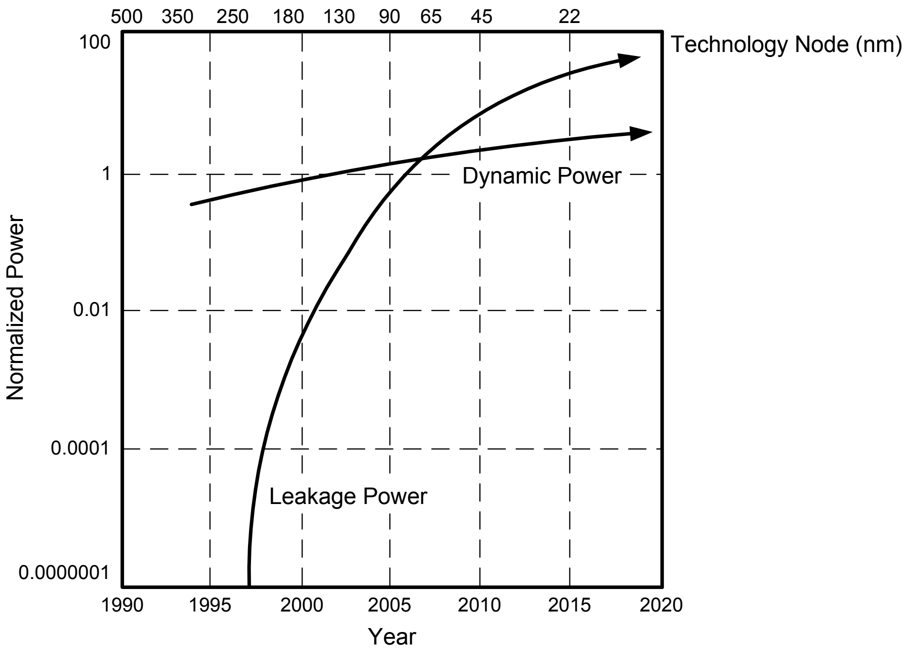


Figure 3.2: Leakage power contribution to the total chip power [1].

3.3 CMOS Devices Leakage Mechanisms

There are six short-channel leakage current mechanisms in CMOS devices. Figure 3.3 summarizes the leakage current types that affect state-of-the-art CMOS devices [46]. I_1 is the reverse-bias pn junction leakage; I_2 is the subthreshold leakage; I_3 is the oxide tunneling current; I_4 is the gate current due to hot-carrier injection; I_5 is the Gate Induced Drain Leakage (GIDL); and I_6 is the channel punchthrough current. Currents I_2, I_5, and I_6 are OFF-state leakage currents, while I_1 and I_3 occur in both ON and OFF states. I_4 can occur in the OFF state, but more typically occurs during the transistor transition [46]. The main sources for leakage power dissipation in current CMOS technologies are the subthreshold leakage and gate oxide leakage currents.

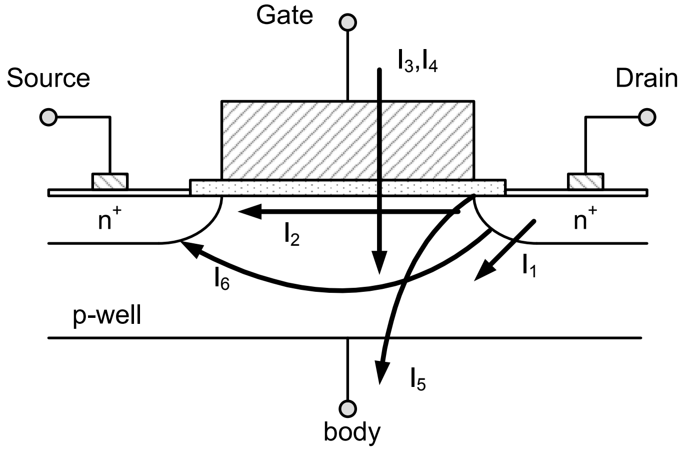


Figure 3.3: Leakage current mechanisms of deep submicron devices [46].

There are two main components for the reverse-bias pn junction leakage I_1; minority carrier diffusion/drift near the edge of the depletion region and electron-hole pair generation in the depletion region of the reverse-biased junction. I_2 flows between the source and drain in a MOSFET when the gate voltage is below V_{th}. I_3 occurs by electrons tunneling from the substrate to the gate and also from the gate to the substrate through the gate oxide layer. I_4 occurs due to electrons or holes gaining sufficient energy from the applied electric field to cross the interface potential barrier and enter into the oxide layer. I_5 is due to the high field effect in the drain junction of the MOSFET. Because of the proximity of the drain and the source, the depletion regions at the drain-substrate and source-substrate junctions extend into the channel. Channel length reduction and the increase in the reverse bias across the junctions push the junctions nearer to each other until they almost merge, thus leading to the punchthrough current I_6.

Out of these six different leakage current mechanisms experienced by current CMOS devices, subthreshold and gate leakage currents are the most dominant leakage currents. Furthermore, the contribution of subthreshold leakage current to the total leakage power is much higher than that of gate leakage current, especially at above room temperature operating conditions. The contribution of gate leakage current to the leakage power dissipation is expected to increase significantly with the technology scaling, unless high K materials are introduced in the CMOS fabrication industry [1].

3.4 Current Situation of Leakage Power in Nanometer FPGAs

For FPGAs to support reconfigurability, more transistors are used than those used in an ASIC design that performs the same functionality. Consequently, leakage power dissipation in FPGAs is higher than that in their ASIC counterpart. It was reported in [47] that on average, the leakage power dissipation in FPGA designs is almost 5.4 times that of their ASIC counterparts under worst-case operating conditions. The excess leakage power dissipated in FPGAs is mainly due to the programming logic that is not present in ASIC designs.

A study of the leakage power dissipation in a 90nm CMOS FPGA was performed in [48], of which the results are summarized in Table 3.2. By comparing the average leakage power dissipation of a typical 90nm CMOS FPGA at 25°C and 85°C, it can be seen that the average leakage power increases by four times. Moreover, the results in the first column are for a utilization of 75%, hence, the leakage power dissipation for a 1,000 CLB FPGA would be in the range of 4.2mW. If these FPGAs are to be used in a wireless mobile application, which has a typical leakage current of 300μA, then the maximum number of CLBs that can be used would be 86 CLBs for the 25°C and 20 CLBs for the 85°C.

Table 3.2: FPGA leakage power for typical designs and design-dependent variations [48].

T	Typical P_{LEAK} (avg. input data; $U_{CLB} = 75\%$)	Best-Case input data	Worst-Case input data
25°C	4.25μW/CLB	-12.8%	+13.0%
85°C	18.9μW/CLB	-31.1%	+26.8%

In addition, the dependance of leakage on the input data increases significantly with the temperature. This can be deduced from Table 3.2 as the variation due to the worst and best cases input vectors change from ±13% at 25°C to about ±28% at 85°C. Furthermore, in another experiment conducted in [48], it was found out that for a 50% CLB utilization, 56% of the leakage power was consumed in the unused part of the FPGA. Hence, in future FPGAs these unused parts have to be turned down to reduce this big portion of leakage power dissipation.

3.5 Leakage Power Reduction Techniques in FPGAs

With the increasing contribution of leakage power to the total power dissipation, leakage power started to take a front seat in FPGAs power reduction. In this Section, a survey of the leakage reduction techniques in FPGAs proposed in the literature is presented.

3.5.1 Leakage Power Reduction in the Logic Blocks

In [49], a dual-VDD power reduction approach for FPGA logic and routing resources is proposed. The buffers in the FPGA fabric have the flexibility to either connect to a high or a low V_{DD} depending on the criticality of the net. Nets with high criticality are routed through logic and routing resources controlled by high-V_{DD} switches and less critical nets are routed through low-V_{DD} resources. Thus, in each LUT, the designer can select either of the two V_{DD}s according to the performance requirements. In addition, some of the logic elements can be hardwired directly to either of the supply lines. Figure 3.4 shows the different logic blocks available (Figure 3.4(a) high-V_{DD} block H-Block, Figure 3.4(b) low-V_{DD} block L-Block, and Figure 3.4(c) programmable-V_{DD} block P-Block).

The CAD flow that makes use of the proposed architecture is developed to efficiently select the values of the dual supply voltage, as pre-defined V_{DD} values were not used to increase the power savings. Furthermore, a level converter is used to transform the output of a low-V_{DD} LUT to the high V_{DD} value to avoid short circuit power dissipation. The average power savings achieved from this approach is about 14%. This approach suffers from the complexity of finding the correct value of the low-V_{DD} during each configuration. Moreover, the use of level converters adds

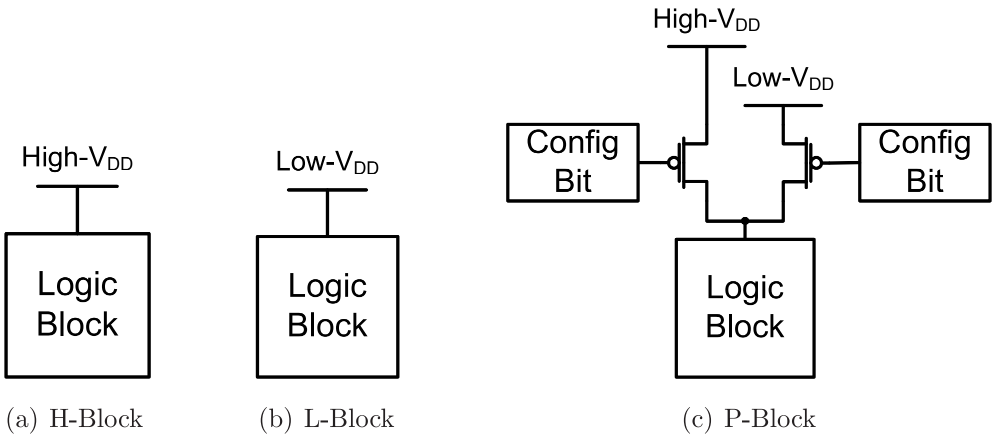


Figure 3.4: V_{DD} programmable low-power FPGA resources [49].

a huge area and power penalty to the final design. This approach also requires external circuitry to generate the two supply voltages needed.

The idea was further extended by the authors in [50] by including a two-bit control for the V_{DD} supply to enable power-gating for the unused FPGA resources. By enabling power gating, the percentage power savings was increased to 50%. Another extension for this methodology was presented in [51], where the authors proposed a dual-Vt fabric in addition to the dual-Vdd. In this architecture, the threshold voltage V_{TH} of the pass transistors inside the LUT is lowered according to the V_{DD} to minimize the performance losses due to lowering V_{DD}. The dual-Vdd dual-Vt approach achieves an average power savings of 13.6% and 14.1% for combinational and sequential circuits, respectively. In addition to the above mentioned issues for the dual-Vdd approach, the dual-Vt requires a double-well technology to support body biasing, which is an expensive process. A similar approach for leakage power dissipation was proposed in [52], where dual voltage is enabled by the use of header and footer devices to change the LUT voltage supply.

In [53,54], an active leakage power reduction methodology was proposed for both the logic blocks and the routing resources that use pass transistor multiplexers. The main idea behind that work is that the amount of leakage current depends on the inputs to the circuit. Hence, by manipulating the inputs, the unused parts of the FPGA can be placed in a low-leakage state. Moreover, by utilizing the complement of the signals, the authors managed to reduce the leakage of the used parts of the FPGA. The static probabilities of the signals were used to estimate the dominant states of the signals and then an optimization problem was developed to make that dominant state a low-leakage one. The average active leakage power reduction achieved by this method was 25% for a 90nm CMOS process.

In [55], a leakage reduction technique that depends on the use of sleep transistors to switch off the power supply from the unutilized parts of the FPGA was proposed. Moreover, the parts of the FPGA that exploit similar idleness periods are turned on/off during runtime. The logic blocks with similar idleness profile are placed close to each other in the FPGA fabric using a region constrained placement. However, the authors of [55] did not provide a method to automatically identify these logic blocks that have similar activity profiles. They resorted to a manual step, which in turn is inappropriate to be included in a CAD flow. Moreover, during the placement stage, the region constrained placement algorithm tries to place all those blocks with the same activity profile right beside each other, irrespective of the connections between these logic blocks and the other logic blocks that have a different activity profile. This approach can seriously affect the delays of the placed design, especially in designs with a large number of critical nets. Finally, the authors presented results based on only one benchmark without considering the fact that the amount of power saving depends on the benchmark used.

In [56], the authors proposed to use the unutilized pins inside the logic block to reduce the glitching power inside the LUT. The methodology is based on forcing the unused pins to constant values in such a way that glitches do not occur in the pass-transistor multiplexer. Moreover, the authors also use the placement and routing stages of the CAD flow to reduce the length of the wires that have transition probabilities.

3.5.2 Leakage Power Reduction in Routing Circuitry

The ability to program and reprogram the FPGA results in a huge overhead in the routing resources of the FPGA. As a result, the routing circuitry is the main source of power dissipation in FPGAs. A number of recent studies reported that 60%-70% of the total FPGA power dissipation is consumed in the routing resources [57].

In [58], two low leakage routing switch designs were proposed to operate in three different modes; high-speed, low-speed, or sleep mode, as shown in Figure 3.5. The main idea is based on the use of sleep transistors to block the supply voltage V_{DD} when the circuit is idle. The difference between the routing switches in Figure 3.5 is that the one in Figure 3.5(b) uses body biasing to change V_{th} of the PMOS transistors by tieing their bodies to the virtual ground line V_{VD}.

In the two configurations in Figure 3.5, both sleep transistors, MNX and MPX are on when the circuit is operating in the high-speed mode, thus acting as a

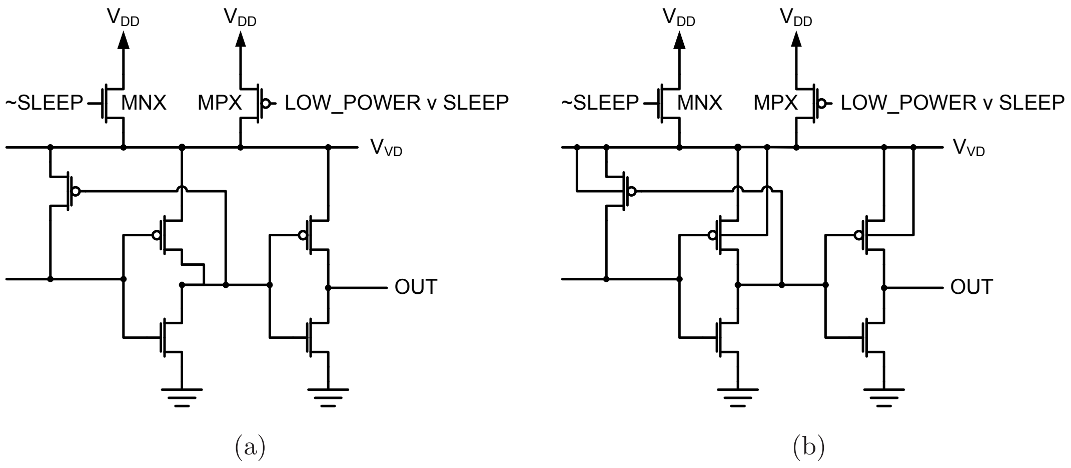


Figure 3.5: Programmable low-power routing switches [58].

transmission gate and driving the circuit with a full rail-to-rail voltage. MPX is turned off in the low-speed mode, while MNX is kept on, thus reducing the circuit drive to $V_{DD} - V_{th}$ to GND and lowering its speed. In the low-speed mode, the circuit power consumption, both dynamic and leakage power dissipation, is reduced. In the sleep mode, both MNX and MPX are turned off, thus blocking the supply from the circuit.

On average, the configuration in Figure 3.5(a) offers leakage and dynamic power reduction in the low-power mode by 36-40% and 28%, respectively, while the leakage reduction in the sleep mode is 61%. The routing switch in 3.5(b) reduces the average leakage and dynamic power dissipation by 28-30% in the low-power mode while offering a 36% area reduction.

In [57], the authors compared the pros and cons of several low-leakage techniques available in the ASIC domain and their chances of being applied to the FPGA domain. All of the leakage mitigation techniques were applied to the routing resources of the FPGA. The authors experimented with logic duplication in the memory cells of the routing resources and and achieved a two-fold decrease in leakage power dissipation with a 1.3X-2X increase in the FPGA area. A dual-Vt design of the routing resources, where LVT devices are used to route critical nets and HVT devices are used to route non-critical ones, achieved a leakage power reduction of 70% without any area penalty. Body biasing techniques, to change V_{TH} of the routing multiplexers pass transistors, achieved a 1.7X-2.5X leakage power reduction, with an area increase of 1.6X-2X. However, body biasing techniques are expensive, process wise, because of the need for an extra step during fabrication. Gate biasing was also used to lower the V_{GS} of the off devices below zero to minimize their subthreshold leakage current, leakage current is proportional to V_{GS}. A 2.5X-4X of leakage power reduction was witnessed by gate biasing with a negligible area penalty.

In [59], the authors proposed a heterogenous routing structure, where the wire lengths are varied and the devices used in the routing resources too are varied. A combination of transistors with high-V_{th}, regular device, devices with long channel lengths are used to trade off the final design performance with the leakage power dissipation. The CAD flow identifies the signals with low criticality and forces them to get rerouted to slower paths that use either high-V_{th} devices or devices with longer channel lengths. A similar approach was proposed in [60], however, only high-V_{th} devices were used.

3.6 Power Reduction Techniques in Commercial FPGAs

In the past few years, FPGA vendors started adding power reduction techniques to their commercial FPGAs. Most commercial FPGAs use SRAM cells with high-V_{th} devices to limit the power dissipation in the configuration SRAM. In this Section, the methodologies used in commercial FPGAs for power reduction in three of the most popular FPGAs will be presented.

3.6.1 Altera Stratix III Power Reduction Techniques

The Stratix III devices are equipped with the ability to operate at two core voltages; 1.1V and 0.9V. The programmable power supply powers the logic and routing resources, DSP blocks, memory blocks, and clock networks. However, the I/O pads are kept operating at a fixed voltage of 1.1V. The ability to operate at a lower supply voltage achieves both dynamic and leakage power dissipation since dynamic power is proportional to the square of the supply voltage while leakage power is proportional to the supply voltage.

In addition, the Stratix III devices offer supply voltage programmability inside the core by giving the designers the flexibility to change the supply voltage of individual logic blocks inside the device. In this case, performance is traded off with power dissipation. However, this property is only available for the logic blocks, thus limiting the power savings achieved. The main disadvantage of the dual core devices is that two external supply voltages are needed, which does not make such a device a strong choice in hand-held applications.

3.6.2 Xilinx Virtex-5 Power Reduction Techniques

The Virtex-5 devices use clock gating techniques for dynamic power dissipation reduction. The logic blocks in the Virtex-5 devices have an enable control for the input clock signals that can disable the clock when needed, thus reducing the dynamic power dissipation. Moreover, the clock tree is also equipped with a control to prevent further propagation of the clock signals when needed.

3.6.3 Actel IGLOO Power Reduction Techniques

A main advantage for IGLOO FPGAs is the use of flash for their configuration bits, compared to SRAM used in Stratix III and Virtex-5. Flash power dissipation is much lower than that of SRAM. Moreover, IGLOO devices are equipped with a Flash Freeze technology that enables the user to put the whole FPGA in a standby mode when needed. The Flash Freeze is programmable, so it can automatically bring the deice up again when certain conditions are satisfied. This technology is enabled by the non-volatile nature of the flash cells used in IGLOO. As a result, IGLOO devices offer significant dynamic and leakage power dissipation.

Chapter 4

Power Estimation in FPGAs

With power dissipation posing as an important factor in the design phase of FPGAs, power estimation and analysis techniques have become huge challenges for FPGA vendors. Power characterization is an important step in designing power efficient FPGA architectures and FPGA applications. Designers need a method to quantify the power advantage of the architectural design decisions without having to go through fabrication. Moreover, FPGA users need to check the power efficiency of the several possible implementations without actually going through the lengthy design phase.

Power models for FPGAs need to consider both components of power dissipation, dynamic and leakage power. To further improve the accuracy of the power model, all the subcomponents of both dynamic power (switching, short circuit, and glitch power), and leakage power (subthreshold and gate leakage power), need to be accounted for. In addition, other factors that affect power dissipation, including spatial correlation and input dependency of leakage power, should be considered, especially in the sub-nanometer regime since their impact is highlighted as the CMOS minimum feature size is scaled down.

In this Chapter, a newly proposed power estimation methodology under spatial correlation for FPGAs is presented. Moreover, the proposed power estimation technique accounts for glitching power as well as the dependency of leakage power on the input states. The Chapter starts by a quick introduction to the power estimation problem in general in Section 4.1. An overview of the main research projects that targeted power estimation in the VLSI in general and FPGAs in particular is given in Sections 4.2 and 4.3, respectively. Section 4.4 describes the impact of spatial correlation on power estimation. The proposed algorithms for power estimation under spatial correlations and glitching power estimation are presented in

Sections 4.5 and 4.6 and Section 4.7, respectively. Section 4.8 describes the power modeling strategy used in this work. Finally, the results and the discussions of this Chapter are presented in Section 4.9.

4.1 Introduction

Modern CMOS processes suffer from two dominant sources of power dissipation; dynamic and leakage power. Dynamic power dissipation can be divided into switching, glitch, and short circuit power dissipation, while leakage power dissipation can be further divided into subthreshold leakage and gate leakage power dissipation. Historically, CMOS circuits were dominated by dynamic power dissipation, however, by the 65nm CMOS process, leakage power is expected to dominate the total power dissipation, as explained in Chapter 3. In addition, further down the scaling road, gate leakage power is expected to surpass subthreshold power dissipation, especially at lower operating temperatures, as predicted by the semiconductors roadmap issued by the ITRS [1], unless high K materials are used to implement the devices gates.

All sources of power dissipation in CMOS circuits exhibit significant state dependency. In order to develop an accurate power model, accurate information about signal probabilities needs to be available. Several works in the literature have addressed this problem and a complete survey was presented in [61].

The switching component of the dynamic power dissipation is expressed as

$$Power_{dyn} = \frac{1}{2} \times f_{clk} \times V_{DD}^2 \times \sum_{i=1}^{n} C_i \times \alpha_i \ , \tag{4.1}$$

where f_{clk} is the circuit clock frequency, V_{DD} is the supply and swing voltage, C_i is the capacitance of the i^{th} node in the circuit, and α_i is a measure of the number of transitions per clock cycle experienced by node i. As a result, the problem of switching power estimation resolves to finding the capacitance and measure of the number of transitions at every node.

Glitching power occurs because of the spurious transitions at some circuit nodes due to unbalanced path delays. Consequently, glitching power results in an increase in the number of transitions at every circuit node that is susceptible to glitches. Hence, the impact of glitching power can be modeled just by adding a factor to the transitions estimate α in Eq. (4.1). Finally, short circuit power dissipation is the power dissipated due the presence of a direct current path from the power supply to the ground during the rise and fall times of each transition. Hence, short circuit power is a function of the rise and fall times and the load capacitance. Several research projects have been directed to provide an accurate estimate of short circuit power dissipation [62–65]. However, the simplest method to account for short circuit power dissipation is to set it as a percentage of the dynamic power dissipation, usually 10% [66].

4.2 Power Estimation in VLSI: An Overview

The power estimation is defined as the problem of evaluating the *average* power dissipation in a digital circuit [61]. Power estimation techniques fall into two main categories: *simulation-based* or *probabilistic-based* approaches.

4.2.1 Simulation-Based Power Estimation Techniques

In simulation-based techniques, a random sequence of input vectors is generated and used to simulate the circuit to estimate the power dissipation. The first approaches developed were based on the use of SPICE simulations to simulate the whole circuit using a long sequence of input vectors [67,68]. However, the use of such methods in today's VLSI industry is impractical, especially with the huge levels of integration achieved, that it might take days if not weeks to simulate a complete chip using SPICE. Moreover, these methods are significantly pattern dependent due to the use of a random sequence of input vectors. If an intelligent method is used to select the input sequence, these methods would provide the most accurate power estimation.

Several simplifications of these methodologies were proposed to reduce the computational complexity of power estimation [69–72]. The methods still rely on simulations, but instead of using SPICE simulations, other level of circuits simulations were performed including switch-level and logic-based simulations. These methodologies trade accuracy for faster runtime. In order to perform these simulations, the power supply and ground are assumed constant. However, these methods still suffer from pattern dependency since there is a need to generate a long sequence of input vectors to achieve the required accuracy. Although these simulations are more efficient than SPICE simulations, yet they are somewhat impractical to use for large circuits, especially if a long input sequence is used to increase the method accuracy.

In order to solve the pattern dependency problem of simulation-based power estimation methods, several *statistical* methods had been proposed [73–78]. These statistical methods aim to quantify two parameters: the *length of the input sequence* and the *stopping criteria* for simulation, required to achieve a predefined power estimate accuracy.

The earliest of these studies focused on the use of Monte Carlo simulations to estimate the total average power [73]. A random sequence of N input vectors was independently generated and used to simulate the circuit. Let $\bar{p}$ and s be the average and standard deviation of the power measured over a time period T and P_{av} is the average power. Hence, the error in the average power estimated can be expressed with a confidence of $(1-\alpha)\times 100\%$ as

$$\frac{|\bar{p}-P_{av}|}{\bar{p}} < \frac{t_{\alpha/2}s}{\bar{p}\sqrt{N}} \, , \tag{4.2}$$

where $t_{\alpha/2}$ is generated from the t-distribution with $(N-1)$ degrees of freedom [73]. Hence, to tolerate a percentage error of ϵ, the required length of the input sequence is expressed as [73]

$$N \geq \left(\frac{t_{\alpha/2}s}{\epsilon\bar{p}}\right)^2 . \tag{4.3}$$

An extension of this work was proposed in [74] to provide an estimate of the average power dissipation in each gate instead of the whole circuit.

A disadvantage of the use of Eq. (4.3) is that the value of N can not be estimated before simulation. In [75], the authors proposed a different formulation of the required length on input vectors a priori to simulation. A single-rising-transition approximation was adopted for circuits that do not experience glitches, and the value of N for an error of ϵ and confidence of $(1-\alpha)$ is approximated by [75]

$$N \approx \frac{z^2_{1-\alpha/2}}{\epsilon^2} \, , \tag{4.4}$$

where $z^2_{1-\alpha/2}$ is the $100\times(1-\alpha/2)^{\text{th}}$ percentile of the standard normal distribution. For circuits with glitches or large logic depths, N for an error of ϵ and confidence of $(1-\alpha)$ is approximated by [75]

$$N \approx \frac{4z^2_{1-\alpha/2}}{49\epsilon^2} \times (t+1)^2 \, , \tag{4.5}$$

where t is the maximum number of transitions that the circuit can experience per each input vector.

Another disadvantage with Eq. (4.3) is the large number of input vectors needed to achieve the required accuracy, since it depends on the square of the sample variance. Moreover, as the resulting power estimate deviates from the normal distribution, the simulation might terminate early, thus compromising the accuracy of the results. In [76,78], solutions for these two issues were proposed using a Markov chain to generate the input sequence. The resulting sequences are more compact than the ones used in [73] and provide a reduction in the simulation time by orders of magnitude while keeping the estimated average power within 5%.

Another statistical method proposed in the literature to provide the needed length for the input sequence is based on the least-square estimation methods [77]. The authors viewed the estimation problem of the input sequence as an approximation problem and explored the use of sequential least square and recursive least square to solve the problem of finding the input sequence that has minimum variance and without making any probabilistic assumptions about the data. It was reported in [77] that least square algorithms need much smaller number of iterations, *i.e.*, smaller input sequence, to provide a close estimate of the average power dissipation to that of [73].

4.2.2 Probabilistic-Based Power Estimation Techniques

Probabilistic power estimation techniques have been proposed to solve the problem of pattern dependency of simulation-based approaches. In these techniques, signal probabilities are propagated through the circuit starting from the primary inputs until the outputs are reached. These estimation techniques require circuit models for probability propagation for every gate in the library.

The first ever probabilistic propagation model was proposed in [79]. In this model, a *zero-delay assumption* was considered, under which the delay of all logic gates and routing resources was assumed zero. The switching activity of node x was defined as the probability that a transition occurs at x. The *transition probability* of x, $P_t(x)$ is calculated according to

$$P_t(x) = 2 \times P_s(x) \times P_s(\bar{x}) = 2 \times P_s(x) \times \left[1 - P_s(x)\right] , \tag{4.6}$$

where $P_s(x)$ and $P_s(\bar{x})$ are the probabilities that $x = 1$ and $x = 0$, respectively. In adopting Eq. (4.6), the authors assume that the values of the same signal in two consecutive clock cycles are independent, which is referred to as *temporal independence*. Moreover, the signal probabilities at the primary inputs are propagated into the circuit while assuming that all internal signals are independent. This assumption is referred to as *spatial independence*. Furthermore, [79] ignores glitching power since a zero delay model was adopted.

In [80], the authors proposed the use of the *transition density* to represent the signal probabilities more accurately than the simple transition probability in Eq. (4.6). The transition density is defined as the average number of transitions per second at a node in the circuit. The transition density at node x, $D(x)$, is given by

$$D(x) = \lim_{T\to\infty} \frac{n_x(T)}{T} \ , \tag{4.7}$$

where $n_x(T)$ is the number of transitions within time T. The authors in [80] formulated the relationship between the transition density and the transition probability by

$$D(x) \geq \frac{P_t(x)}{T_c} \ , \tag{4.8}$$

where T_C is the clock cycle. Hence, the transition probability will always be less than the transition density, thus underestimating the power dissipation. The transition density at node y with a set of inputs $x_0, x_1, ..., x_n$ is given by the following set of relationships

$$D(y) = \sum_{i=1}^{n} P(\frac{\partial y}{\partial x_i}) D(x_i) \ , \tag{4.9}$$

$$\frac{\partial y}{\partial x_i} \triangleq y|_{x_i=1} \oplus y|_{x_i=0} \ , \tag{4.10}$$

where $\frac{\partial y}{\partial x_i}$ is the boolean difference of y with respect to its i^{th} input and $\oplus$ denotes exclusive OR operation. In order to evaluate the boolean difference at each node, the probabilities at each node need to be propagated through the whole circuit. It should be noted that the use of (4.9) only provides a better estimate for the number of transitions than the transition density given in Eq. (4.6) and this model still suffers from both spatial and temporal independence assumptions.

In [81], the authors proposed the use of *Binary Decision Diagrams* (BDDs) to account for spatial and temporal correlations. The regular boolean function of any logic gate stores the steady-state value of the output given the inputs. However, BDDs are used to store the final value as well as the intermediate states, provided that circuit delays are available beforehand. As a result, such a probabilistic model can predict the signal probabilities at each circuit node under spatial and temporal correlation. However, this technique is computationally expensive and only practical for moderate-sized circuits. In addition, a BDD is required for every logic gate, if some gates have a large number of intermediate states, then this technique might become impractical even for medium-sized circuits.

Several other research projects have been proposed in the literature to handle spatial and temporal correlations [82–87].

4.3 A Survey of FPGA Power Estimation Techniques

Power modeling in FPGAs did not receive wide attention in the literature, especially for generic FPGA architectures. Several works that targeted specific FPGA architectures includes [41, 48, 88–91]. These works are only applicable to one architecture because they depend on the specific architecture details to extract the power dissipation. Moreover, all of these works targeted only dynamic power estimation in FPGAs, except for [48] that provided an insight into leakage power dissipation in a 90nm CMOS commercial FPGA. On the other hand, power dissipation in general architecture FPGA has been targeted in a limited number of research projects [92–96]. However, all of these research projects did not provide an *analytical* methodology for estimating the effect of spatial correlation on the total power dissipation.

In [41], a Xilinx XC4003A$^{\text{TM}}$ FPGA was used to study the power breakdown inside the FPGA. The power reported was the actual power measured recorded from the physical FPGA itself. The authors of [88] introduced a technology dependent empirical correction factor for dynamic power estimation in the Xilinx Virtex$^{\text{TM}}$ FPGA. The factor introduced was used to adjust the switching activity estimated through regular probabilistic analysis similar to [80]. The power dissipation in the FPGA under consideration was physically measured for different benchmarks and the power dissipation was calculated in a similar manner to [80] for the same benchmarks. The ratio between the measured and estimated power values was calculated as the correction factor for power estimation. The main reason for using that correction factor is to account for the factors that affect power dissipation, including temporal and spatial correlations, that are not captured in the probabilistic calculations presented in [80]. It should be noted that this power estimation methodology is heavily dependent on the benchmarks used to compute the value of the correction factor as different types of benchmarks will result in different value of the factor. It should be noted that both [41,88] do not account for leakage power dissipation in the FPGA under consideration.

A model for dynamic power dissipation in Xilinx Virtex-II$^{\text{TM}}$ FPGAs was described in [89]. The switching activity was estimated using logic simulation of some practical input vectors obtained from the users. The different node capacitances were evaluated from post-silicon capacitance extraction techniques. An extension of this work was presented in [90], where the node capacitances were evaluated using simple RC models for the Xilinx Spartan-3$^{\text{TM}}$ FPGA. The methodology is still simulation-based since it relies on logic-level simulation to find the node activities.

The power model proposed in [91] tries to predict accurate switching activities in Xilinx FPGAs using curve fitting theories and empirical formulae. The main goal of this work is to account for glitches in the transitions activity used in dynamic power calculations. A prediction function was proposed to calculate the change in transitions activity to account for glitching in the form of

$$\begin{aligned} PR_i = \;& \alpha \times \text{GEN}_i + \beta \times \text{GEN}_i^2 + \gamma \times \text{PROP}_i + \upsilon \times \text{PROP}_i^2 \\ & +\nu \times D_i + \xi \times D_i^2 + \eta \times \text{PROP}_i \times \text{GEN}_i \\ & +\iota \times \text{GEN}_i \times D_i + \rho \times \text{PROP}_i \times D_i + \phi \, , \end{aligned} \tag{4.11}$$

where α, β, γ, υ, ν, ξ, η, ι, ρ, ϕ are scalar constants, GEN_i is a parameter used to quantify the amount of glitches *generated* at node i, PROP_i represents the amount of glitch *propagation* at node i, and D_i is a parameter to represent the depth of node i. The authors used a commercial Xilinx Virtex-II$^{\text{TM}}$ PRO FPGA and using a predefined input sequence and a commercial simulation tool, generated the number of transitions experienced by every gate per clock cycle in several FPGA benchmarks. Afterwards, using curve fitting theories, the constants in Eq. (4.11) were evaluated.

Although the method proposed in [91] tries to formulate the impact of glitches on dynamic power estimation in FPGAs, it has several drawbacks. Firstly, the method is architecture dependent and can not be readily used to estimate dynamic power dissipation in other FPGA architectures than the one used. Moreover, in order to apply this method on a new architecture, the linear regression model needs to be trained, hence, there is a need for a reference power estimator to train the linear regression model. In addition, the model accuracy is heavily dependent on the type of circuits used to evaluate the curve fitting parameters, which makes it very susceptible to errors for the different circuit types.

A study of the leakage power dissipation in the CMOS 90nm Xilinx Virtex-II$^{\text{TM}}$ FPGA was presented in [48]. The leakage power modeling was performed using lookup tables of HSpice simulations. The leakage power for every input vector measured, obtained using HSpice simulations, was recorded and used to study the impact of the state dependency and utilization on FPGA leakage power dissipation. The study showed that the input state dependency can vary leakage power in modern FPGAs by about 60%. Moreover, a breakdown of leakage power dissipation in the different parts of the FPGA was provided. It should be noted that this study ignored the effect of signal correlations on leakage power. Moreover, the authors of [48] did not evaluate the total leakage power dissipation in the whole FPGA, only the leakage power per logic block was evaluated.

The power models that target general architectures consider dynamic, leakage, and short circuit power dissipation in FPGAs. The first power model for generic FPGA architectures was proposed in [92]. This power model is analytic in nature, making it very easy to implement with fast runtime. However, for this model to have such fast runtime, several approximations were assumed by the authors. Firstly, for dynamic power estimation, the authors of [92] assume spatial and temporal independence among the internal design signals, as well as ignoring the impact of glitching power. Secondly, the leakage power was calculated across all the transistors in the circuit while considering the V_{GS} to be half the threshold voltage V_{TH}, thus, significantly reducing the accuracy of the leakage power estimation. Moreover, the state dependency of leakage power was not considered in that model, which has a significant impact on FPGAs built using current nanometer CMOS technologies, as explained in [48].

In [93–95], the authors presented another FPGA power model for generic FPGAs. Leakage power dissipation was calculated through the use of lookup tables, that do not consider the state-dependency of leakage power. Moreover, the power model depended on logic simulation to estimate the switching activities of the different circuit nodes. Although this method can achieve high accuracy, its computational cost is quite high. The authors tried to limit the execution time by limiting the number of input vectors to 2000, irrespective of the circuit size and its number of inputs. However, this approach sacrifices the accuracy of the algorithm significantly as explained in [97].

The input dependency of leakage power dissipation in generic FPGAs was first addressed in [96]. The authors proposed a leakage power model based on the BSIM4 models while accounting for the state dependencies. However, spatial correlation among internal signals was not addressed. Moreover, the authors used some empirical constants in the power formulation obtained using curve fitting, thus, rendering the power model technology dependent. Finally, The temperature dependence of power dissipation in FPGAs was studied in [98]. The authors tried to estimate a factor that captures the dependence of total power dissipation on the temperature using empirical experimentation.

4.3.1 Commercial FPGA Power Estimation Techniques

Commercial FPGA vendors offer a variety of power estimation techniques for customers. There are basically two categories of commercial FPGA power estimation techniques: device-specific spreadsheets [99–101] and CAD-based power estimation techniques [100, 102, 103].

4.3.1.1 Spreadsheet Power Estimation Tools

FPGA power spreadsheets analyze both the leakage and dynamic power dissipation in the FPGA. This method of power estimation is usually used in the early stages of the design process to give a quick estimate of the power dissipation of the design [99–101]. In spreadsheet-based power estimators, the users must provide the clock frequency of the design and the toggle percentage for the logic blocks. Consequently, this method gives a rough approximation of power and requires designers to thoroughly understand the switching activity inside their circuits.

For dynamic power computation, designers provide the average switching frequency α for all the logic blocks, or for each module in the design. Coefficients for adjusting the dynamic calculation are provided in the device data sheet. The total dynamic power is calculated according to

$$P_{\mathrm{dyn}} = K \times f_{clk} \times V_{DD} \times \sum_{\mathrm{all\_components}} \alpha_i C_i V_{\mathrm{swing},i} \ , \tag{4.12}$$

where K is the coefficient used in adjusting the power estimate for each device family. The value of the node capacitance is usually provided as an average value for each family in the power spreadsheet.

In modeling leakage power dissipation, spreadsheets list the leakage power of each FPGA component $P_{\mathrm{leak\_per\_component}}$, and the total leakage is calculated by summing the leakage power for each component used in the design according to

$$P_{\mathrm{leak}} = \sum_{\mathrm{all\_components}} P_{\mathrm{leak\_per\_component}} \ . \tag{4.13}$$

The value of $P_{\mathrm{leak\_per\_component}}$ is provided for each device family.

The total estimated power is the summation of the dynamic and leakage power evaluated using Eq. (4.12) and (4.13), respectively.

4.3.1.2 CAD Power Estimation Tools

Commercial CAD power estimation tools rely on cycle accurate simulations to capture the switching at each node in the design [100, 102, 103]. The user specifies a simulation testbench, and the design is simulated using logic simulators. The number of transitions per clock cycle and logic probability are computed at each node during the logic simulation. Although this power estimation technique provides a better estimate for power dissipation than spreadsheets, the runtime of this technique is very long and the power estimate accuracy is dependent on the length of the test vector used.

4.4 Spatial Correlation and Signal Probabilities Calculations

The signal probability can be computed at the output of each logic block through simple probabilistic calculations to evaluate the probability of the output of the logic block being high. In all of the available probabilistic power models for FPGAs, the inputs to any specific logic block are assumed independent, the spatial independence assumption. This assumption is made to simplify the probabilities calculation. However, the spatial independence assumption reduces the accuracy of any analytical power model by overestimating the signal probabilities, as will be explained later. As an example, for the small circuit shown in Figure 4.1, assuming that the probability of $A = 1$ is 0.5, dynamic power estimators operating under the spatial independence assumption will calculate the probability of B being high as

$$P(B = 1) = P(A = 1) \times P(\overline{A} = 1) = 0.5 \times 0.5 = 0.25 \,, \tag{4.14}$$

thus resulting in a transition probability of 0.375 ($2 \times 0.25 \times 0.75$), according to Eq. (4.6). However, by clear inspection of Figure 4.1, this circuit suffers from a *reconvergent path*. Both A and $\overline{A}$ can never be '1' at the same time, hence, the probability of B being high should be zero, assuming the zero delay model. It should be noted that such a structure is common in FPGA circuits which will be depicted later in Table 4.4.

As noticed from the previous example, the spatial independence assumption can significantly affect the dynamic power calculations. If spatial correlation is to be taken into consideration, conditional probability should be used to evaluate the

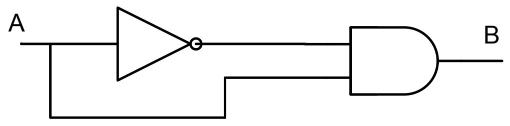


Figure 4.1: A circuit that exhibits spatial correlation through reconvergent paths.

probability of all signals. As an example, considering the circuit in Figure 4.1, the probability of B being high is formulated as

$$\begin{aligned} P(B=1) = \quad & P(A=1|A=1) \times P(\overline{A}=1|A=1) \times P(A=1) \\ & + P(A=1|A=0) \times P(\overline{A}=1|A=0) \times P(A=0) \;, \end{aligned} \tag{4.15}$$

which resolves to zero. From the above example, it can be deduced that the spatial independence assumption can significantly affect the accuracy of power estimation in VLSI circuits in general. Moreover, by inspecting Eq. (4.15), it can be noticed that spatial correlation will cancel one or more of the conditional probabilities listed. Hence, the impact of spatial independence will always be towards overestimating the power dissipation by overestimating the signal probabilities. Thus, the spatial independence assumption costs the designers in terms of over-design to account for the overestimated power dissipation.

In this work, a methodology to calculate the signal probabilities under spatial correlation is proposed. In order to consider spatial correlation for power calculations, such reconvergent paths, as the one shown in Figure 4.1, need to be identified, as discussed in Section 4.5, and their signal probabilities corrected accordingly, as explained in Section 4.6. The proposed methodology is explained in the following two sections.

4.5 Exploration Phase: Locating Spatial Correlation

Spatial correlation among signals in VLSI circuits occurs whenever 2 signals are correlated. Correlations arise when 2 or more signals share a common driver or a common parent logic block (x) and are connected as inputs to another logic block (y), *i.e.*, reconvergent paths. If the circuit is converted to a cyclic graph with the gates as the nodes and the signal wires as the edges, the connections between x and y form a cycle with 2 paths, as shown in Figure 4.2. Hence, the detection of signals that might exhibit spatial correlation resolves to identifying possible cycles in the circuit. For example, in Figure 4.2, a cycle would be detected that goes through $A \rightarrow \bar{a} \rightarrow \bar{A} \rightarrow b \rightarrow A$.

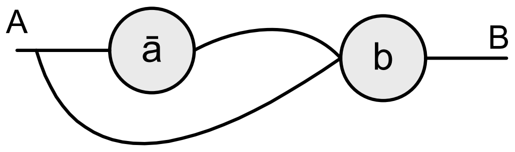


Figure 4.2: A graph representation of the circuit in Figure 4.1.

In this work, depth-first search algorithm is used to identify such loops in the design. The algorithm starts with the circuit primary inputs i, and depth-first search is used to navigate through all the logic blocks that share a path with each primary input. Whenever a logic block j is visited, it is marked with the name of the primary input used in this search. When a logic block j gets visited twice, this means that there are two paths from the current primary input i to logic block j. Afterwards, the two different paths are recorded as cycles that might result in spatial correlation among the inputs to logic block j. By employing the depth-first search algorithm, the complexity of the exploration phase gets significantly reduced. A pseudocode for the exploration phase is shown in Figure 4.1.

As a result of the algorithm listed in Algorthim 4.1, every logic block that experiences spatial correlation among its inputs will have all the paths that contribute to the reconvergent paths recorded. However, these paths can be very long, especially in circuits with long logic depth. As a result, a cleanup stage is performed on these paths to remove the redundancy in these paths. For example, in Figure 4.2, the two paths recorded for B would be $A \rightarrow \bar{A}$ and A. If A is not a primary input, then the two paths will also include logic blocks that generate the inputs to A in the circuit and so on until the primary inputs. Hence, the cleanup stage deletes all the nodes in the path that are common and only keeps the fanout stem of the reconvergent path, which is A in this example.

4.6 Proposed Signal Probabilities Calculation Algorithm Under Spatial Correlation

Once all of the cycles in the circuit that contribute to spatial correlation are identified and recorded, the algorithm starts correcting the signal probabilities for all the logic blocks that have correlated inputs. As a first step, all of the logic blocks are sorted in a topological order according to their connections. In this ordering,

Algorithm 4.1 The exploration phase pseudocode used to identify reconvergent paths in a circuit.

```
Function: explore
for all primary inputs $i$ do
  for all blocks $j$ connected to $i$ do
    depth_first($j$,$i$)
  end for
end for
return
Function: depth_first($j$,$i$)
if $j$ has been visited before by $i$ then
  a cycle is found
  record the path $i \rightarrow j$
else
  label $j$ as visited by $i$
end if
for each block $k$ connected to $j$ do
  depth_first($k$,$i$)
end for
return
```

the primary inputs come in first, followed by those logic blocks that only have primary inputs as their inputs, and so on. This ordering is essential because it is the same order at which the signal probabilities, and hence, transition densities are calculated according to Eq. (4.9).

In the second step, all the signal probabilities in the design are calculated based on the spatial independence assumption among the circuit signals. Processing the logic blocks according to the topological ordering performed earlier ensures that whenever a logic block is processed, all of its inputs have already been processed and signal probabilities have been calculated for them.

In the third step, the logic blocks that have cycles are examined. For each logic block that has cycles, the number of different fanout stems for the cycles are recorded. As an example, in Figure 4.3, logic block E will have 2 cycles with A and B being the fanout stem of the 2 cycles. The first cycle has A as the first path and $A \rightarrow C$ as the second path. The second cycle has $B \rightarrow C$ as the first path and $B \rightarrow D$ as the second path. It should be noted that these 2 cycles are not independent, they both share C, thus making the calculation of the signal probabilities more complex.

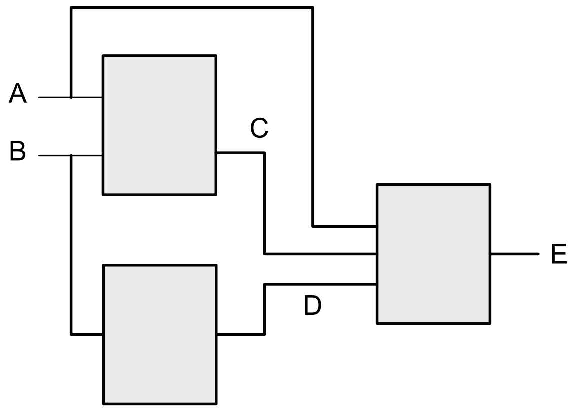


Figure 4.3: A circuit that exhibits spatial correlation.

If the number of fanout stems for logic block i is n, then for the conditional probability calculations, there are 2^n different conditional probabilities to calculate for every input to the logic block that experiences reconvergent paths. For the circuit in Figure 4.3, the conditional probabilities that need to be calculated are:

$$\begin{array}{ll} P(C=1|A=0\&B=0), & P(D=1|A=0\&B=0) \\ P(C=1|A=0\&B=1), & P(D=1|A=0\&B=1) \\ P(C=1|A=1\&B=0), & P(D=1|A=1\&B=0) \\ P(C=1|A=1\&B=1), & P(D=1|A=1\&B=1) \end{array} \tag{4.16}$$

The algorithm starts by assuming the first combination for the fanout stems, which is $A = 0$ and $B = 0$ in this case, and then propagates through all the cycle paths recorded for logic block E and evaluates their probabilities. This phase is stopped when all of the probabilities of the inputs to block E are evaluated. Afterwards, the conditional probability for block i is calculated and multiplied by the probability of occurrence of the tested input combination, *i.e.*, $P(A = 0\&B = 0)$ in the example in Figure 4.3. This process continues until all of the 2^n combinations are processed. The probability of logic block E output being high under spatial correlation, will be the summation of the probabilities evaluated for each input combination for block E according to Eq. (4.16). This process continues on until all of the logic blocks in the design are processed. The importance of the cleanup phase mentioned in Section 4.5 is that it reduces the number of probabilities calculations to a greater extent by getting rid of the common paths. A pseudocode for the algorithm is listed in Algorithm 4.2.

Algorithm 4.2 Probabilities calculation under spatial correlation algorithm.

```
Order_Logic_Blocks()
Calc_Prob_Under_Independence()
for each block $i$ with cycles do
  $n$ = Find_Num_Fanout_Stems($i$)
  $\text{prob}_i = 0$
  for $j$=0 : $j = 2^n$ do
    Adjust_Prob_Fanout_Stems($j$)
    prob_fanout_stems = Find_Prob_Fanout_Stems($j$)
    $\text{prob}_i = \text{prob}_i$ + Find_Prob($i$) $\times$ prob_fanout_stems
  end for
end for
```

The function `Adjust_Prob_Fanout_Stems(`j`)` in Algorithm 4.2 converts the integer j to its binary equivalent and adjusts the probabilities of the fanout stems accordingly. As an example, for the circuit in Figure 4.3, if $j = 2$, then $P(A) = 1$ and $P(B) = 0$. `Find_Prob_Fanout_Stems(`j`)` finds the probability of the fanout stems combination given by j, *e.g.*, when $j = 2$, `prob_fanout_stems =` $P(A = 1) \times P(B = 0)$, assuming that A and B are independent. If A and B are not dependent, then the probability of $P(A) = 1$ and $P(B) = 0$ was calculated by the algorithm when it processed A and B. `Find_Prob(`i`)` evaluates the probability of block i for the current input combination of the fanout stems. This is performed by evaluating the probabilities of all the logic blocks in the paths contributing to the cycles connected to block i. It should be noted that similar algorithms had been used in the literature for fault detection in VLSI circuits [104–107].

By inspecting the algorithm in Algorithm 4.2, it can be deduced that the complexity of the algorithm is $O(m \times 2^n \times k)$, where m is the number of logic blocks with cycles, n is the number of fanout stems that any logic block can have, and k is the maximum number of cycles that any logic has. Signal probabilities under spatial correlation depends on the maximum number of cycles handled. If all the cycles at the input of any logic block are handled, then the algorithm would have the highest accuracy at the expense of the increased complexity and execution time. Hence, having a maximum for the number of cycles to be considered by the algorithm would result in slightly less accurate value for the probabilities but with a faster runtime.

The proposed algorithm was executed first for all the FPGA benchmarks listed in Table 4.4 while considering all the cycles present in the design. Afterwards a maximum limit on the number of fanout stems for each node to be considered was set and the algorithm was executed several times for different maximum values. The paths that are rejected are those that their children have the lowest probabilities. For example, in Figure 4.3, if one of the paths is to be rejected, if the $P(C = 1) < P(B = 1) < P(A = 1)$, then the paths that have C in them are rejected. This decision is taken because they will have the least impact on the final probability [74]. It was found that when the number of fanout stems handled by the algorithm was limited to five, the accuracy of the signal probabilities calculated, when compared to the first case, had an error below 4%, while the algorithm execution time got reduced significantly when compared to the case with all the cycles. Hence, in this work, the number of fanout stems handled is limited to five to reduce the algorithm complexity while achieving the best accuracy. The results of this experiment are presented later on in Section 4.9.

4.7 Power Calculations due to Glitches

Glitches occur in VLSI circuits due to the difference in the arrival times of the inputs to any logic block, *e.g.*, both A and C have different arrival times as inputs of E in Figure 4.3. To identify the logic blocks that might generate glitches, the post layout arrival times of all the design signals are extracted using VPR [28]. VPR takes as input the capacitances and resistances of the different wire segments in the FPGA fabric under considerations. Using this information, VPR calculates the arrival times for every signal at every circuit node using simple Elmore delay calculations. There are two conditions needed for glitch generation: (i) the differences in the arrival times should be larger than the intrinsic delay of the logic cell and (ii) the logic implemented results in a glitch. Moreover, it should be noted that glitches are filtered out of the circuit through re-timing elements like latches and buffers.

The proposed algorithm for glitch probability calculations consists of three phases: glitch generation, glitch propagation, and glitch termination. Starting with the logic cells connected to the primary inputs, and parsing the circuit in a depth-first strategy, when conditions (i) and (ii) are satisfied, glitches are generated at the output of the logic cell. For instance, in Figure 4.3, if a glitch at E occurs when A switches to '1' and C to '0', then the probability that such a glitch occurs is

$$P_g(E) = P(C = 0|A = 1) \times P(A = 1) \ , \tag{4.17}$$

It should be noted that the algorithm for calculating the signal probabilities under spatial correlation calculates the conditional probability in Eq. (4.17) if they are correlated, otherwise, Eq. (4.17) resolves to $P(C = 0) \times P(A = 1)$.

When a glitch from the output of one cell is fed to the input of the next cell, that glitch propagates only if the logic function of the second cell allows the glitch to. The probability of that certain glitch will propagate is equal to the probability of the glitch multiplied by the conditional probabilities of the inputs needed to propagate the glitch. The proposed glitch propagation algorithm keeps on parsing the circuit by the depth-first search until the probability of glitch propagation is less than 0.01, at which point the glitch is dropped, glitch termination. It should be noted that no new probabilities are calculated by the glitch processing algorithm.

4.8 Signal Probabilities and Power Dissipation

In this Section, the effect of signal probabilities on the components of power dissipation is discussed. Moreover, the method used in this work to model both dynamic and static power dissipation is presented.

4.8.1 Dynamic Power Dissipation

Utilizing the transition density in calculating the dynamic power dissipation using Eq. (4.1) results in [80]

$$P_{dyn} = \frac{1}{2} \times f_{clk} \times V_{DD}^2 \sum_{i=1}^{n} C_i D_i \ , \tag{4.18}$$

where D_i is the transition density of node i. The transition density is calculated from the signal probabilities using Eq. (4.9) and (4.10) [80]. From the above equations, it is shown that the calculation of the transition density depends heavily on the proper calculation of the signal probabilities. Consequently, spatial correlation directly affects the accuracy of the transition density calculation. The transition density at each node is calculated efficiently by simple propagation algorithms that depend on the signal probabilities at each node [80].

The above discussion is valid for combinational logic, however, for sequential circuits, some approximations are made. In [108], iterations were used to calculate the output probability of sequential feedback loops. Initially, the input and output probabilities are set to the same value, then by performing several probability calculation iterations, the output probability is adjusted. The authors of [108] also demonstrated that the transition probability of the feedback loop is within 5% compared to the exact transition probability value, provided that sufficient number of iterations are performed. In this work, the same methodology proposed in [108] and used in [66] are used to calculate the transition density at the output of sequential feedback loops. It should be noted that this methodology has been selected due its ease of implementation, rather that the quality of its results.

The capacitances used in Eq. (4.18) are extracted from commercial CMOS processes using the post-layout capacitance extractor available in Cadence tools. A small fabric is designed using the fully custom design flow and the layout of the circuit was performed together with the routing tracks and multiplexers. Afterwards, Cadence is used to extract the resistances and capacitances of all the routing tracks with different lengths in our FPGA architecture. The chosen frequency in this work to calculate the power dissipation is 600MHz. This value was chosen because it corresponds to the maximum clock frequency that state-of-the-art FPGAs operate at [11, 109].

4.8.2 Leakage Power Dissipation

In FPGAs, logic functions and routing resources are implemented using pass-transistor based multiplexers, as shown in Figure 4.4. In the logic resources Look-Up Tables (LUTs), the inputs (S0-S3) are connected to SRAM cells, while the controls (C0-C1) are connected to the inputs of the LUT. However, in the FPGA routing resources, the inputs are connected to the signals to be routed and the controls are connected to SRAM cells that control the routing switch.

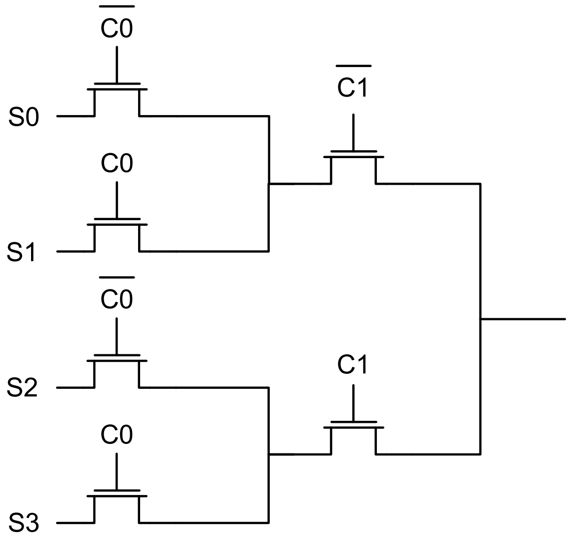


Figure 4.4: A 2:1 pass transistor logic multiplexer.

In [53], the authors demonstrated the dependence of leakage power dissipation in the pass-transistor based multiplexers on the input vector. For a 90nm CMOS process, the leakage power dissipation in the pass-transistor multiplexer in Figure 4.4 can vary by 14X depending on the input combination [53]. Two main factors affect the threshold voltage of the pass-transistors, hence, leakage power dissipation in these multiplexers; body effect and drain induced barrier lowering (DIBL). The effect of body bias on V_{TH} is formulated as

$$V_{TH} = V_{TH0} + \gamma\Big(\sqrt{\Phi_s - V_{BS}} - \sqrt{\Phi_s}\Big) \;, \tag{4.19}$$

where V_{TH0} is the ideal V_{TH} at zero V_{BS}, γ is the body bias coefficient, and Φ_s is the surface potential. Having a negative V_{BS} would result in increasing the subthreshold voltage, which in turn will reduce the subthreshold leakage current. It should be noted that CMOS devices in pass transistor multiplexers will never experience a positive V_{BS}. Pass transistors with logic '0' or opposite signal polarity at both terminals will not experience body effect because their V_{BS} would be zero. However, those devices with logic ‘1’ at both their terminals will experience subthreshold leakage current reduction due to body effect because their $|V_{BS}|$ would be maximum (either V_{DD} or $V_{DD} - V_{TH}$).

In nanometer CMOS devices, the DIBL effect causes the threshold voltage to be a function of the drain source voltage. Applying a large drain source voltage to the CMOS device results in decreasing the subthreshold voltage, hence, increasing the subthreshold current. For minimum sized 90nm NMOS devices, V_{TH} can vary by almost 25% and leakage current by 4.5X due to a difference in V_{DS} equal to the supply voltage.

Pass transistor multiplexers used in FPGAs can experience four different values of V_{DS}. The transistors in the first and last stages of the multiplexer are the only ones that can experience the worst case V_{DS} of V_{DD}. The middle stages can experience a maximum of $V_{DD} - V_{TH}$ because of the weak ‘1’ passed by the NMOS pass transistors. Figure 4.5 shows the four different values of V_{DS} that the pass transistors can experience in FPGAs and the impact on leakage current. Since the signal probability is an indication of the probability that a certain signal is high, then a more accurate leakage power model needs to take into account the different signal states. This fact was used by [96] to develop the first FPGA leakage power model that considers state dependency, however, spatial independence was assumed.

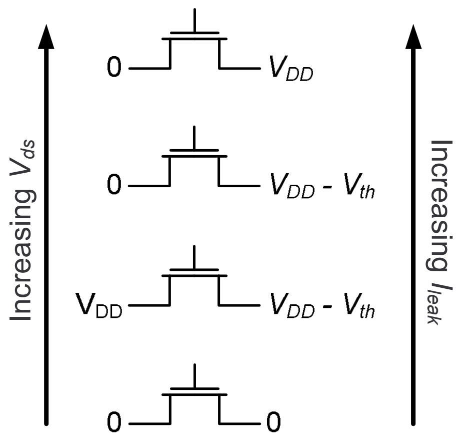


Figure 4.5: DIBL impact on subthreshold leakage in FPGA pass transistor devices.

In this work, the pass transistor multiplexer in Figure 4.4 is simulated using HSpice using all the possible input combinations and the resulting leakage power dissipation is recorded in each case. The leakage values are recorded in a lookup table and used in the power modeling technique. The total leakage power dissipation in any multiplexer in the design is the sum of the leakage power dissipation for a certain input combination (P_{leak_i}) multiplied by the probability of occurrence of this combination (P_i),

$$P_{leak} = \sum_{i=0}^{l} P_{leak_i} \times P_i \ , \tag{4.20}$$

where l is the total number of input combinations. Using Eq. (4.20), the proposed power model will take into consideration the state-dependency of subthreshold leakage power under spatial correlation if the probabilities are computed under spatial correlation using the algorithm presented in Section 4.6.

4.8.3 Gate Leakage Power Dissipation

Under the predictions of [1], the contribution of gate leakage is expected to increase significantly when compared to subthreshold leakage power in future technology nodes. Unlike subthreshold leakage, gate leakage is available in both the ON and OFF states of the CMOS devices. The value of gate leakage is again a strong function of both V_{GS} and V_{DS}. Large values of V_{GS} and small values of V_{DS} result in a larger gate leakage current. Hence, an accurate power model for future FPGAs should consider the state dependency, including spatial correlation, of gate leakage power dissipation.

In this work, the values of the gate leakage of all the basic circuit elements that are used in FPGAs are evaluated using HSpice simulations. The values of the gate leakage current under all the input combinations are recorded in a lookup table and used to evaluate the gate leakage power in a similar manner to Eq. (4.20).

4.9 Results and Discussions

The proposed power estimation methodology under spatial correlation is implemented and integrated into the VPR CAD tool [28]. In order to evaluate the performance of the proposed algorithm for signal probabilities estimation under spatial correlation, several experiments were performed. Firstly, in order to test the accuracy of the algorithm in evaluating the signal probabilities for the different signals in the design, several FPGA benchmarks are simulated using a logic simulator under the zero-delay assumption. A pseudo-random input vector is applied to the inputs of each benchmark and the signal probabilities of the circuit internal nodes are recorded. The length of the input vector used is 10^5, which is proven to result in small inaccuracies [97]. In order to quantify the accuracy of the proposed algorithm, the following metrics are used. The *average relative error* of the signal switching activity is used as a metric of the algorithm accuracy in estimating the switching activity [110]

$$e = \frac{1}{\#\ \text{signals}} \sum_{\text{signal } i} \left| \frac{\alpha_{i,\text{alg}} - \alpha_{i,\text{sim}}}{\alpha_{i,\text{sim}}} \right| , \tag{4.21}$$

where $\alpha_{i,\text{alg}}$ and $\alpha_{i,\text{sim}}$ are the switching activities estimated by the proposed algorithm and the input vector simulation method, respectively. We also define the maximum and minimum relative error of the signal switching activity as

$$e_{\text{min}} = \min_{\text{signal } i} \left| \frac{\alpha_{i,\text{alg}} - \alpha_{i,\text{sim}}}{\alpha_{i,\text{sim}}} \right| , \tag{4.22}$$

$$e_{\text{max}} = \max_{\text{signal } i} \left| \frac{\alpha_{i,\text{alg}} - \alpha_{i,\text{sim}}}{\alpha_{i,\text{sim}}} \right| \tag{4.23}$$

The switching activities evaluated from the input simulations are then compared to those evaluated from the proposed algorithm and [92]. The resulting relative errors are reported in Table 4.1. The benchmarks marked with a grey background are those with a combinational section while the others are datapath circuits. It can be noticed from Table 4.1 that the proposed algorithm manages to capture the correlation between the internal signals of the design even though only 5 cycles were included in the switching activity estimation. The average e_{max} resulting from the proposed algorithm is almost 4X smaller than the average e evaluated from [92].

The averages of the relative errors according to the circuit type are listed in Table 4.2. It can be noticed that the average error for datapath circuits is much less than that for non-datapath circuits. This observation agrees with [110]. Moreover, the big gap in the signal probabilities estimation accuracy between datapath and non-datapath circuits is much less in the proposed algorithm than [92].

In another experiment to evaluate the optimum number of cycles to be considered by the algorithm, the same experiment above was repeated for different number of maximum cycles and the results are plotted in Figure 4.6. It can seen that the accuracy of the algorithm does not improve a lot after the 5 cycles limit. This is mainly because the rejected cycles are those with very small probabilities, which have insignificant effect on the final probabilities. It should be noted that the error does not converge to zero with increasing the number of cycles because of the error in power estimation in sequential circuits that is inherited from the methodology adopted for sequential power calculations.

Table 4.1: Relative error in the switching activity from the proposed algorithm when compared to [92].

Benchmark	Relative error [92]	Relative error (%) (this work)		
	e	$e_{\min}$	e	$e_{\max}$
alu4	28.94	3.58	4.78	6.88
apex2	38.05	3.92	5.78	8.17
apex4	29.98	3.66	5.02	7.67
bigkey	43.62	6.87	8.91	13.20
clma	40.53	4.68	6.06	10.30
des	38.02	2.97	4.19	6.92
diffeq	43.89	4.75	6.08	8.93
dsip	43.18	4.58	6.77	10.16
elliptic	42.49	4.05	6.00	9.30
ex1010	38.12	4.61	5.90	9.86
ex5p	33.94	3.73	4.84	8.07
frisc	42.93	6.55	8.67	12.52
misex3	23.33	3.99	5.42	8.91
pdc	38.58	3.27	5.09	7.95
s298	43.15	5.08	6.39	10.23
s38417	47.78	6.40	8.22	13.26
s38584.1	49.31	4.40	6.25	9.36
seq	36.1	3.82	5.07	7.58
spla	37.95	4.31	5.55	8.90
tseng	48.65	4.73	7.48	11.95
Average	39.427	4.498	6.12	9.51

Table 4.2: Relative error in the switching activity from the proposed algorithm when compared to [92].

	e [92]	Relative error (this work)		
		$e_{\min}$	e	$e_{\max}$
Datapath Circuits	34.30	3.79	5.16	8.09
Mixed Circuits	44.55	5.21	7.08	10.92

In order to study the accuracy of the proposed power model in estimating the total FPGA power, the algorithm is applied to 4 small FPGA circuits. A brief description of the 4 test circuits is presented in Table 4.3. Moreover, circuits 1 and 2 are datapath circuits while 3 and 4 are mixed circuits containing sequential logic and feedback loops. Moreover, the circuits are selected to feature almost no glitches to remove their effect on the results accuracy. The architecture of the

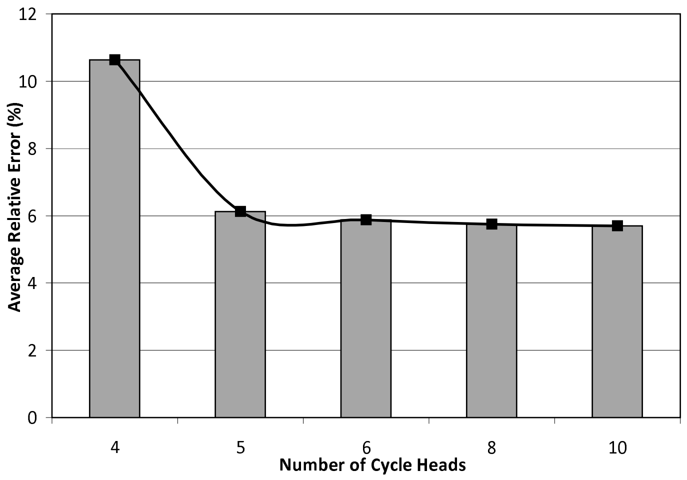


Figure 4.6: Average relative error in estimating the signal probabilities under spatial correlation by varying the number of cycles considered.

target FPGA employed has a 4-input LUT and each logic cluster contains 4 LUTs. The different signal probabilities of all the signals in the test circuits are evaluated with and without considering spatial correlation. Afterwards, the benchmarks are designed and simulated using HSpice using a random function generator to generate the circuit inputs. The length of the stream generated by the function generator is varied from 10 to 10,000.

Table 4.3: Small benchmark circuits.

Benchmark	**# of logic blocks**	**# of inputs**	**# of cycles**
circuit1	7	3	2
circuit2	10	5	3
circuit3	10	4	6
circuit4	14	3	8

Figures 4.7 and 4.8 plot the average error in the signal probabilities for the case when spatial correlation is considered and when spatial independence is assumed, respectively, against the length of the input vector. The percentage error is calculated between the average signal probability calculated from the HSpice simulation and the estimated ones. In Figure 4.7, it is noticed that the percentage error goes below 1% for an input vector of length 100. It should be noted that 'circuit2' initially has the largest error because it has a large number of inputs, 5, which are not fully covered by the 10 input combinations. On the other hand, 'circuit4' has the least number of inputs, 3, hence, has the least percentage error for an input vector of length 10.

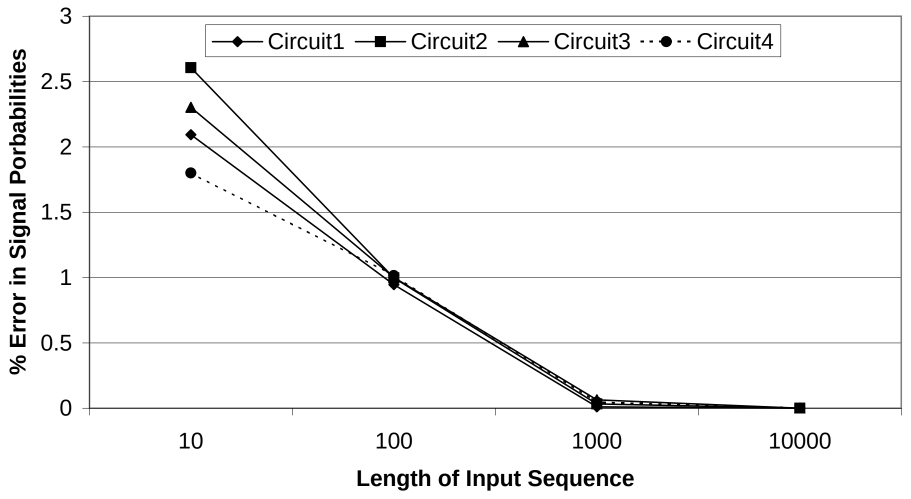


Figure 4.7: Percentage error in estimating the signal probabilities under spatial correlation when compared to HSpice versus the length of the input sequence.

Figure 4.8 plots the average percentage error in estimating the signal probabilities under the spatial independence assumption. It can be noticed that the graphs for 'circuit1' and 'circuit4' saturate very quickly, mainly because they have the smallest number of inputs, hence, they reach their final probabilities using a small number of inputs. 'circuit2' has the maximum number of inputs that are not probably covered by a vector length of 10, thus it has the maximum percentage error for that input vector length. It should be noted that the final values of the percentage error for each circuit is due to spatial correlation. An interesting point in Figure 4.8 is that both 'circuit2' and 'circuit3' have the same number of logic blocks, yet the average error in estimating the signal probabilities in 'circuit3' is higher than that of 'circuit2'. This is because 'circuit3' has more cycles than 'circuit2' as well as having sequential feedback paths, hence, the error due to the spatial independence assumption is magnified.

In the next set of experiments, the same four circuits are simulated using HSpice using a CMOS 90nm technology and their total power dissipation values are recorded. Similarly, the power dissipation in these four circuits was evaluated using the proposed power model. The total power dissipation is calculated twice, using the same equations, using transition density values computed with and without spatial correlations and the results are plotted in Figure 4.9. The maximum error between HSpice power calculation and that evaluated using spatial correlation is 8.8%. On the other hand, the error between the power recorded by HSpice and that calculated while assuming spatial independence is 24.2%.

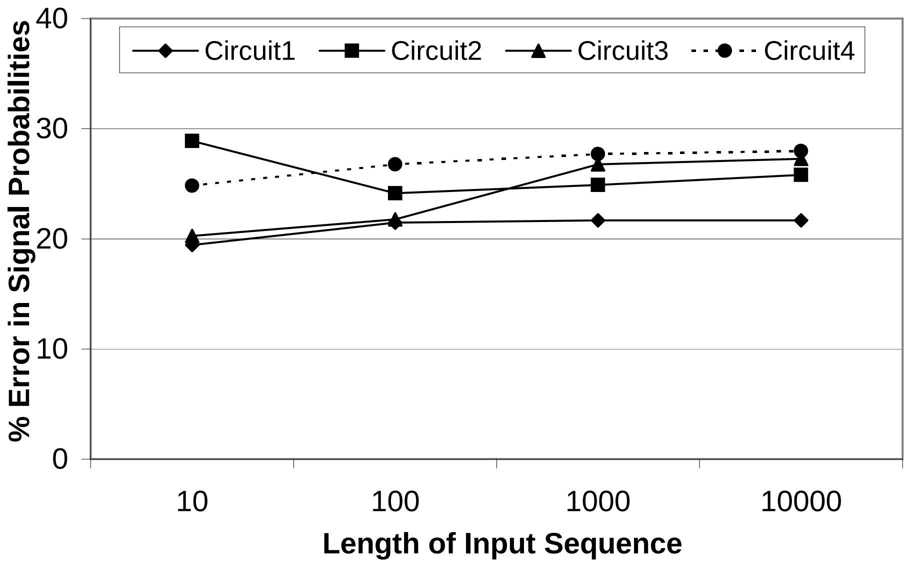


Figure 4.8: Percentage error in estimating the signal probabilities under spatial independence when compared to HSpice versus the length of the input sequence.

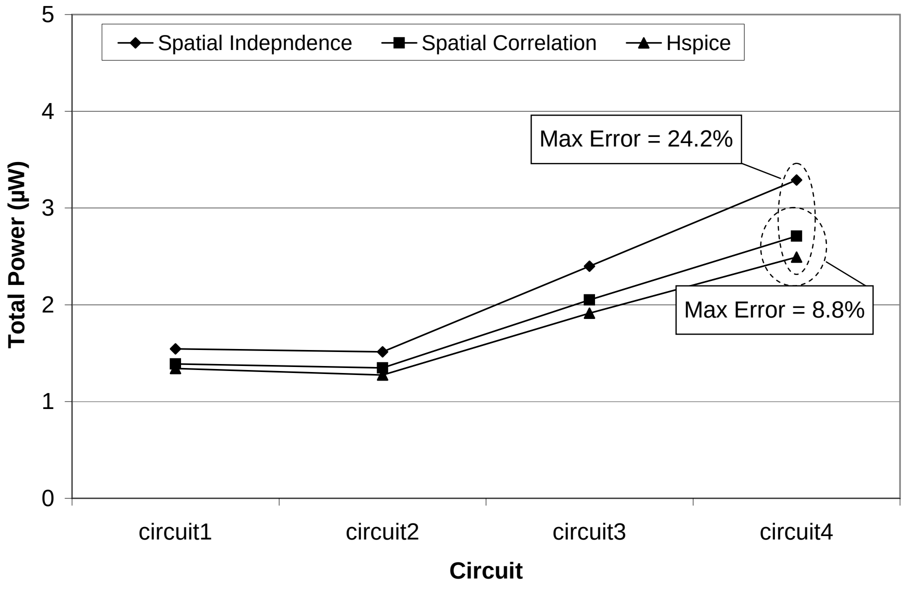


Figure 4.9: Percentage error between power estimated with and without spatial correlation when compared to HSpice.

In the next set of experiments, the proposed power model is used to calculate the power dissipation under spatial correlation in several FPGA benchmarks. Moreover, the same power model is used to calculate power dissipation in the same benchmarks while assuming spatial independence between the different design signals. The experiments were run on a quad Xeon processor machine running at 3.4GHz with a total of 16GB RAM. Table 4.4 lists the percentage difference between the power evaluated when considering spatial correlations and while assuming spatial independence. It should be noted that the times reported in 4.4 correspond to the proposed power modeling technique with spatial correlation. Moreover, Table 4.4 lists the number of cycles found in each benchmark, which suggests that cycles are frequent in VLSI circuits, hence, spatial correlation among the different signals in a design is common. An interesting point in Table 4.4 is that the runtime is almost proportional to the number of cycles, except for two benchmarks, 'clma' and 's38584.1'. This is mainly because these two are the largest benchmarks and it takes long time to process them, even without considering spatial correlation. For small benchmarks, regular probability propagation consumes considerable runtime which covers for the increase in runtime to account for signal correlations. However, for bigger benchmarks, the runtime gets dominated by the correlation processing algorithm.

Table 4.4: Percentage change in power estimation under spatial correlation when compared to spatial independence.

Benchmark	**# of Logic Blocks**	**# of Cycles**	**Run Time(s)**	**% Change in Dynamic Power**	**% Change in Leakage Power**	**% Change in Total Power**
alu4	1522.00	606.00	9.00	-20.54	-26.39	-21.45
apex2	1878.00	623.00	13.00	-24.63	8.80	-20.44
apex4	1262.00	600.00	6.00	-26.54	9.83	-20.04
bigkey	1707.00	452.00	12.00	-25.82	4.84	-20.52
clma	8381.00	2343.00	614.00	-20.04	-15.74	-17.44
des	1591.00	715.00	9.00	-25.17	8.29	-19.51
diffeq	1494.00	304.00	12.00	-18.95	22.55	-9.38
dsip	1370.00	454.00	9.00	-22.97	15.19	-17.91
elliptic	3602.00	722.00	100.00	-27.29	14.24	-18.86
ex1010	4598.00	3257.00	216.00	-37.12	33.96	-24.39
ex5p	1064.00	721.00	6.00	-22.61	-19.43	-20.57
frisc	3539.00	1417.00	128.00	-26.24	21.31	-13.17
misex3	1397.00	615.00	7.00	-23.27	-26.12	-25.35
pdc	4575.00	3882.00	264.00	-18.61	-13.80	-10.15
s298	1930.00	609.00	18.00	-22.20	-11.43	-17.09
s38417	4096.00	117.00	195.00	-25.07	32.83	-16.20
s38584.1	6281.00	1396.00	281.00	-28.12	13.42	-16.90
seq	1750.00	641.00	11.00	-22.54	-33.87	-25.28
spla	3690.00	3082.00	131.00	-18.12	-30.53	-20.01
tseng	1046.00	234.00	8.00	-29.48	-16.65	-24.64

By examining the results in Table 4.4, it can be noticed that after considering spatial correlation, the dynamic power estimated for all the designs decreased because of the over-estimation nature of the spatial independence assumption. On the other hand, there is no clear trend on the impact of spatial correlation among the design signals on leakage power dissipation. This is because considering spatial correlation will only change the probabilities of some of the leakage states. The state that experiences a change in its probability might be the one with the highest or lowest leakage current. Hence, there is no limitation on the change in the leakage power estimation due to spatial correlation. The average change in the total power dissipation is almost 19%.

As the CMOS process is scaled down, the contribution of leakage power dissipation to the total power dissipation is expected to increase notably until it surpasses the dynamic power by the 65nm process [1]. In order to evaluate the scalability of the proposed power modeling technique with the increasing contribution of leakage power, the proposed power model is applied to all of the FPGA benchmarks in Table 4.4 using several CMOS processes (90nm, 65nm, and 45nm). The percentage change in power estimation between the spatial correlation and independence assumptions are recorded in each case, and the average change per technology is calculated.

Figure 4.10 plots the average difference between the dynamic, leakage, and total power dissipation with and without spatial correlation. It can be deduced that the average difference in the total power dissipation estimation is almost the same for the CMOS 90nm, 65nm, and 45nm technologies. Although, the percentage of dynamic power dissipation decreases across technologies, the impact of spatial correlation on the total power dissipation still remains the same. This is because the impact of spatial correlation on leakage power dissipation stays almost the same across technologies, while the contribution of leakage power increases with the technology scaling, thus compensating for the decrease in dynamic power dissipation.

This conclusion is verified by Figures 4.11 and 4.12 which plot the similar changes broken down for leakage and dynamic power dissipation, respectively. The dependence of leakage power dissipation on the spatial correlation increases as the technology scales down, thus compensating for the decrease in dependence in dynamic power dissipation.

In another experiment, the cluster size is varied between 4BLEs, 6BLEs, and 8BLEs and the results are plotted in Figure 4.13 for a 90nm CMOS process. From Figure 4.13, it can be deduced that as the cluster size increases, the impact of considering the spatial correlation on the power dissipation decreases. This observation can be justified by the fact that increasing the cluster size, decreases the wire-length of the whole circuit and hence, the total capacitance value in (4.1), hence, the total

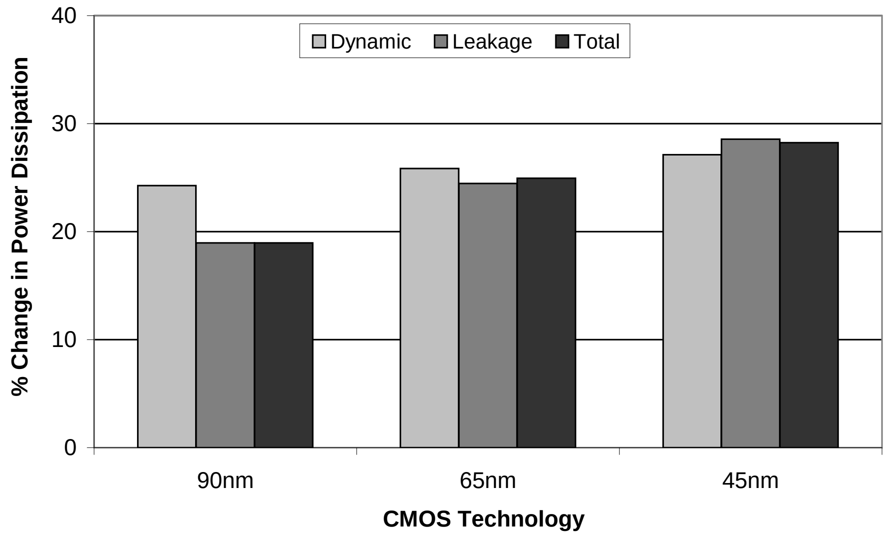


Figure 4.10: Average percentage change in power dissipation to account for spatial correlation for different technology nodes.

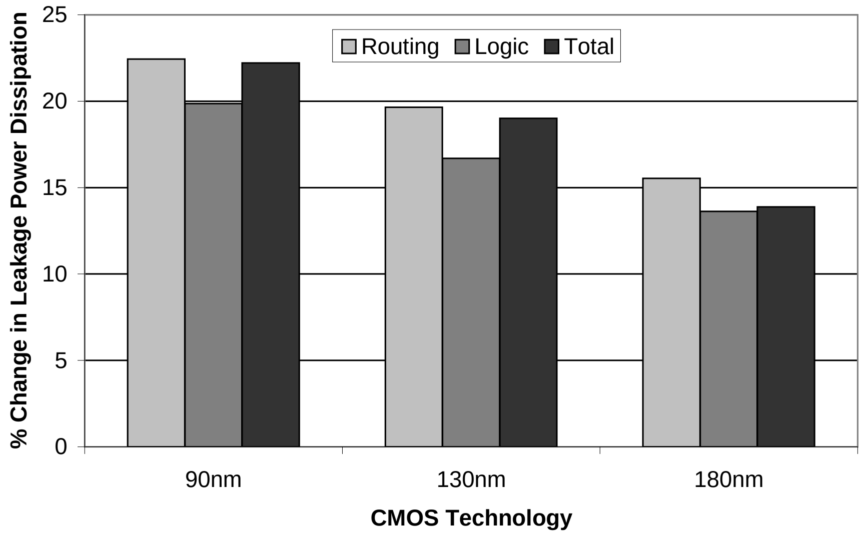


Figure 4.11: Average percentage change in leakage power dissipation with and without spatial correlation for different technology node.

circuit dynamic power dissipation decreases. As a result, the dependency of dynamic power of the transition density will decrease as well. Moreover, increasing the cluster size, increases the levels of pass-transistors levels for the multiplexer in Figure 4.4, thus resulting in a decrease in the subthreshold leakage power dissipation of the multiplexers. Hence, the total leakage power dissipation of the circuit decreases significantly, resulting in a decrease in the dependency of leakage power on the signal probabilities.

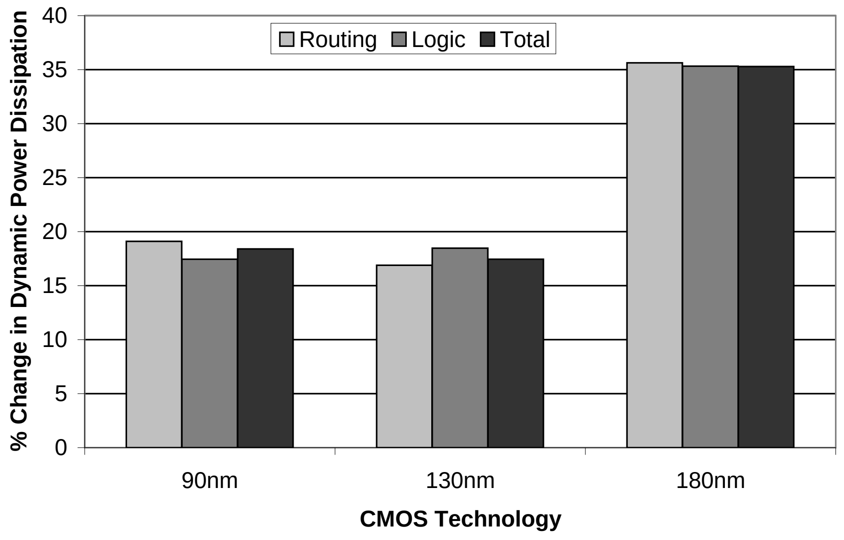


Figure 4.12: Average percentage change in dynamic power dissipation with and without spatial correlation for different technology node.

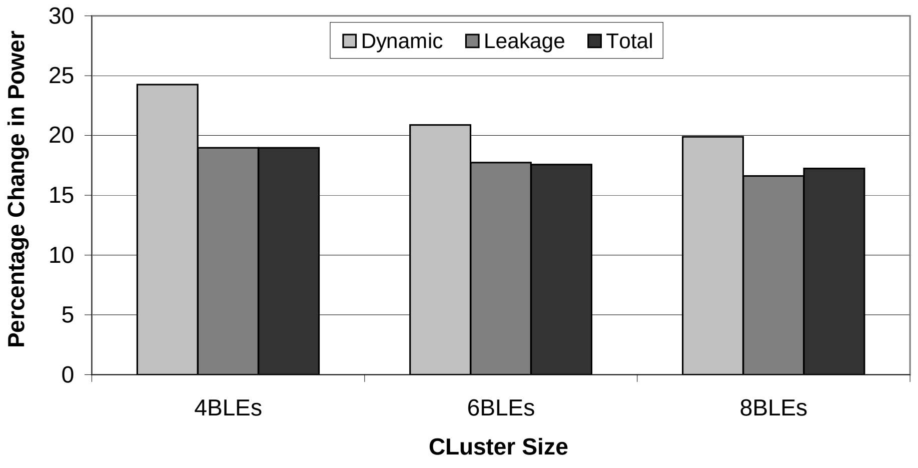


Figure 4.13: Percentage change in power dissipation between spatial correlation and independence versus the cluster size.

4.10 Conclusion

This Section described a methodology for calculating the signal probabilities in FPGAs under spatial correlation. The signal probabilities under spatial correlation are integrated into a novel power model for FPGAs to model both the dynamic and leakage power dissipation in FPGAs. The accuracy of the model is within 10% of HSpice simulations while spatial independence approaches are within 25% of HSpice simulations.

Chapter 5

Leakage Power Reduction in FPGAs Using MTCMOS Techniques

This Chapter proposes supply gating techniques in FPGAs through the use of Multi-Threshold CMOS (MTCMOS) approaches for subthreshold leakage power reduction. A modified FPGA architecture with sleep transistors is proposed and the CAD algorithms needed to benefit from the architecture changes are developed. Specifically, a new activity profiling phase is introduced in the CAD flow to identify the blocks that exhibit similar idleness to collectively turn them OFF during their idle times. Moreover, new packing techniques are developed to pack those blocks with similar activity profiles together to easily turn them OFF.

This Chapter is organized as follows: an introduction to MTCMOS in general is given in Section 5.1. The MTCMOS FPGA architecture is proposed in Section 5.2. Section 5.3 discusses the problem of sizing the sleep transistor in the MTCMOS architecture. The methodology developed to identify the logic blocks that exhibit similar idleness periods is presented in Section 5.4. Section 5.5 proposed the packing algorithms used to benefit from the activities generated by the activity profiling stage. Finally, the results are discussed in Section 5.7.

5.1 Introduction

In FPGA designs, leakage power reduction has been overshadowed by performance improvements and dynamic power minimization techniques. However, recently, leakage power started to gain increased attention by both FPGA circuits and CAD designers. The leakage power dissipation problem is more crucial in FPGAs compared to custom ASIC designs because of the unutilized resources in FPGAs. On average, the percentage utilization of resources in FPGAs is around 60% [55], thus, almost 40% of the FPGA consumes standby leakage power without delivering useful output. Moreover, FPGAs employed in wireless applications can go into idle mode for long periods of time [111]. In such designs, even the utilized resources need to be forced into a low-power (standby) mode during their idle periods to save leakage power.

One of the most popular techniques used in leakage power reduction in ASIC designs is Multi-threshold CMOS (MTCMOS) [112, 113]. In an MTCMOS implementation, a high V_{TH} (HVT) device called the *Sleep Transistor*, connects the pull-down network employing low V_{TH} (LVT) devices of a circuit to the ground, as shown in Figure 5.1(a). When the sleep transistor is turned OFF, the circuit subthreshold leakage current is limited to that of the sleep transistor which is significantly low. Hence, the circuit benefits from the high-performance of the LVT pull-down network when the sleep transistor is turned ON, while limiting the circuit subthreshold leakage current when the sleep transistor is turned OFF.

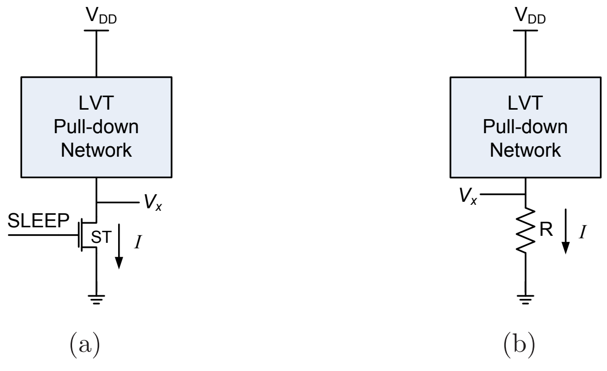


Figure 5.1: MTCMOS architecture. (a) General MTCMOS architecture, (b) Equivalent ST circuit in the active mode.

The sleep transistor acts as a small finite resistance R to the ground when the *SLEEP* signal is *high* with a finite small voltage at the virtual ground rail V_x, as shown in Figure 5.1(b). However, the sleep transistor resistance R incurs a performance penalty because the driving potential of the circuit is reduced to $V_{DD} - V_x$ [111,114]. When the *SLEEP* signal is *low*, the circuit goes into a standby mode with the voltage at V_x rising to a voltage between 0 and V_{DD}, with the sleep transistor acting as a very high resistance, thus reducing the standby subthreshold leakage current considerably.

In FPGAs, sleep transistors can reduce subthreshold leakage by: (i) *permanently* powering-down the unutilized parts of the chip per configuration, (ii) *dynamically* turning ON and OFF the utilized parts of the chip depending on their activity, and (iii) powering down all of (or a large part of) the FPGA during the design idle time.

In this Chapter, the MTCMOS technique is employed in FPGA design and the changes needed at the CAD level are developed to take full advantage of the technique in maximizing the leakage savings. These changes are integrated into the academic Versatile Place and Route (VPR) flow [27]. A flowchart of a typical VPR CAD flow is shown in Figure 5.2(a) and a flowchart of the proposed modifications is shown in Figure 5.2(b). In Figure 5.2(b), a new stage is added to the CAD flow, the activity generation phase, in which the design is analyzed to identify the logic blocks that exhibit similar *activity profiles*. Blocks with similar activity profiles are forced into a standby mode together. The activity profiles generated by the activity generation algorithms are then integrated into the T-VPack algorithm to result in the activity T-VPack algorithm (AT-VPack), as shown in Figure 5.2(b). A modified power model, that takes into consideration the proposed changes in the FPGA architecture is used to properly calculate the power savings from the proposed MTCMOS FPGA architecture.

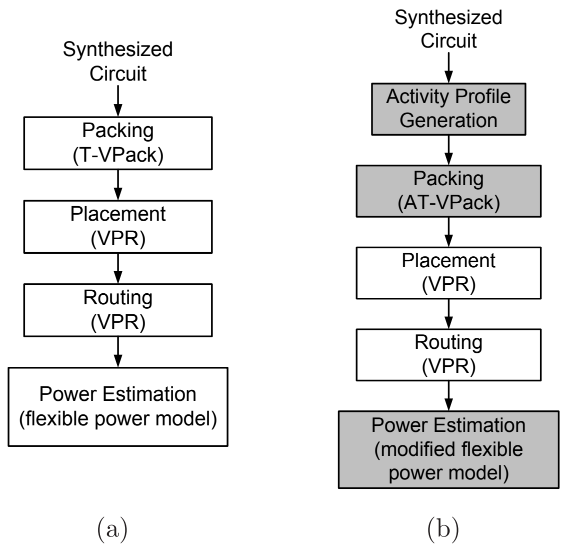


Figure 5.2: FPGA CAD flowchart. (a) Conventional VPR flowchart. (b) Proposed CAD flowchart integrated in the VPR flow.

5.2 MTCMOS FPGA Architecture

The conventional hierarchial FPGA architecture, adopted by most modern FPGAs, utilizes logic blocks, which are conventionally made of a 4-input Look-Up Table (LUT), a flip-flop, and a 2:1 multiplexer, as shown in Figure 5.3. Several logic blocks are further grouped together to form a cluster of logic blocks. Inside each cluster, the logic blocks are connected using the local routing resources, while the clusters are connected using the global routing resources.

The MTCMOS FPGA architecture proposed in this thesis follows the broad guidelines of the hierarchial architecture, however, every N clusters are connected to the ground through one sleep transistor, as shown in Figure 5.3. Moreover, the latches in each cluster are used to retain the value of the logic blocks outputs when they enter the sleep mode, thus they are not connected to the sleep transistors. The logic blocks served by one sleep transistor are called the *sleep region*. It should be noted that the sleep transistors are not confined to the logic resources of the FPGA, but applied to the routing resources of the fabric as well.

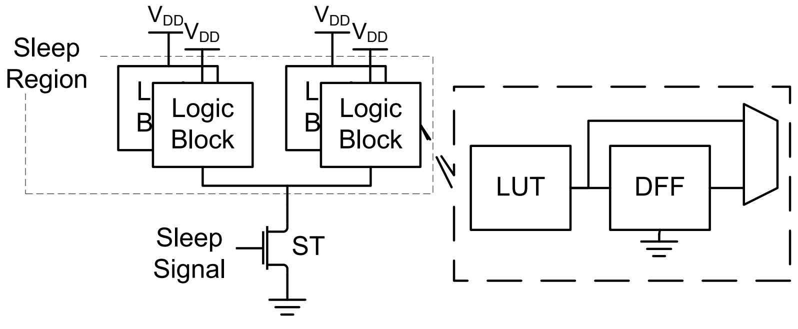


Figure 5.3: MTCMOS FPGA architecture. The logic blocks connected to one sleep transistor are called sleep region.

Each sleep transistor is controlled by a *SLEEP* signal, deactivating the *SLEEP* signal forces the N clusters in the corresponding sleep region into low-power mode during their inactive periods. Before entering the sleep mode, the output of each logic block is stored in the latch so it can be recovered when the sleep region wakes up again. The *SLEEP* signals of the unutilized, whether logic or routing, resources of the FPGA are kept deactivated at all times to turn them permanently OFF.

The *SLEEP* signals are generated dynamically during the device runtime using the partial reconfiguration logic available in modern FPGAs [115, 116], thus providing minimum area overhead. The *SLEEP* signals can be generated if the application of the design is well-known in advance. For example, if the design is used to implement an MPEG decoder, then the sequence of operations to be executed is known in advance as well as the statistics of each signal, which can then be used to generate the *SLEEP* signals as will be explained later in Section 5.4. This is a very interesting point since the majority of the FPGA applications are indeed dedicated ones where the application is well-known in advance. However, if the design application is a general one, earlier works formulated a methodology for predicting the statistics of the design signals in a methodology similar to branch prediction methodologies [117].

The number of clusters that can fit in one sleep region is determined by: (1) the size of the sleep transistor, which in turn corresponds to the maximum performance loss allowed, (2) leakage power savings, (3) area overhead permitted in the design due to sleep transistors, and (4) the maximum permitted ground bounce on the virtual ground lines. For the same performance penalty, large sleep regions employ larger, but fewer in number, sleep transistors. As a result, the control circuitry needed to generate the *SLEEP* signals is typically less complex, consumes less power, and occupies smaller area when compared to the small sleep regions case. However, large sleep regions have limited leakage power savings capability due to the use of large sleep transistors, which sink larger subthreshold leakage current. Moreover, large sleep regions suffer from a smaller selectivity in turning OFF idle clusters, thus reducing their resulting leakage power savings. Hence, the optimum granularity is set based on a compromise between the area overhead and the required leakage power savings. In [118], the authors concluded that the optimum granularity ranges between 4 to 8 logic blocks.

A diagram of the proposed FPGA fabric is shown in Figure 5.4, where the sleep transistors are prefabricated with a fixed size in the FPGA fabric. It was shown in [119] that such a placement provides the minimum area overhead while ensuring full connectivity between the sleep transistors and the logic blocks. Moreover, the *SLEEP* control signals for each sleep transistor are hardwired during the FPGA fabrication. The virtual ground $VGND$ line is used to connect the pull down networks of the logic blocks to the sleep transistor, as shown in Figure 5.4. The $VGND$ lines are hardwired to their corresponding sleep transistors. Several research works proposed optimum layouts for the sleep transistors to provide the minimum area overhead [120], and the average area overhead of MTCMOS architectures with fine granularity (between 4 to 8 logic blocks) in FPGAs is reported to be around 5% [121, 122].

It should be noted that there are two approaches for sleep transistor implementations: header or footer devices. Header devices use a PMOS sleep transistor to

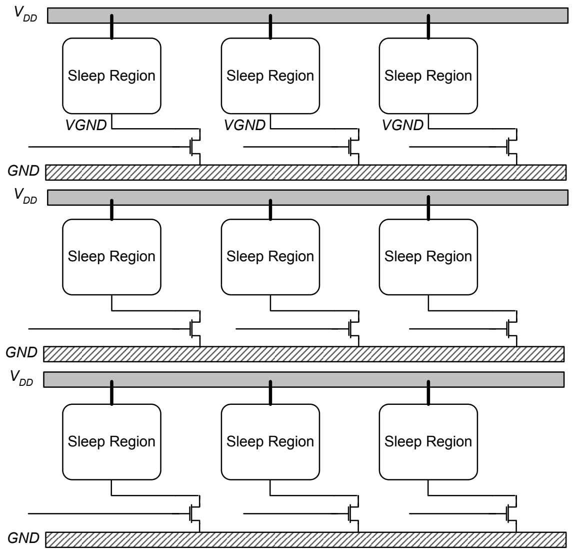


Figure 5.4: MTCMOS-based FPGA fabric with sleep transistors.

block the path from the supply rail, while the footer approach uses a NMOS to block the path to the ground, as shown in Figure 5.5. The PMOS header approach has the disadvantage if incurring a large area penalty when compared to the NMOS footer approach. This is mainly because of the lower drive current of PMOS devices due to the lower mobility of holes when compared to electrons. As a result, to have the same performance penalty due to sleep transistors, PMOS headers with larger areas are needed. Consequently, in this work, only footer NMOS sleep transistors are used.

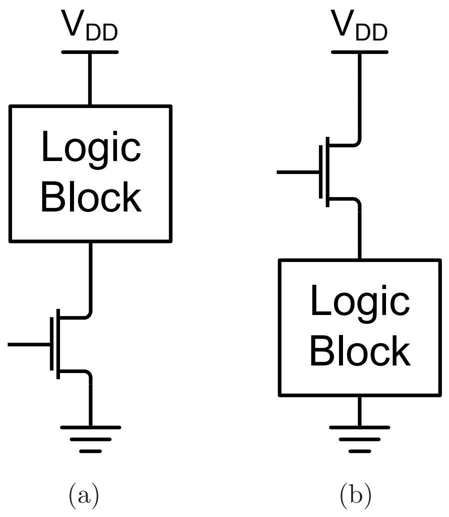


Figure 5.5: Sleep transistor implementations. (a) NMOS footer. (b) PMOS header.

Typically, there are two sleep transistors architectures; local and global sleep transistors. Local, or distributed, sleep transistors are placed at the local block level, where the local block is defined as a part of the circuit that can be independently idle. On the other hand, a global sleep transistor architecture employs a single sleep transistor for a large circuit block that includes several local blocks. In this work, a local sleep transistor architecture is adopted for the following reasons: (1) the $VGND$ lines are short enough to be treated as local connections, hence, there is no need to fabricate them using wide metal lines like V_{DD} and GND rails, (2) the routing complexity of the $VGND$ lines is significantly easy in local sleep transistor architectures when compared to the global architecture, and (3) local sleep transistors provide less routing overhead in terms of the criticality of the sleep signals, better noise margins, and higher turn OFF flexibility, thus higher power savings. However, the control systems for local sleep transistors architecture are more complicated.

5.3 Sleep Transistor Design and Discharge Current Processing

In this Section, several issues related to the sleep region are discussed. Firstly, the design problem of the sleep transistor is introduced and a formulation for the transistor size is presented in terms of the total discharge current of the sleep region. Secondly, two methods for total sleep region discharge current calculations are proposed, the first one is a modified version of the mutually exclusive discharge current algorithm proposed in [114]. The second method is a newly proposed algorithm that considers the logic function implemented by the logic blocks.

5.3.1 Sleep Transistor Sizing

The proper sizing of the sleep transistor is crucial to achieve the maximum subthreshold leakage power savings without incurring large performance and area penalties, as explained in Section 5.2. While the delay penalty is inversely proportional to the width of the sleep transistor, a large sleep transistor results in a large subthreshold leakage current and higher parasitic capacitances, which results in high dynamic power dissipation during the switching of the sleep transistor. Moreover, a large sleep transistor consumes a larger part of the total chip area. The first step in sizing the sleep transistor is to formulate the delay penalty experienced by the FPGA circuitry due to the sleep transistors.

The delay of a CMOS gate without any sleep transistors t_d is expressed as [112, 123]

$$t_d \propto \frac{C_L V_{DD}}{(V_{DD} - V_{TH_l})^\alpha} , \tag{5.1}$$

where C_L is the gate load capacitance, V_{TH_l} is the threshold voltage of the circuit low V_{TH} transistors (LVT), and α is the velocity saturation index. The delay of the same gate in the presence of a sleep transistor $t_{d,\text{sleep}}$ is expressed as [114]

$$t_{d,\text{sleep}} \propto \frac{C_L V_{DD}}{(V_{DD} - V_x - V_{TH_l})^\alpha} , \tag{5.2}$$

where V_x is virtual ground rail voltage, as shown in Figure 5.1(a).

In order to balance between the performance penalty and power savings, the maximum allowable performance loss should be limited to a predefined value. Let the ratio between t_d and $t_{d,\text{sleep}}$ given by

$$\frac{t_{d,\text{sleep}} - t_d}{t_{d,\text{sleep}}} = x , \tag{5.3}$$

where x is the performance loss due to sleep transistors. For simplicity, assume that α can be approximated to be equal to 1 [114]. Therefore, substituting with Eq. (5.1) and (5.2) into (5.3), yields

$$V_x = x \times (V_{DD} - V_{TH_l}) . \tag{5.4}$$

When the sleep transistor is turned ON, it will operate in the linear mode of operation, as explained earlier. Using the square law for the MOS device current, the drain to source current flowing through the sleep transistor I_{sleep}, *i.e.*, discharge current, can be approximated by

$$I_{\text{sleep}} = \mu_n C_{ox} \left(\frac{W}{L}\right)_{\text{sleep}} \left[(V_{DD} - V_{TH_h}) \times V_x - \frac{V_x^2}{2}\right] , \tag{5.5}$$

where μ_n is the device mobility, C_{ox} is the oxide thickness, W and L are the device width and length, respectively, and V_{TH_h} is the threshold voltage of the sleep transistor, which is a HVT device. Substituting with the expression of V_x given in Eq. (5.4) into the value of I_{sleep} in Eq. (5.5) and rearranging the relationship results in [114]

$$\left.\frac{W}{L}\right|_{\text{sleep}} = \frac{I_{\text{sleep}}}{x \mu_n C_{ox} (V_{DD} - V_{TH_l})(V_{DD} - V_{TH_h})} . \tag{5.6}$$

The parameters in Eq. (5.6) are all technology parameters except for the speed penalty x and the maximum discharge current allowed through the sleep transistor

I_{sleep}. In most previous works, x has been set to a constant value, usually 5% [114]. As a result, all the circuit paths will experience a fixed speed degradation. However, in this thesis, two possibilities are explored, setting x to a fixed value as well as using variable speed penalties to improve the final design performance, as will be explained in Section 5.5.

The next step in finding the proper size of the sleep transistor is to compute the sleep region maximum discharge current I_{sleep}. It should be noted that this work employs footer devices, *i.e.*, NMOS devices, as sleep transistors. Hence, the main criterion that controls the sizing of footer devices is the discharge current of the pull-down network. The charging current flows through the pull-up circuit and the sleep transistor is not involved in this process, hence, the charging time is not affected.

The worst-case maximum value for I_{sleep} is the sum of discharge currents of all the logic blocks inside the sleep region. Since all the logic blocks in FPGAs are identical, then the values of their discharge currents would be equal. As a result, the value of I_{sleep} would be expressed as

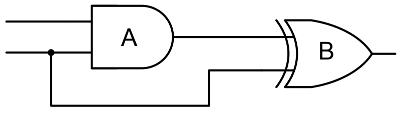
$$I_{\text{sleep}} = I_{\text{discharge}} \times N \; , \tag{5.7}$$

where $I_{\text{discharge}}$ is the discharge current of one logic block and N is the granularity of the sleep region, *i.e.*, the number of logic blocks in one sleep region.

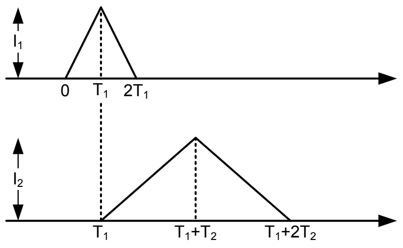
However, this is more of an upper bound on the value of I_{sleep} due to two factors: (1) the delays of the logic blocks are finite and (2) not all the logic blocks inside the sleep region will discharge simultaneously. The choice and computation of the discharge current I_{sleep} inside the sleep region is explained in the following subsections. Selecting a value for I_{sleep} depends on the allowable number of logic blocks in each sleep region. In order to find the optimum value of I_{sleep} to be used in this work, the discharge current of each cluster placed using the conventional VPR tool is calculated for several FPGA benchmarks. It was found that the value of the discharge current of all the clusters is usually less than 75% of the worst-case discharge current, which is therefore used in this work. However, it should be noted that the sum of discharge currents of all the logic blocks inside the cluster must not exceed the value of I_{sleep}, or else the sleep region will experience a bigger speed penalty than that used to evaluate $\left(\frac{W}{L}\right)_{\text{sleep}}$ in Eq. (5.6).

5.3.2 Mutually Exclusive Discharge Current Processing

The mutually exclusive discharge current processing technique was first proposed in [114] for standard cells MTCMOS design. This technique makes use of the finite delays of each gate to provide a sequence of discharge current patterns inside each sleep region. The discharge current of any logic gate is represented using a symmetric triangular approximation, as shown in Figure 5.6(b). Due to the finite delay of A in Figure 5.6(a) and the dependence of B on the inputs to A, B will not start discharging before the discharge current of A reaches its peak [114]. In this case, A and B are said to be mutually exclusive in their discharge current, since they are not going to discharge simultaneously.



(a) A circuit example for the timing diagram.



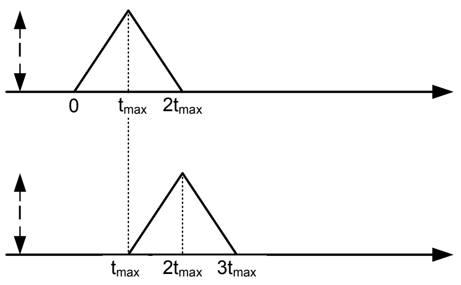
(b) Timing diagram for mutually exclusive logic gates.

Figure 5.6: Mutually exclusive discharge current processing.

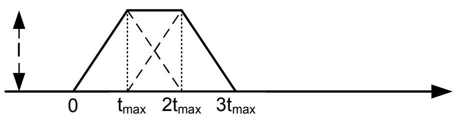
In Figure 5.6(a), two parameters characterize the discharge current of each gate: the maximum value the discharge current can reach I_i and the time it takes for the discharge current to reach its peak T_i, as shown in Figure 5.6(b). The values of these two parameters depend on the type of the gate, since every gate would have a different delay and maximum discharge current, and the fanout, increasing the gate fanout slows down the discharge by decreasing the value of I_i and increasing T_i. Hence, in order to use this technique, all gates in the design library are characterized initially by simulating their discharge currents under all possible loading scenarios using HSpice.

Applying this technique for discharge current processing in FPGAs is much simpler than the standard cells case due to the regularity of FPGAs. Firstly, all FPGA logic blocks are identical, since a k-input logic block can implement any k-input logic function. As a result, only one circuit is characterized using HSpice. Secondly, the loading effect in FPGAs is very uniform due to the use of routing switches. Hence, there is a very limited number of loading scenarios that can be experienced. These two facts decrease the number of HSpice simulations needed to characterize the logic gates significantly. As an approximation, this work assumes that the discharge current patterns, in terms of peak value and duration, are the same for all the logic blocks in the design.

If the small circuit example in Figure 5.6(a) is implemented in an FPGA, the discharge current of these two logic blocks will be represented as shown in Figure 5.7(a). It can be noticed in Figure 5.7(a) that the discharge currents of both A and B are identical to reflect the fact that FPGA logic blocks are identical. These two discharge currents can be summed in a vector manner to result in the total discharge current for these two logic blocks, as shown in Figure 5.7(b). Hence, if these two logic blocks are placed in one sleep region, then the maximum discharge current that this sleep region will ever experience is only equal to $I_{\text{discharge}}$ of one logic block. This proves the worst case value of I_{sleep} given in Eq. (5.7) is not always needed for the logic blocks inside the sleep region.



(a) FPGA timing diagram of the circuit in Figure 5.6(a).



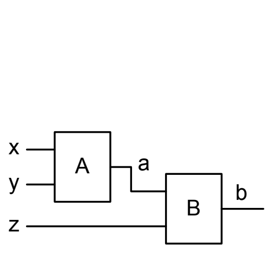
(b) Summation of discharge currents.

Figure 5.7: Mutually exclusive discharge current processing.

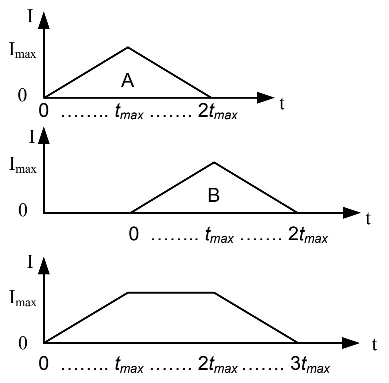
5.3.3 Logic-Based Discharge Current Processing

Earlier MTCMOS works adopted a worst-case discharge current processing algorithms by assuming that whenever a logic block A discharges, all of its outputs will start discharging after the discharge current of A reaches its maximum [114], as shown in Figure 5.8(b). However, The discharge of the fanout logic blocks of A will depend on the logic they implement. Therefore, a more efficient current processing algorithm has to include the probability of the circuit actually discharging based on the logic function implemented by the circuit.

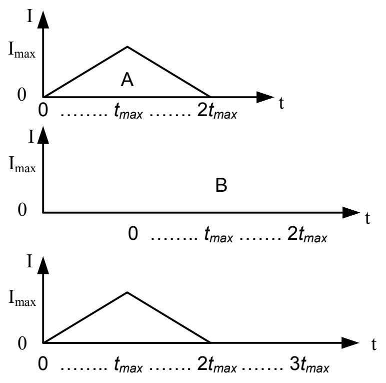
For example, consider the small circuit in Figure 5.8(a), assume that B implements the following logic function: $b = \bar{a} + az$. Hence, whenever the output of A goes low, the output of B will always go high. Consequently, A and B are mutually



(a) Small circuit example.



(b) Logic-based discharge current processing for non-mutually exclusive logic blocks.



(c) Logic-based discharge current processing for mutually exclusive logic blocks.

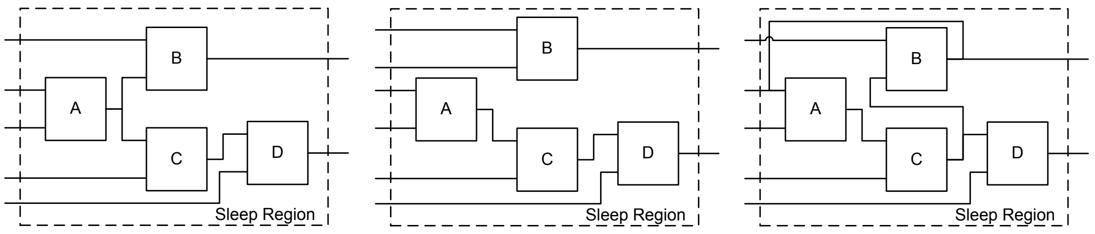
Figure 5.8: Linear vector approximation of discharge current and logic-based current vectors summation.

exclusive in their discharge. As a result, the total discharge current of these two logic blocks would be only equivalent to that of one of them, as shown in Figure 5.8(c). As a result, adding block B to the sleep region that contains A comes at no expense in term of the discharge current of the sleep region, hence, speed penalty. This property will give the packing algorithms, which will be introduced later on in Section 5.5, more flexibility in packing logic blocks in the same sleep region without violating the maximum discharge current constraint of the sleep region. This new technique used in calculating the total discharge current inside a sleep region is called *logic-based discharge current processing.*

5.3.4 Topological Sorting and Discharge Current Addition

In this work, topological sorting is used to properly align the current vectors in the sleep region to find the total discharge current. The topological sorting algorithm encounters three different types of sleep regions: a combinational sleep region where each logic block shares at least one net with any other logic block in the sleep region (Figure 5.9(a)), a combinational sleep region with at least one logic block not sharing any net with any other logic block in the sleep region (Figure 5.9(b)), or a sequential sleep region that contains one or more loops (Figure 5.9(c)).

For a combinational connected sleep region, as shown in Figure 5.9(a), the algorithm starts by converting the logic blocks inside the sleep region into an undirected graph. The graph in Figure 5.10(a) is equivalent to the sleep region in Figure 5.9(a).



(a) Combinational connected. (b) Combinational with unconnected blocks. (c) Sequential with loops.

Figure 5.9: Different types of sleep regions.

Afterwards, a topological sorting of the resulting graph is used to find the relationship between all the logic blocks in the sleep region by converting the graph to a hierarchal data structure. An example of the topological sorting procedure is shown in Figure 5.10, where A is found as the parent node, B and C are ordered in the *same level*, and D in the last level. The linear approximation for the discharge current for the logic blocks in the sleep region is shown in Figure 5.10(e) as well as the resulting sum of the discharge current vectors. It should be noted that the summation in Figure 5.10(e) assumes that the logic blocks will have non-mutual exclusive discharge.

For a combinational graph with unconnected nodes, as shown in Figure 5.11, instead of using the triangular approximation as discussed before, the discharge current is assumed to be constant and equal to the peak value for the unconnected logic blocks because it is difficult to predict when the unconnected node is expected to discharge. The unconnected node is identified only during the first iteration of the algorithm. Figure 5.11(a) represents the graphical representation of the sleep region in Figure 5.9(b), node B is identified as an unconnected node. The algorithm then continues as the previous case to sort the rest of the graph. Afterwards, the current vector of the unconnected node B is represented as a rectangle with width equal to the sum of widths of the other vectors and added to the rest of the currents, as shown in Figure 5.11(e).

The last case is when the graph contains one or more loops. Having a loop in the graph makes the topological ordering infeasible, hence, a loop has to be detected before starting the topological sorting algorithm. Thus, before the topological sorting phase, loop detection is employed on the sleep region graph, if a loop is found, then a loop resolving algorithm is used. It was found out that the presence of loops does not change the value of the peak current of the sleep region, it only affects the shape of the discharge current pattern. Whenever a loop is detected,

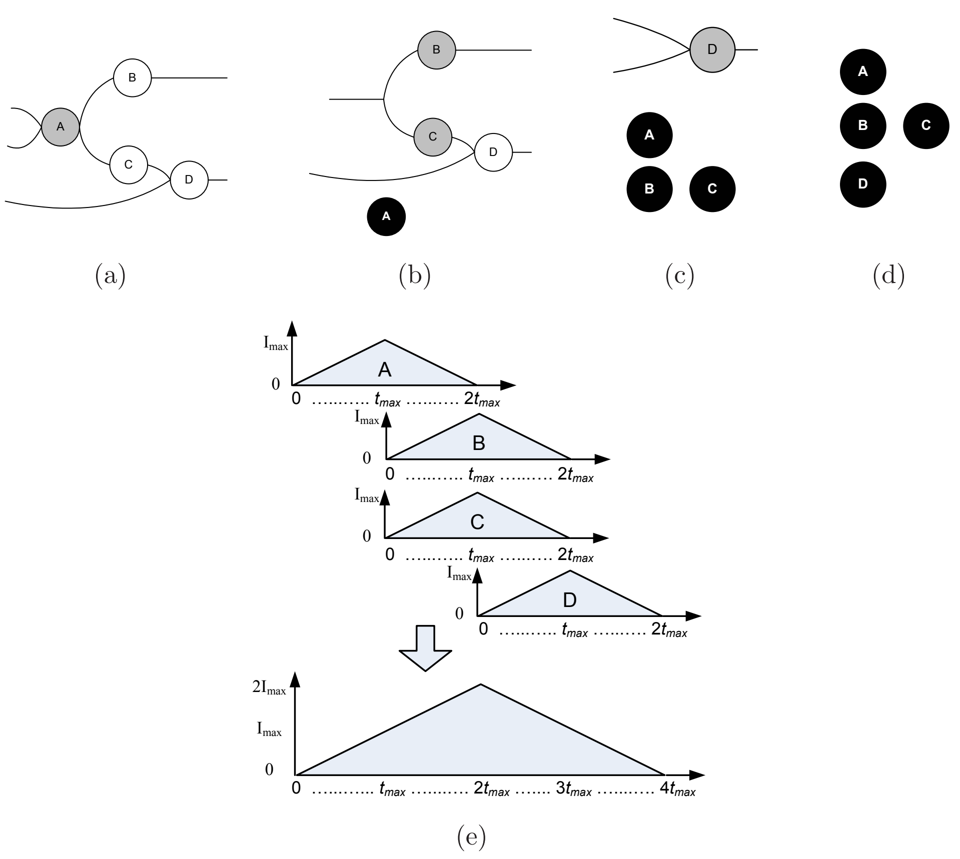
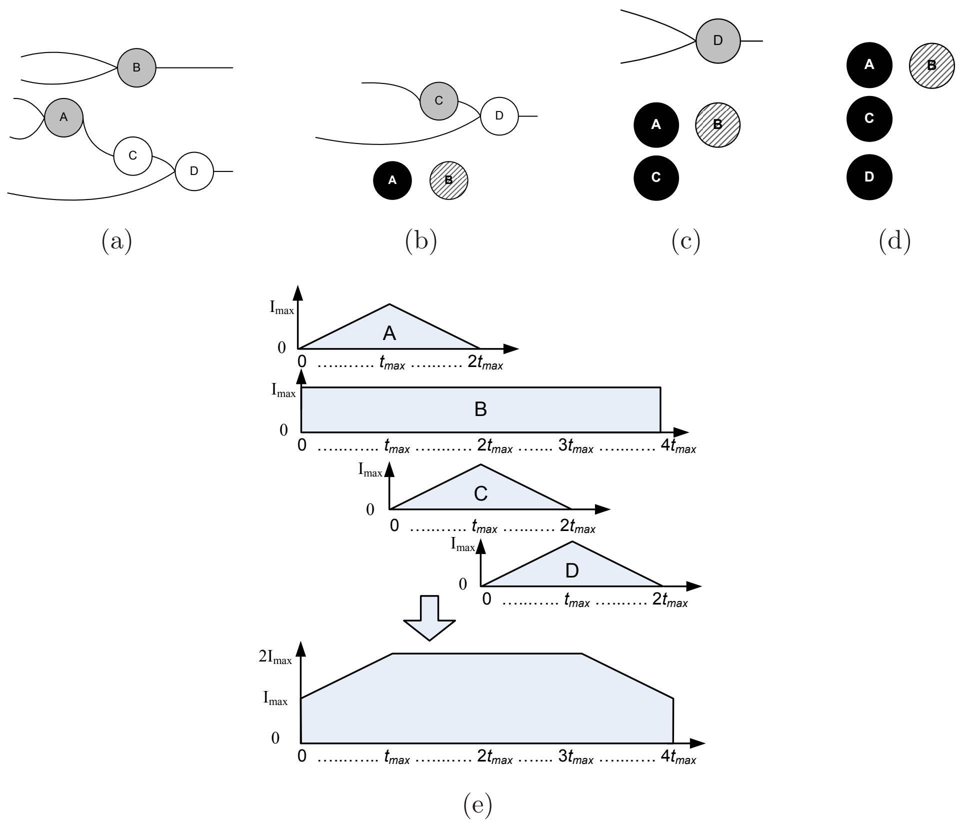


Figure 5.10: Steps of the current feasibility check for a combinational connected sleep region. (a) A is selected to be deleted, (b) A is ordered in the first position and B and C are selected for deletion, (c) B and C are ordered in the same position, (d) Final ordering, (e) Current vectors summation

it is broken at any point while and a virtual edge to represent the broken edge is kept. Afterwards, the algorithm continues in the same manner as earlier.

A pseudocode for the discharge current processing algorithm with topological sorting is listed in Algorithm 5.1. In the first step in Algorithm 5.1, the logic block under consideration is added to the cluster under consideration. Following topological sorting (`Top_Sort`), the logic blocks are checked according to their order in the sorting. If a block is found to be unconnected, then its current vector is treated as a rectangle with a maximum of $I_{\max}$ and starting time of 0 ($rect(I_{\max}, 0)$), as shown in Figure 5.11(e). If 2 blocks are sorted in the same level, as blocks B and C in Figure 5.10(d), then their triangular discharge current either start at the same time or only one of them is considered, depending whether they are mutually exclusive from the block on the upper level. Similarly for blocks from different levels, their triangular discharge current depends on whether their discharge current



(a)
(b)
(c)
(d)
A
B
C
D
I_max
2I_max
0
0 .... t_max .... 2t_max .... 3t_max .... 4t_max
(e)

Figure 5.11: Steps of the current feasibility check for a sequential connected sleep region. (a) Both A and B are selected for deletion, (b) A and B are ordered on the same position, with B an unconnected node and C is marked for deletion, (c) C is ordered in the next position, (d) Final ordering, (e) Current vectors summation.

is mutually exclusive or not.

5.4 Activity Profile Generation

In order to properly explain this phase of the CAD flow, a few definitions will be first presented. An *activity profile* is a representation of the periods that a logic block is active (switching). If a group of logic blocks are expected to switch in the same time periods, then it is said that they have similar activity profiles. In order to maximize the power savings from the use of sleep transistors, logic blocks with similar activity profiles should be packed together and connected to one sleep transistor. The main goal of the activity generation is to identify the logic blocks that have similar activity profile so that the packing algorithm can cluster them together. By the end of this phase, all the logic blocks in the design are given labels to divide them into several

Algorithm 5.1 Pseudocode of the proposed logic-based discharge current processing algorithm.

```
for each block $B$ to be added to activity region $C$ do
  $C = C + B$
  $Sort\_C$ = Top_Sort($C$)
  $curr_C = 0$
  for $i = 0 : \text{size}(Sort\_C) - 1$ do
    if block$_i$ is unconnected then
      $curr_i = rect(I_{\max}, 0)$
    else
      if level$_i$ == level$_{i-1}$ then
        if !(block$_i \oplus$ block$_{i-2}$) then
          $curr_i = trig(I_{\max}, (i-1) \times t_{\max})$
        end if
      else
        if (block$_i \oplus$ block$_{i-1}$ ) then
          $curr_i = trig(I_{\max}, i \times t_{\max})$
        end if
        $curr_C = curr_c + curr_i$
      end if
    end if
  end for
  if $\max curr_C > I_{sleep}$ then
    $C = C - B$
  else
    break
  end if
end for
```

activity regions according to their activity profile. Logic blocks with similar activity labels have similar activity profiles. In this work, two activity profile generation algorithms are proposed: Connection Activity Profile (CAP) generation and Logic Activity Profile (LAP) generation, as well as a modification for the LAP algorithm Reverse-LAP (R-LAP).

5.4.1 Connection-based Activity Profile Generation Algorithm (CAP)

The main criterion used by CAP to identify the activity profiles is *connectivity*. Logic blocks that are connected to each other are expected to have similar activity profiles. The main reasoning behind this assumption is that whenever the inputs to a logic block change, its output is expected to change as well, which in turn will cause the logic blocks connected to its output to switch too. This is a pessimistic approximation of the real case as the change in the output depends on the logic implemented by the logic block.

The algorithm begins with the circuit primary inputs and greedily allocates activity regions as it traverses the circuit netlist by means of a simple depth-first graph search algorithm, thus, resulting in a fast and computationally efficient algorithm. While traversing the circuit netlist, whenever a new logic block is reached, it is necessary to determine whether to add this logic block to the current activity region, or to place it in a new activity region. There are two principal driving costs that need to be considered at each node: the size of the activity region and the attraction of a certain logic block to that activity region.

Reducing the size of the activity region provides the clustering algorithm with more flexibility to pack only those logic blocks that manifest the same activity, not those that have close activity profiles. Although this leads to a greater leakage savings, increasing the number of activity regions results in increasing the number of sleep signals used, thus causing a power-inefficient implementation, as well as complicating the control circuitry for generating these signals. Furthermore, the algorithm must be expansive while each logic block is processed. The addition of any logic block to the current activity region, implies the addition of all of its fan-in and fan-out logic blocks, because the algorithm is connection-based. Consequently, the number of fan-ins and fan-outs of any logic block, should be considered during the process and the cost of adding the current logic block to the current activity region is expressed as

$$cost_1 = \frac{currCap + \alpha \times Neighbors - maxCap}{maxCap}\,, \tag{5.8}$$

where $maxCap$ is the predefined maximum capacity for the activity region, $currCap$ is the current capacity of the activity region, $Neighbors$ is the number of logic blocks *directly* connected to B and not yet placed in any activity region, and α is a weighting constant to control the quality of the final solution. The use of $Neighbors$

provides the cost function with the ability to look around the current logic block to examine which other logic blocks are expected to be attracted to the current activity region, if the logic block under investigation is placed in it. It should be noted that the value of $Neighbors$ can be easily evaluated during the file parsing stage without the need for a special pre-processing phase. The parameters that need to be tuned in (5.8) are $maxCap$ and α.

The value of $maxCap$ should be a function of the circuit size to ensure its scalability with the different circuits. In this work, $maxCap$ is selected as a function of the number of logic blocks on the longest path in the circuit. Reducing $maxCap$ enables the activity generation algorithm to pack only those logic blocks that manifest the same activity, *i.e.*, *closely* connected to each other, not those that have close activity profiles, thus resulting in a large number of activity regions, as well as sleep regions. Although this leads to more leakage savings, yet increasing the number of activity regions results in increasing the number of sleep signals used, thus complicating the control circuitry needed for generating these signals. On the other hand, a large value for $maxCap$ will result in a large activity region with a short sleep time, thus reducing the leakage power savings resulting from the algorithm.

By running the algorithm on several benchmarks for a wide variety of values for $maxCap$, it was found that a value for $maxCap$ of 1.5 times the longest path from input to output in the circuit provides the best results in terms of power savings. The average leakage power savings across the tested benchmarks is plotted in Figure 5.12. Increasing $maxCap$ than 1.5 times the longest path in the circuit results in having excessively large activity regions that have limited leakage power saving capability. On the other hand, decreasing $maxCap$ increases the number of activity regions in the final design, thus resulting in a complex and power-hungry sleep-signal generation circuitry.

On the other hand, α controls the expansive ability of the algorithm. The value of α should range between 0 and 1, where a 0 value means that the algorithm considers that adding the current logic block to the cluster will not attract other logic blocks to it. A value of 1 for α means that adding the current logic block to the cluster will result in adding all of the logic blocks connected to it as well. In this work, the value of α is updated adaptively according to $currCap$ based on the following relation

$$\alpha = \begin{cases} 0.3 & currCap < 0.5 \times maxCap \\ 0.6 & 0.5 \times maxCap \leq currCap < 0.7 \times maxCap \\ 1 & 0.7 \times maxCap \leq currCap \end{cases}$$

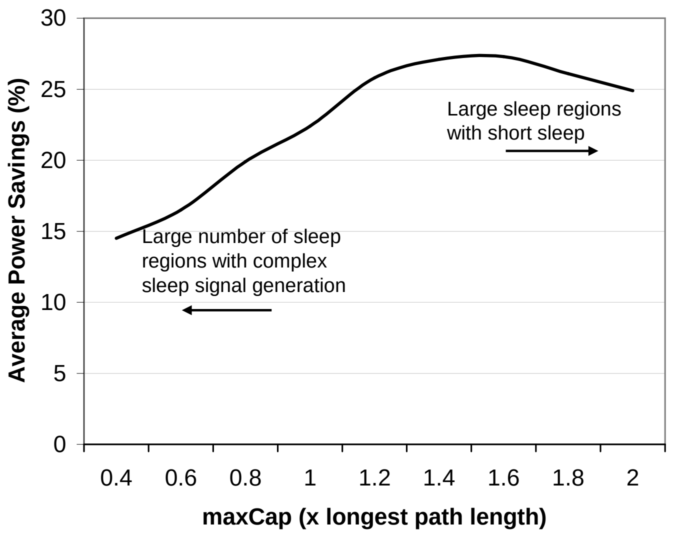


Figure 5.12: Leakage power savings vs the maximum activity region capacity.

The second cost function is the attraction between logic block B and the current activity region C, which is expressed as

$$cost_2 = Nets(B) \cap Nets(C) \,, \tag{5.9}$$

where $\cap$ denotes the number of nets shared between B and C. The decision of whether or not a certain logic block should be added to the current activity region resolves to a comparison between $cost_1$ and $cost_2$

$$\begin{aligned} \delta \times cost_2 - cost_1 \geq 0 &\Rightarrow \text{ add to the current activity region} \\ \delta \times cost_2 - cost_1 < 0 &\Rightarrow \text{ start a new activity region} \end{aligned}$$

where δ is a normalization factor. It should be noted that $cost_1$ is always negative, ranging from -1 to 0, unless the activity region capacity exceeds that of the maximum capacity. On the other hand, $cost_2$ is a positive integer. When δ is close to 0, the activity region maximum capacity $maxCap$ is the main limiting factor to assigning activity labels, thus all activity regions will have capacity equals to $maxCap$. When δ is close to 1, the attraction to the activity region is the driving factor for activity labeling, thus the activity region capacity might exceed $maxCap$. In order to determine the optimum value of δ, the CAP algorithm is run several times for values of δ ranging from 0 to 1 across different benchmarks and the capacity of the activity region is recorded in each case. The average activity region size, in terms of $maxCap$, across all the tested benchmarks is plotted in Figure 5.13, which shows that the average activity region size increases with δ. The value

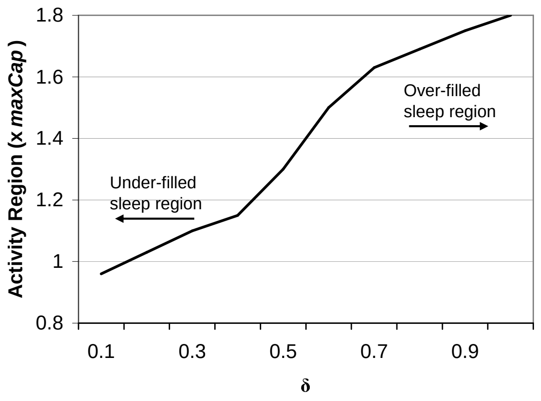


Figure 5.13: Average activity region size across several benchmarks vs. δ.

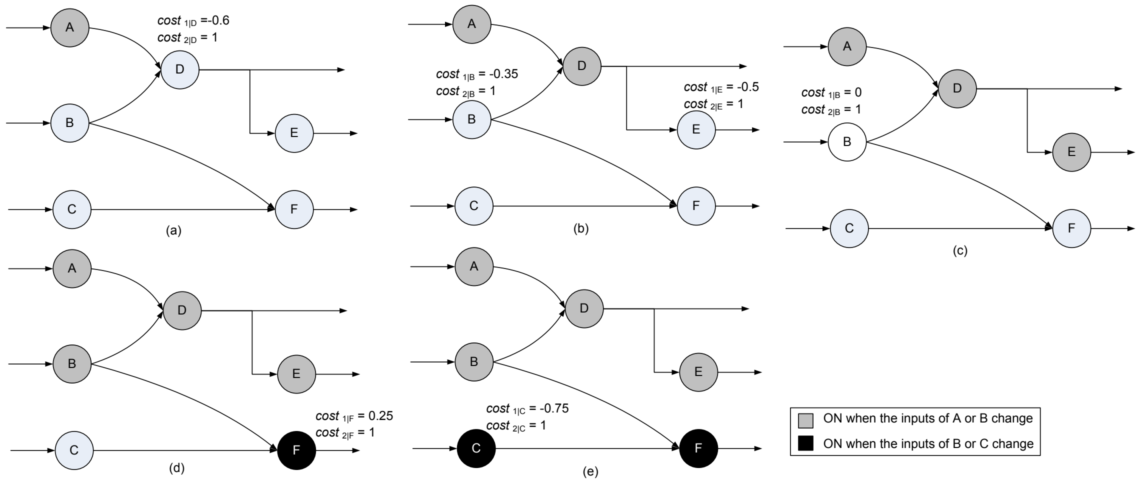


Figure 5.14: CAP activity generation flow for $maxCap = 4$ and $\delta = 0.2$

of δ used in this work is 0.2 which results in an average activity region capacity of about $1.03maxCap$. A value greater than that will result in larger activity regions.

Figure 5.14 depicts an example of activity generation by the modified CAP algorithm for a maximum activity region size of four. Figure 5.14 indicates that the algorithm begins with node A and then studies its child D to select the path that minimizes the total cost function, which in this case is D. Following that the children and parents of D are examined (E and B, respectively). At that point, both B and E have equal $cost_1$, hence, $cost_2$ is checked and E is selected because it has the smallest $cost_2$. The procedure continues until the algorithms starts processing F, at which the activity region will be full and a new activity region will start. Hence, F and C will be in the same activity region. The pseudocode for the modified CAP algorithm is given in Algorithm 5.2.

Algorithm 5.2 Pseudocode of the proposed CAP algorithm.

```
Create an undirected graph from the netlist
Traverse the graph using DFS
for each node $i$ do
  for each node $j$ connected to $i$ do
    calculate $cost_{1,j}$ and $cost_{2,j}$
    if $cost_{1,j} \leq min\_cost_1$ then
      $min\_node = j$
    end if
  end for
  if $\delta \times min\_cost_2 - min\_cost_1 > 0$ then
    add to the current activity region
  else
    start a new activity region
  end if
end for
```

5.4.2 Logic-based Activity Profile (LAP) Generation

The Logic-based Activity Profile (LAP) generation algorithm depends on representing the activities of the logic blocks as a binary sequence. The circuit topology is ignored in the LAP algorithm and instead the circuit logic function is used to find the optimum clustering that prolongs the OFF periods of each logic block. In order to properly explain this algorithm, several definitions and notations will be first introduced.

5.4.2.1 Activity Vectors

Definition 1: Activity Vector
Given a net x in a circuit netlist, the *activity vector* A_x of x is defined as:

$$A_x = [\ a_1 \quad a_2 \quad a_3 \quad .... \quad a_{2^n-1} \quad a_{2^n}]^T \quad ,$$

where n is the total number of inputs to the circuit, a_i is a binary variable that is '1' if any of the outputs of the circuit depend on net x for evaluation when the inputs to the circuit are given by the i^{th} input vector, and T represents the transpose of the vector.

In FPGAs, each logic block has only one output; thus, the activity vector of each net resolves to be the activity vector of the logic block driving that net. The circuit in Figure 5.15 provides an example of the operation of LAP, where the logic of each block is depicted underneath the circuit. Logic blocks F and G must be ON all of the time to generate the outputs of the circuit f and g, respectively. Consequently, the activity vectors A_f and A_g for blocks F and G, respectively, are given by

$$A_f = [\ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ ]^T\ ,$$

$$A_g = [\ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ ]^T\ .$$

On the other hand, for computing the activity vector at the inputs of block F,

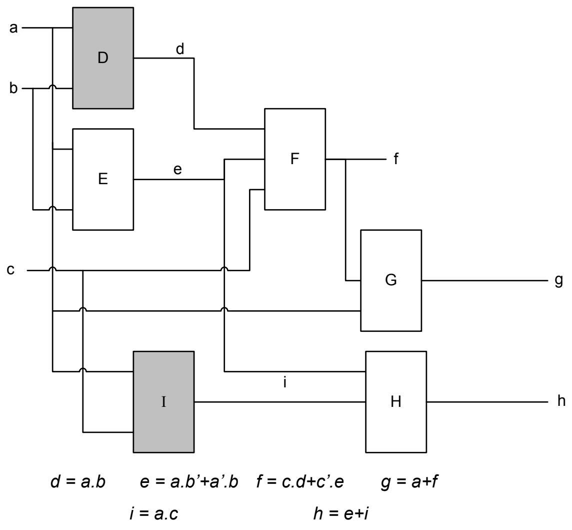


Figure 5.15: A circuit example.

it is noteworthy that block D will be only used to generate the output signal f if the input c is ‘1’. Similarly, block E is only used when c is ‘0’. Hence, the activity vectors for D and E, when f is evaluated, are represented by

$$A_d = [\ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ ]^T\ ,$$

$$A_e|_f = [\ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ ]^T\ .$$

However, to evaluate h, E will have the following activity vector:

$$A_e|_h = [\ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ ]^T\ .$$

Hence, the resulting A_e is given by

$$A_e = A_e|_f + A_e|_h = [\ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ ]^T\ .$$

Finally, the activity vector for i is given by

$$A_i = [\ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ ]^T\ .$$

From this discussion, it can be deduced that if F, G, and H are active for all the input combinations, packing them together will result in improved results. Moreover, E will be active for almost all of the input combinations except for only one, thus it can also be packed with F, G, and H in the same cluster. Therefore, the cluster containing E, F, G, and H will be always ON. On the other hand, D and I have similar activity profiles for half of the input combinations, thus it will be a good strategy to group them together and turn OFF this cluster for half of the circuit operational time.

From the above discussion, it can be deduced that the complexity of the original LAP algorithm is proportional to 2^n where n is the total number of circuit inputs. In large circuits with hundreds of inputs, this complexity renders the LAP algorithm impractical. In the next subsections, several modifications are proposed to reduce its complexity significantly.

5.4.2.2 Hamming Distance: A Measure of the Correlation Between Activity Profiles

The relation between the activity profiles of the different logic blocks is evaluated by means of the *Hamming distance* between their activity vectors.

Definition 2: Hamming Distance
Given two binary sequences of length n; A_n and B_n, the *Hamming distance* $d_{(a,b)}$ between these two sequences is defined as

$$d_{(a,b)} = \sum_{k=0}^{n-1} |a_k - b_k| \,, \tag{5.10}$$

where a_k and b_k are the k^{th} elements of A_n and B_n, respectively.

Hence, the Hamming distances between the activity vectors of the signals in Figure 5.15 are evaluated as

$$\begin{array}{cccc} d_{(f,g)} = 0 & d_{(f,d)} = 4 & d_{(f,e)} = 1 & d_{(f,i)} = 4 \\ d_{(g,d)} = 4 & d_{(g,e)} = 1 & d_{(g,i)} = 4 & d_{(e,d)} = 5 \\ d_{(e,i)} = 5 & d_{(e,h)} = 1 & d_{(d,i)} = 4 & d_{(d,h)} = 4 \\ & d_{(i,h)} = 4 & & \end{array}$$

It can be noticed that the Hamming distance between the activity vectors of any two logic block is a measure of the correlation between their activity profiles. A Hamming distance close to the absolute minimum of zero, indicates that the two blocks will exhibit the same activity profile, thus when positioned together in the same cluster will result in maximum power savings and vice versa. This is verified by examining the values of the Hamming distances above and the results stated in the previous sub-section. However, the Hamming distance between the activity vectors of two logic blocks does not take into consideration the probability of occurrence of the different input combinations, which can notably affect the quality of the results. The *Weighted Hamming Distance* is used to efficiently incorporate the various probabilities of the circuit input combinations.

Definition 3: Weighted Hamming Distance
Given two binary sequences of length n; A_n and B_n, and a weighting vector W_n, the *weighted Hamming distance* $dw_{(a,b)}$ between these two sequences is defined as

$$dw_{(a,b)} = \sum_{k=0}^{n-1} w_k \times |a_k - b_k| \,, \tag{5.11}$$

where a_k, b_k, and w_k are the k^{th} elements of A_n, B_n, and W_n, respectively.

The use of the weighted Hamming distance is not a sufficient measure for the difference in activity between the different logic blocks. If for example, there are 2 logic blocks with a weighted Hamming distance between them of 1. However, this net at which they differ is an active net that continuously toggles. This implies that the ST will switch frequently, increasing the dynamic power dissipation to the extent that it might override any savings in leakage power dissipation. In order to avoid such condition, the *transition density* [80] of the net needs to be considered while calculating the Hamming distance. The transition density D is defined as the average number of transitions per unit time.

Definition 4: Transition Weighted Hamming Distance
Given the weighted Hamming distance $dw_{(a,b)}$ between two activity vectors; A and B, and D_i the transition density of signal i, the *transition weighted Hamming distance* $\overline{dw}_{(a,b)}$ between these two activity vectors is defined as

$$\overline{dw}_{(a,b)} = dw_{(a,b)} \times \max\{D_A, D_B\} \,. \tag{5.12}$$

5.4.2.3 The LAP Algorithm Operation

The LAP algorithm consists of two main phases: activity vector generation and activity labeling. The activity vector generation phase exhaustively simulates the circuit by iterating all the input vectors and finding the values of all the circuit nets resulting from that input vector. Afterwards, for each input vector iteration, each signal (or block) is tested to investigate whether or not it affects the evaluation of the circuit outputs. This is performed by complementing the value of the signal under consideration and then proceeding from that point towards the circuit outputs. If the output of the logic blocks that have this net as an input will change, then this change is taken to the next circuit level, otherwise, a '0' is placed in the corresponding row of the input vector. If a loop is found, then this net is given '1' in its activity vector for that input combination. It should be noted that the number of levels checked from the net under consideration increases the computational time significantly. In order to limit this computational complexity, the number of levels to be checked is limited to three. After exhaustively generating all the activity vectors for all the circuit nets, the static probability of each net is calculated.

The next step is the calculation of the Hamming distance between each two logic blocks in the design. This is performed recursively through all the design elements. The transition weighted Hamming distance $\overline{dw}$ between every two logic blocks is then calculated. At this point, the activity labels can be assigned according to the weighted Hamming distance. However, this approach can result in performance deterioration as the connections between the different logic blocks is not considered. Since those logic blocks that will have a similar activity label are expected to be placed in the same sleep region, *i.e.*, will be placed close to each other. Hence, it seems that the wire length should be included in the activity labeling as well. Since at this stage the algorithm does not have any information about where each block will be placed, an approximation for the wire length is adopted. If any two logic blocks share one net, then the distance between them l is considered as zero, *e.g.*, E and F in Figure 5.15. If there is one level of logic blocks in between any two logic blocks, then the distance is considered as 1, and so on.

In order to combine the transition weighted Hamming distance and the distance between logic blocks, logic blocks are assigned activity labels by minimizing the cost function given below

$$\min\{\overline{dw} + \delta \times l\} \ , \tag{5.13}$$

where δ is a normalization constant selected to be 0.5. To avoid having activity regions with a large number of logic blocks, which will decrease the leakage savings, the size of the activity region is limited to 1.5 times the longest path from input to output in the circuit. This value was obtained by running the algorithm on several benchmarks. Assigning a constant value for activity region size, irrespective of the circuit size, results in impractical results. Increasing the activity region size more than 1.5 times the longest path in the circuit results in having excessively large activity regions that are usually not fully filled up by the algorithm. On the other hand, reducing the activity region size increases the number of activity regions in the final design.

Hence, the algorithm starts to greedily assign activity labels to the logic blocks according to (5.13) until the maximum activity region size is reached. Afterwards, a new logic block is selected as a seed cell for a new activity region and the procedure is repeated. The pseudocode of the algorithm is listed in Algorithm 5.3.

Algorithm 5.3 Pseudocode of LAP.

```
for all the input combinations do
    for all the nets in the circuit do
        find the value of the net
    end for
    for each net in the circuit do
        toggle the value of the net
        Activity[input_vector][net] = 0
        proceed with the new value of the net
        if the value of any output changes then
            Activity[input_vector][net] = 1
        end if
    end for
end for
for each net in the circuit do
    find the static probability
    find the transition density
    find the distance to each net in the circuit
end for
```

5.4.2.4 Reverse Logic Activity Profile (R-LAP) Generation Algorithm

In this section, the Reverse Logic Activity Profile (R-LAP) generation algorithm is presented. The R-LAP algorithm is a modification of the LAP algorithm that offers a significant execution time improvement as well as more leakage power savings. R-LAP represents the logic blocks activity profiles using an activity vector, similar to the LAP algorithm.

5.4.2.4.1 R-LAP Algorithm Operation

In order to reduce the complexity of the LAP algorithm, this work proposes the Reverse Logic-based Activity Profile (R-LAP) generation algorithm. In the R-LAP algorithm, instead of generating the activity vectors for the outputs of each logic block, R-LAP generates the activity vectors of the inputs to each logic block. This is performed by checking each logic block whether its ith input will contribute to the output when the other inputs are given by a certain combination. If the ith input is needed for evaluating the output, then a '1' is placed in the input signal activity vector that corresponds to this input combination, otherwise, a '0' is entered. Hence, the complexity of the algorithm is $O(2^{(m-1)} \times m)$, where m is the number of inputs to the logic block, which is usually around four [28].

As an example, for the circuit shown in Figure 5.15, when 'b' is '0', 'a' is not needed to evaluate 'd'. On the other hand, 'a' is always needed to evaluate 'e'. By taking a look at logic block F, 'e' is not needed to evaluate 'f' when 'c' is '1'. Similarly, 'd' is only needed when 'c' is '1'. Furthermore, 'f' is needed to evaluate 'g' when 'c' is '0'. Hence, the R-LAP algorithm will evaluate the activity vectors of the different signals in Figure 5.15 as

$$A_d|_c = [\ 0 \quad 1\ ] \qquad A_e|_c = [\ 1 \quad 0\ ] \qquad A_f|_c = [\ 1 \quad 0\ ]$$

Hence, it can be deduced that placing F and E in the same sleep region will result in maximum power leakage savings as they both can be turned OFF when 'c' is '1'.

For each logic block, the different activity vectors for all of its inputs are generated as mentioned above. As a result, each net will have p different activity vectors associated with it, where p is the number of the net fanout. In order to find the Hamming distance between the logic blocks inside any sleep region, a large activity vector is generated using the smaller activity vectors of each logic block inside it, and then the Hamming distance is evaluated from the sleep region activity vector. It should be noted that the R-LAP algorithm generates the activity vectors for each logic block with respect to all the other logic blocks it is connected to. Furthermore, sleep regions are usually filled by logic blocks that share connections to reduce the total wirelength and enhance the final design performance. Hence, there is no need to generate the activity vectors for each logic block with respect to all the other logic blocks in the circuit that they do not share a connection with. The pseudocode of the R-LAP algorithm is listed in Algorithm 5.4.

Algorithm 5.4 Pseudocode of the proposed R-LAP.

```
for each logic block $i$ do
    for each input $j$ of $i$ do
        for each of the other inputs combinations $k$ do
            Evaluate the output of block $i$ when input $j$ is ‘0’
            Evaluate the output of block $i$ when input $j$ is ‘1’
            if $i|_{j=0} == i|_{j=1}$ then
                $A_j|_i[k] = 0$
            else
                $A_j|_i[k] = 1$
            end if
        end for
    end for
end for
```

5.5 Activity Packing Algorithms

Modern island-style FPGAs have a hierarchal architecture, where several logic blocks are packed together to form clusters. The packing process takes a netlist of LUTs and registers and outputs a netlist of logic clusters. The main aim of the available packing algorithms is to minimize the total area (by packing clusters to their full capacity), minimize the delay (by packing LUTs on a certain critical path together [18]), and/or maximize routability (by minimizing the number of inputs to each cluster). However, the goal of minimizing power dissipation, either dynamic or leakage power dissipation, has been rarely addressed. In this work, the activity profiles obtained earlier are incorporated into the T-VPack [18] algorithm to pack logic blocks to minimize leakage power dissipation.

5.5.1 Activity T-VPack (AT-VPack)

In this work, the T-VPack algorithm is modified to include activity profiles, thus the modified T-VPack is called Activity T-VPack (AT-VPack). In AT-VPack, a set of logic blocks are selected as candidates to be added to the cluster under investigation. The selection criteria for these candidate logic blocks are: (i) the combined discharge current of the logic blocks inside the cluster plus the logic block to be added does not exceed I_{sleep} and (ii) the activity label of the logic block to be added is the same as that of the logic blocks inside the cluster. From the pool of candidate logic blocks, the one that maximizes *Attraction* and satisfies the three hard constraints of VPack is selected to be added to the cluster. If AT-VPack fails to fill out all of the spaces in the cluster, the hill-climbing approach used in the original T-VPack is invoked to start filling the vacant places while satisfying both (i) and (ii).

Unlike T-VPack, AT-VPack might still be unable to fill the cluster to its maximum capacity due to the additional two constraints (i) and (ii). Hence, a second hill-climbing stage is used that employs simulated annealing to swap the logic blocks in the cluster with other candidate logic blocks that have not been clustered yet and then try to fill the cluster. If the set of logic blocks currently in the cluster is given by A and the set of logic blocks that had not been clustered is called B, the algorithm swaps block i from set A with block j from set B while satisfying constraint (iii) using the following cost function

$$\min\left\{\alpha\left[\kappa\Big(\mathit{Attraction}(A_i)-\mathit{Attraction}(B_j)\Big)+(1-\kappa)\frac{\text{number of vacant places}}{\text{total number of places}}\right]\right\}, \tag{5.14}$$

where α is a variable that represents the transition weighted Hamming distance between A and B_j ($\alpha = 1 + \overline{dw}$) and κ is a weighting constant ($0 \leq \kappa \leq 1$) that is used to give importance to either filling up the cluster with any blocks or to consider the attraction force. A small value of κ would result in a faster filling for the cluster, while a decrease in the *Attraction*() can be tolerated, while a large value will keep the decrease in *Attraction*() to a minimum and accepting partially filled clusters. By performing several experiments using AT-VPack for different FPGA benchmarks, it is found that the best value for κ is 0.5. The value chosen for α forces the algorithm to start looking at first for blocks with the same activity as the cluster before looking for blocks with other activities. Even when it does look for logic blocks with different activities, it always searches for those with close activity profiles. This ensures maximum leakage savings (clusters with blocks that have different activity profiles will be *on* for a longer period).

Moreover, the cost function in (5.14) minimizes the loss in the quality of the solution, in terms of the attraction force, by minimizing the difference between *Attraction*(A_i) and *Attraction*(B_j). Similarly, the current constraint is kept as a hard constraint throughout this hill-climbing stage. By the end of this hill-climbing stage, the cluster is full to its maximum capacity. A pseudocode for AT-VPACK algorithm is listed in Algorithm 5.5. In addition, the starting temperature parameter of the simulated annealing and the number of inner iterations to be performed are chosen not to be large to speed up the packing process and avoid decreasing the quality of the original solution.

Algorithm 5.5 Pseudocode for the AT-VPACK algorithm.

```
Perform T-VPACK with 2 extra constraints:
- $I_{cluster} \leq I_{sleep}$
- $activity_{blocks\ in\ cluster}$ is constant
while there are empty spaces in the cluster do
  for all unclustered blocks do
    find blocks $i$ and $j$ with min cost (equation(5.14))
    add $i$ to the cluster
    if max $I_{cluster}$ ¿ $I_{sleep}$ then
      remove $i$
    end if
  end for
end while
```

The ability of AT-VPack in minimizing the number of logic blocks used in the design can be verified by finding the maximum number of unfilled clusters in each benchmark. Among all of the benchmarks tested, the maximum number of unfilled clusters is 4, which is less than 1% of the total number of clusters in the design.

5.5.2 Force-based Activity T-VPack (FAT-VPack)

The Activity T-VPack (A-VPack) algorithm suffers from long execution time because it added two hard constraints to the conventional T-VPack algorithm; (i) the combined discharge current of the logic blocks inside the cluster plus the logic block to be added does not exceed I_{sleep} and (ii) the activity profile of the logic block to be added is the same as that of the logic blocks inside the cluster. As a result this algorithm suffers from a long runtime. In this work, the FAT-VPack algorithm is proposed to reduce the complexity of the AT-VPack algorithm by getting rid of one of the added hard constraints used in AT-VPack. The activity profiles of the different logic blocks are added to the *Attraction*() function in Eq. (2.7) by including a new activities gain function. It should be noted that the constraint on the maximum discharge current of the activity region is still adopted in the FAT-VPack algorithm.

The *ActivityGain* is a representation of how close is the activity vector of block B to that of cluster C. In R-LAP algorithm, the *ActivityGain*(B, C) of adding block B to cluster C is calculated as

$$ActivityGain(B) = \frac{2^n - \bar{dw}_{(b,c)}}{2^n} \; . \tag{5.15}$$

where n is the number of cluster inputs.

The total FAT-VPack gain function used in this work is given by

$$\begin{aligned} Attraction(B) = (1-\alpha) \quad &\times \quad \Big[\lambda \times Criticality(B) + (1-\lambda) \times SharingGain(B,C)\Big] \\ &+ \quad \alpha \times ActivityGain(B,C) \end{aligned} \tag{5.16}$$

where α is weighting constant ($0 \leq \alpha \leq 1$). Setting a large value for α will force the packing algorithm to pack the blocks that have the shortest Hamming distance in the same cluster, hence, same sleep region, without giving much weight to the timing information and wirelength. In the following experiments, α will be set to 0.5 and the impact of its value on both the leakage savings and speed penalty will be discussed later.

5.5.3 Timing-Driven MTCMOS (T-MTCMOS) AT-VPack

In both the AT-VPack and FAT-VPack algorithms, the total discharge current constraint was kept as a hard constraint for all the clusters. In the T-MTCMOS algorithm, the maximum discharge current is varied from one cluster to the other. From Eq. (5.6), it can be noticed that for the same $\frac{W}{L}\big|_{\text{sleep}}$ of the sleep transistor, the performance loss depends on I_{sleep}. A sleep region with a large I_{sleep} will have a larger performance loss than another one with smaller I_{sleep}, if they employ equal-sized sleep transistors.

This work makes use of this observation to avoid incurring a large performance penalty on the critical path. Hence, the maximum performance loss along the critical path can be limited to a value smaller than that along non-critical ones, *i.e.*, timing-driven MTCMOS (T-MTCMOS). The timing information of the logic blocks is used to vary I_{sleep} of each cluster according to its criticality using the proposed T-MTCMOS technique.

The maximum discharge current inside any cluster should not exceed the value used in (5.6) for a speed penalty of $x\%$. The value of the discharge current can vary from one cluster to the other depending on the criticality of each cluster, hence, the speed penalty imposed on the cluster. In order to account for the different criticalities along the signal paths, $I_{\text{sleep}}(C)$ is formulated as

$$I_{\text{sleep}}(C) = \left[1 + \delta\left(1 - \frac{Criticality(C)}{Max\_Criticality}\right)\right] \times \widehat{I}_{\text{sleep}} \; , \tag{5.17}$$

where $\widehat{I}_{\text{sleep}}$ is the maximum discharge current calculated for the minimum performance penalty, *i.e.*, 3%, δ is a weighting constant, $Criticality(C)$ is the criticality of cluster C, and $Max\_Criticality$ is the criticality of the critical path(s) of the circuit. From (5.17), it can be noticed that if the criticality of the cluster is equal to the maximum criticality of the circuit, *i.e.*, the cluster lies on the critical path, the value of I_{sleep} will be equal to that for the minimum performance penalty, *i.e.*, 3%, otherwise, a larger value for I_{sleep} will be used, hence, a larger performance penalty.

The weighting factor δ is used to make sure that after adding block B to cluster C, the path does not become a critical path itself. If δ is set to a value close to 0, then all of the sleep regions will have an I_{sleep} very close to that for a 3% performance penalty. On the other hand, if δ is set to 1.6, the sleep regions will have a wide variety of I_{sleep} values, hence speed penalties, with a maximum penalty of 8%. However, a large value for δ increases the possibility that the added performance penalty might cause some uncritical paths to become critical. By conducting several experiments on the value of I_{sleep}, it was discovered that by adopting an adaptive update technique for δ, shown below, depending on the criticality ratio ($Criticality(C)/Max\_Criticality$), resulted in no new critical paths while having a wide variety for I_{sleep} values.

$$\begin{array}{rl} 0 < Criticality(C)/Max\_Criticality \leq 0.5 & \delta = 1.6 \\ 0.5 < Criticality(C)/Max\_Criticality \leq 0.8 & \delta = 0.8 \\ 0.8 < Criticality(C)/Max\_Criticality \leq 1 & \delta = 0 \end{array}$$

5.6 Power Estimation

In order to evaluate the performance of the proposed algorithms, the power dissipation in the placed and routed design is compared to that of the same benchmark without sleep transistors. The power model proposed in Chapter 4, which calculates dynamic, short-circuit, and leakage power, is used to estimate the power dissipation in the design without sleep transistors. In order to measure the power dissipation in the design with sleep transistors, several modifications are added to the power model.

There are two standby modes for any circuit; full standby and partial standby. In the fully standby state, the whole circuit is in the idle state and all of the sleep transistors in the circuit should be turned off. During that period, the circuit only consumes standby leakage power. During partial standby, some parts of the circuit are in the active state and other parts are in the idle state. Hence, some of the sleep transistors are turned ON and others are OFF. Thus, the circuit will consume a combination of dynamic power and active and standby leakage power.

The total power dissipation P_t is expressed as

$$P_t = t_{on} \times P_{on} + t_{off} \times P_{idle} , \tag{5.18}$$

where t_{on} and t_{off} are the percentages of ON and OFF times of the FPGA and P_{on} and P_{idle} are the power dissipation during the active and idle modes of operation of the FPGA. P_{on} is expressed as

$$P_{on} = [P_{dyn} + P_{sckt} + P_{leak}]_{utilized} + P_{leak}|_{unutilized} , \tag{5.19}$$

where P_{dyn}, P_{sckt}, and P_{leak} are the dynamic, short-circuit, and active leakage power dissipations, respectively, in the utilized logic blocks, while $P_{leak}|_{unutilized}$ is the standby leakage in the unutilized logic blocks.

Three different modifications were done to the power model presented in Chapter 4. (1) Leakage current is calculated only due to the sleep transistor rather than calculating the leakage through all the devices in the circuit because the sleep transistors act as a bottleneck for the subthreshold leakage current. (2) Short circuit power dissipation is approximated as 15% of the dynamic power dissipation rather than the 10% used in the original model to account for the increased rise/fall times of the logic blocks with sleep transistors. The 15% approximation was evaluated by simulating logic blocks with and without an sleep transistor using HSpice. (3) The dynamic power consumed in the sleep transistor during switching between the ON and OFF states is calculated and added to the total power dissipation.

5.7 Results and Discussions

In this section, the capability of the proposed activity profile generation algorithms presented in Section 5.4 and the packing algorithms discussed in Section 5.5 to reduce leakage power dissipation will be tested. The CAP, LAP, and R-LAP algorithms are integrated into the VPR tool together with the AT-VPack, FAT-VPack, and T-MTCMOS. In addition, the power model presented in Chapter 4 with the modifications discussed in Section 5.6 is used to estimate the power savings achieved by each combination of activity profile generation algorithm and activity packing algorithm in the final design. It should be noted that the proposed logic-based discharge current processing algorithm is used to calculate the discharge currents of the sleep regions. The proposed algorithms are tested on several FPGA benchmarks to assess their capability in minimizing both standby and active leakage power dissipation.

The decision whether to keep a utilized sleep region ON all of the time or dynamically switching between ON and OFF depending on its activity profile is based on a balance between the leakage power savings resulting from turning it OFF during its idle periods and the dynamic power dissipated in the sleep transistor during the transition. Whenever the dynamic power dissipated in the sleep transistor during waking up or putting the cluster into sleep exceeds the leakage savings from any cluster, the cluster is kept always ON. Leakage power savings can be achieved when the cluster stays OFF for a certain period of time, $T_{\text{break even}}$. In this work, the transition density [80], the average number of transitions per cycle, is used as a measure of how long a signal stays in a certain state. Based on the transition density of each sleep signal, if the signal experiences a large number of transitions such that $T_{\text{break even}}$ is never or rarely reached, the sleep region is kept always ON, otherwise, it is dynamically turned ON and OFF depending on the activity profile.

5.7.1 Experimental Setup

In the first set of experiments, the sleep region size is set to one cluster and each cluster has four logic blocks. This is only a starting point for the experiments and later on the optimum size of the sleep region will be evaluated. Moreover, the selected size is close to the optimum sleep region size reported in [118].

It should be noted that the maximum allowable performance loss due to the sleep transistors in all of the benchmarks is kept fixed at 5% in the AT-VPack and FAT-VPack algorithm, *i.e.*, x in Eq. (5.6) is set to 0.05. However, in the case of the T-MTCMOS algorithm, the value of x is varied depending on the criticality of the logic blocks in the sleep region.

All the circuits tested are mapped onto the smallest square FPGA array that can accommodate them, *i.e.*, maximum utilization percentage. The case where the design is mapped onto the minimum FPGA array is called *100% utilization*. Moreover, the design is assumed to be operating without standby periods for the whole benchmark. This case is referred to as *100% ON time*. Initially, the results reported are for a 90nm CMOS process, however, towards the end of this section, the proposed leakage reduction algorithms are applied to 130nm, 65nm, and 45nm CMOS technologies.

5.7.2 Algorithms Comparison

Table 5.1 lists the results of applying the different activity generation algorithms with a variety of the proposed activity packing algorithms on several FPGA benchmarks under the above mentioned conditions. The power dissipated by each design is calculated using the modified power model discussed in Chapter 4 and the percentage savings in the total power are listed in Table 5.1. It should be noted that Table 5.1 does not iterate the results from all the possible combinations of the proposed algorithms. Only the combinations that achieve large leakage power savings are reported.

Table 5.1: Leakage power Savings for the different activity profile packing algorithms across several FPGA benchmarks.

Benchmark	% of Unutilized Clusters	% Savings in Power (100% on time)			
		CAP & AT-VPack	LAP & AT-VPack	R-LAP & FAT-VPack	R-LAP & T-MTCMOS
alu4	4.5	10.82	17.23	37.05	61.78
apex2	2.48	10.35	15.54	29.09	55.8
apex4	2.16	8.59	13.38	25.74	50.91
bigkey	2.72	9.59	14.92	30.42	56.62
clma	0.76	7.37	12.09	24.48	50.82
des	0.25	7.49	11.99	24.47	51.97
diffeq	6	11.51	17.74	36.07	57.48
dsip	4.7	8.79	14.18	30.23	50.49
elliptic	5.9	10.2	16.42	32.03	51.24
ex1010	0.26	7.73	12.09	23.07	48.49
ex5p	6.92	11.14	17.51	35.92	54.3
frisc	1.56	8.08	12.97	24.51	50.98
misex3	2.21	9.37	14.57	29.86	58.33
pdc	0.78	6.8	11.29	21.5	46.34
s298	8.32	14.28	21.06	40.94	56.75
s38417	5.6	12.46	18.21	36.34	60.19
s38584.1	1.56	8.17	12.93	25.73	51.51
seq	0	4.62	7.14	15.23	32.03
spla	3.6	8.73	12.85	27.41	50.08
tseng	8.3	13.25	19.75	39.32	56.37
Average Leakage Power Savings (%)		9.47	14.69	29.47	52.62

The power savings presented in Table 5.1, show that the combination of the R-LAP and the FAT-VPack algorithms provide more leakage power savings than the combination of CAP and LAP with AT-VPack. Furthermore, integrating the T-MTCMOS algorithm with R-LAP results in the highest power savings.

The combination of logic-based discharge current processing, R-LAP, and FAT-VPack result in higher power savings than the combination of CAP and LAP with

AT-VPack because the FAT-VPack algorithm can cluster logic blocks that have close activity profiles, not necessarily the same activity profile, by finding a correlation between their activity profiles, hence, achieve more leakage savings. On the other hand, the AT-VPack algorithm can only pack the logic blocks with the same activity profiles in the same cluster, otherwise, the cluster is left ON at all times. In addition, the logic-based discharge current processing algorithm provides the packing algorithm with the flexibility to pack those logic blocks that have similar activity profiles without violating the discharge current constraint. As a result, more power savings can be achieved.

On the other hand, the T-MTCMOS algorithm achieves more power savings than FAT-VPack across all of the benchmarks. The average improvement in the power savings is almost 50%. The main reason behind the increase in power savings is that in FAT-VPack, the discharge current constraint is a hard constraint across all of the benchmarks, thus the algorithm might fill a cluster with logic blocks that have different activity profiles and satisfy the current constraint, although there are other blocks that have closer activity profiles but violate the current constraint. On the other hand, T-MTCMOS allows the current constraint to be violated to a certain extent along non-critical paths, thus giving more freedom to the packing algorithm to pack logic blocks with close activity profiles to achieve more leakage power savings.

5.7.3 Impact of Activity Packing on Performance

As mentioned earlier, the use of sleep transistors results in a performance penalty due to the added resistance of the sleep transistor to the ground. Moreover, both FAT-VPack and T-MTCMOS do not result in the same packing as that found by the conventional T-VPack, because of the added constraints, either discharge current or activity profile, or gain functions to the optimization problem. Hence, the resulting packing might suffer from an additional speed degradation because of that reason. In this work, the performance loss of the critical path is considered as an indication of the performance loss for the whole design. In this experiment, the delays along the critical paths in the placed designs and packed using the proposed packing algorithms are compared to those when the designs are packed and placed using the conventional VPR flow. Figure 5.16 plots the average speed penalties among all of the benchmarks used in this work. Moreover, Figure 5.16 shows the maximum and minimum speed penalties experienced in the different benchmarks for each case.

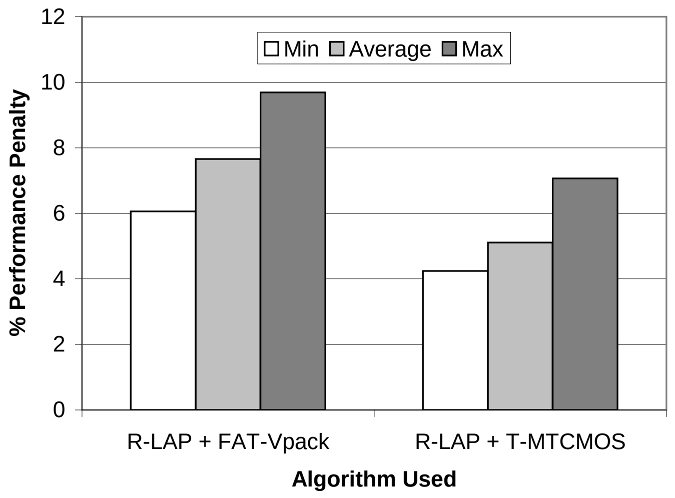


Figure 5.16: Speed penalty experienced in the different benchmarks due to the use of sleep transistors.

From Figure 5.16 it can be deduced that the resulting design from T-MTCMOS outperforms that of FAT-VPack in terms of timing properties. The FAT-VPack algorithm incurs a minimum of 5% delay penalty across all the paths in the design. However, T-MTCMOS increases the delay across the critical path by a minimum of 3% while making sure no other critical paths get created.

In another experiment, the maximum performance penalty allowed in T-MTCMOS is varied from 8% to 14%, while the minimum performance penalty is kept at 3% and the results for the 's298' benchmark are plotted in Figure 5.17. It was noticed that for a sleep region of size 4 logic blocks, the leakage savings increased with the maximum speed penalty until a speed penalty of 10%, after which the curve almost flattens. The increase in leakage savings can be justified by the fact that increasing the maximum speed penalty allows the packing algorithm to pack logic blocks that exhibit similar activity profiles in the same cluster without worrying about their discharge current. On the other hand, as the speed penalty is increased beyond a certain limit, the packing algorithm can not achieve more leakage savings because the sleep regions are now packed to their maximum (4 logic blocks). However, as the number of logic blocks per sleep regions is increased, more leakage savings can be experienced, as shown in Figure 5.17. It should be noted that increasing the size of the sleep region beyond 8 logic blocks, results in an increase in the leakage savings, however, the dynamic power dissipation in the sleep transistors, which are significantly up-sized, increases to cancel out most of the leakage savings.

In another experiment, the maximum speed penalty is set to 12 for the 's298'

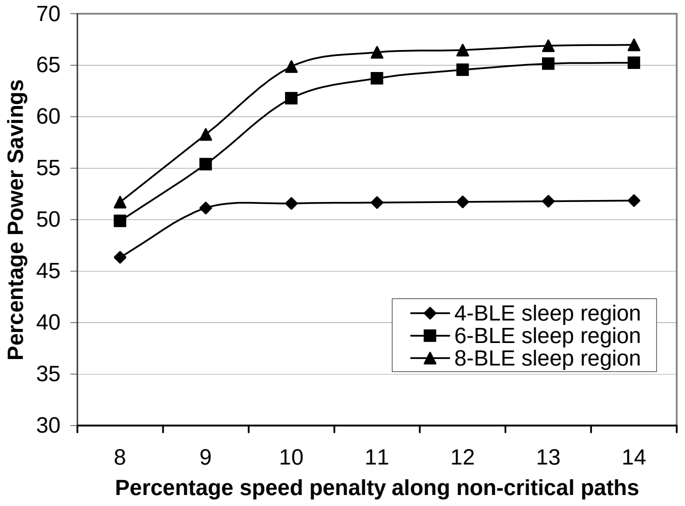


Figure 5.17: 's298' leakage savings vs maximum speed penalty for the R-LAP and T-MTCMOS combination.

benchmark while varying the minimum speed penalty (Figure 5.18). As the minimum performance penalty increases (by up-sizing the sleep transistor), the leakage savings increases, until a certain limit after which the savings decrease because of the increase in dynamic power of the sleep transistors. It can be noticed that the breakpoint gets smaller as the size of the sleep region increases because larger sleep regions employ large sleep transistors.

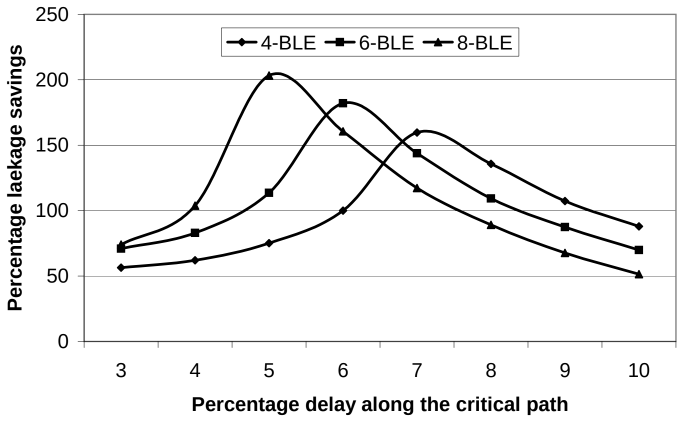


Figure 5.18: 's298' leakage savings vs minimum speed penalty for the R-LAP and T-MTCMOS combination.

Figure 5.19 plots the relative path delays distribution in the 'ex5p' benchmark with respect to the critical path delay. From Figure 5.19 it can be deduced that the number of critical paths did not increase. In addition, the maximum circuit delay changed by only 3%. The final shape of the delay distribution can be varied by changing δ.

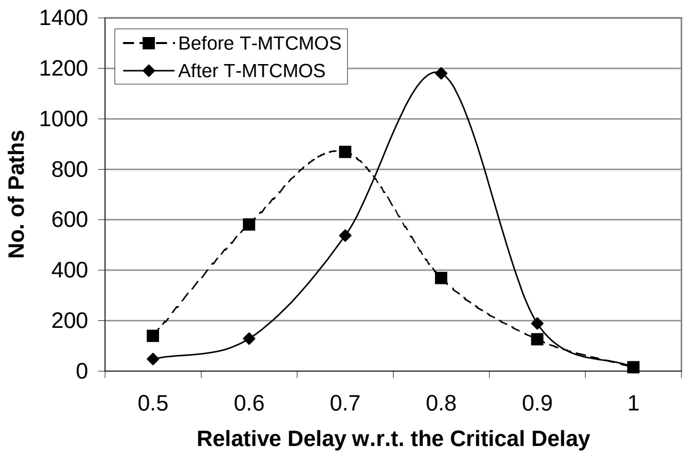


Figure 5.19: Critical path distribution for timing-driven MTCMOS designs.

5.7.4 Leakage Savings Breakdown

In each benchmark, the leakage power savings consist of two parts; savings from permanently turning OFF all the unused clusters and savings from dynamically turning ON and OFF the used clusters in the design depending on their activity profile. By taking a look at the results for the 'seq' benchmark in Table 5.1, this benchmark has no unused clusters while the power savings achieved ranges from 7.14% to 15.23%, depending on the combination of the activity profile generation and the packing algorithms used. This power savings is entirely from dynamically turning ON and OFF the different used clusters in the design depending on their activity profile. On the other hand, the 's298' benchmark has the maximum percentage of unused blocks among all of the benchmarks and it resulted in the maximum power savings, ranging from 21.06% to 40.94% depending on the activity generation and packing algorithm used.

In order to quantify the leakage power savings provided by the unused and used clusters of the design, the power savings from each source is recorded for each benchmark. Figure 5.20 plots the average power savings achieved by each combination of algorithms used across all the benchmarks. It should be noted that the average power savings achieved by turning OFF the unused clusters is 4.92% and is constant across all the combination of algorithms used. This is mainly because all the combination of algorithms provide almost the same number of clusters after packing, hence, the number of unused clusters remains the same. Figure 5.20

shows that even for the least power efficient combination of algorithms (CAP & AT-VPack), the power savings from the used clusters is almost double that from the unused clusters. The contribution of the used clusters to the total power savings increases with the algorithm efficiency.

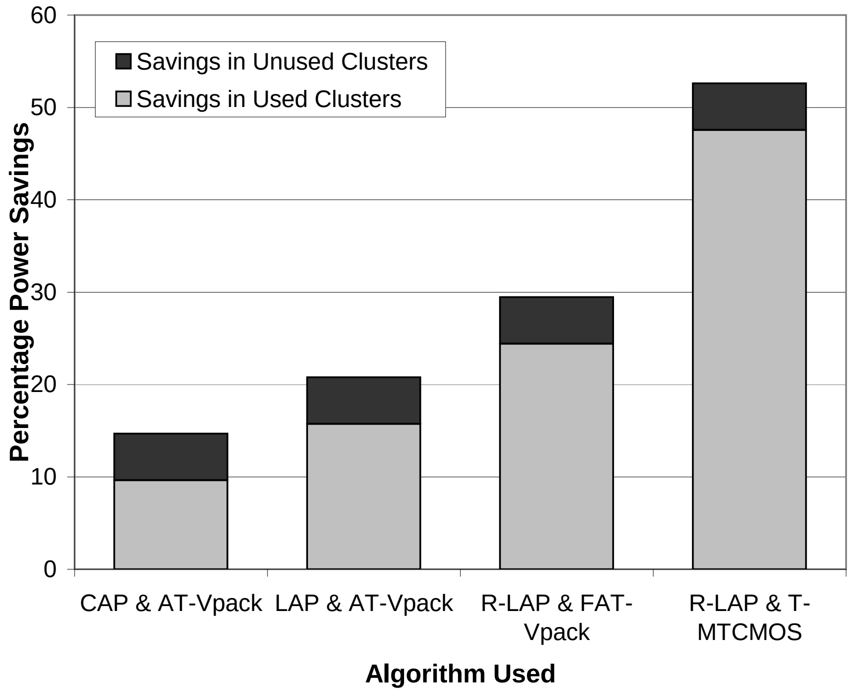


Figure 5.20: Leakage power savings breakdown.

5.7.5 Impact of Utilization and ON Time on Leakage Savings

In reality, the utilization percentage is less than the 100% utilization assumption used in finding the results in Table 5.1. Typically, the utilization in FPGAs ranges from 80% to 60% [55]. In order to investigate the impact of the utilization percentage on the total power savings by turning OFF the unused logic blocks, the benchmarks are mapped on a larger FPGA fabric and the results are plotted in Figure 5.21 for utilization percentages of 80% and 60% using the R-LAP and T-MTCMOS combination. It can be noticed that the power savings achieved by permanently turning OFF the unused clusters increases almost exponentially with decreasing the utilization percentage.

Moreover, the 100% ON time assumption made earlier is impractically high. The average on time of most applications is around 50% to 20% for some hand-

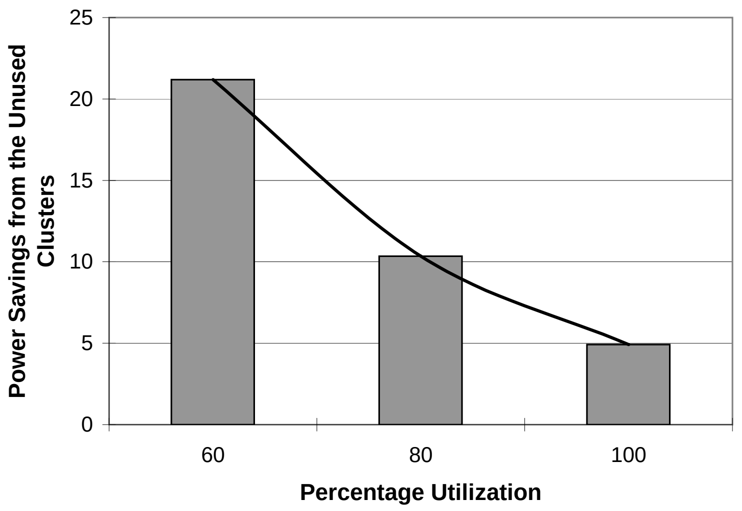


Figure 5.21: Percentage savings in power for different FPGA fabric utilizations using the combination of R-LAP and T-MTCMOS.

held applications [111]. Hence, the same benchmarks are tested again using ON times of 100%, 60%, and 40% and the average power savings from the used clusters of the FPGA are plotted in Figure 5.22. From Figure 5.22, it can be noticed that with reducing the operational time increases the total power savings from the proposed algorithms significantly.

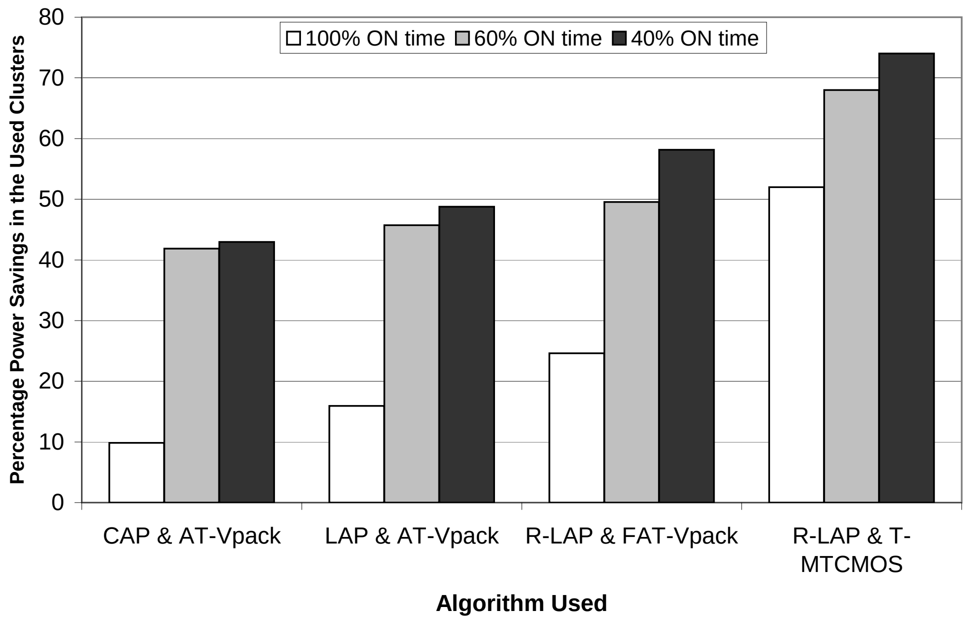


Figure 5.22: Percentage savings in power for different utilizations and operational time using the combination of R-LAP and T-MTCMOS.

5.7.6 Impact of the Sleep Region Size

In another experiment, several sizes for the sleep region are tested for different cluster sizes. The size of the cluster is changed from 3 to 6 logic blocks and the size of the sleep region is changed from 1 to 5 clusters. The leakage savings in each of these experiments are recorded and plotted in Figure 5.23. From Figure 5.23, it can be noticed that for each cluster size, there is an optimum sleep region size. Moreover, leakage savings is always maximum for sleep regions of size around 8 logic blocks. This proves the fact stated earlier that too large (will require a large sleep transistor, which results in large standby leakage and dynamic power dissipation in the sleep transistor) and too small (will result in partially unfilled clusters, which will increase the area and decrease the number of permanently OFF sleep regions, hence, increases the total leakage power) sleep regions will result in lower leakage savings.

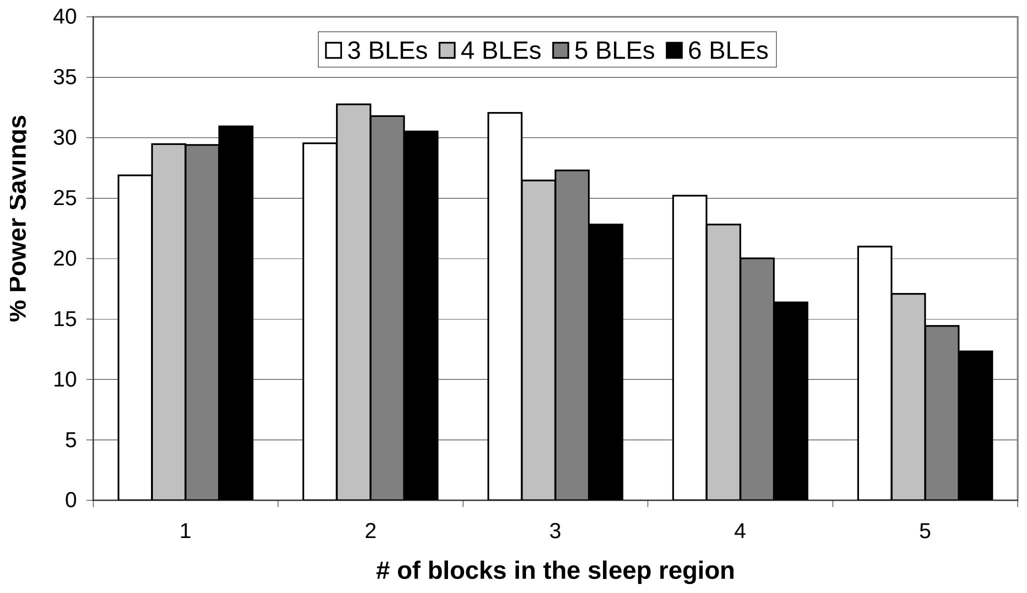


Figure 5.23: Impact of the sleep region size on the leakage savings.

5.7.7 Scalability of the Proposed Algorithms with Technology Scaling

In order to investigate the scalability of the proposed algorithms, the R-LAP FAT-VPack combination as well as the R-LAP T-MTCMOS combination are applied to several current CMOS technologies (130nm, 90nm) and predictive CMOS technologies (65nm, 45nm) [124]. The average power savings across all the benchmarks are plotted in Figure 5.24.

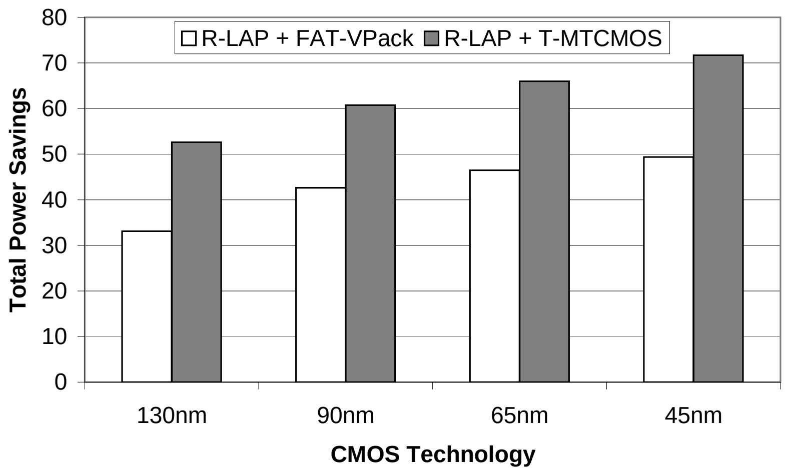


Figure 5.24: Impact of technology scaling on power savings.

5.8 Conclusions

In this Chapter, the proposed MTCMOS architecture for FPGAs is introduced. Moreover, the developed algorithms for activity generation (CAP, LAP, and R-LAP) and activity packing (AT-VPack, FAT-VPack, and T-MTCMOS) are presented. It was found that the combination of the R-LAP algorithm and T-MTCMOS results in the maximum leakage power savings and minimum performance penalty on the final design.

Chapter 6

Leakage Power Reduction in FPGAs Through Input Pin Reordering

Input dependency of leakage power has been witnessed in VLSI circuits in general [125] and in FPGAs in particular [48], where it was reported that 4X variations in leakage power can be experienced in commercial 90nm FPGAs.

Input signal forcing techniques have been used in [53] to reduce the active leakage power dissipation in FPGAs. Since leakage current is heavily state dependent, by manipulating the inputs of some logic blocks, the unused parts of the FPGA can be placed in a low-leakage state. Moreover, by utilizing the complements of the signals, the authors have managed to reduce the total leakage power of the utilized parts of the FPGA. However, the methodology in [53] is based on the assumption that only one output state can result in the minimum leakage power dissipation. This is basically due to the fact that the authors only studied the power dissipation in the inverters, without trying to find a low-leakage state in the pass-transistor multiplexers. It will be demonstrated in this thesis that there is more than one low leakage state that can be further exploited to achieve a bigger reduction in leakage power dissipation. The technique proposed in [53] focuses on leakage power minimization only in the inverters and buffers of the FPGA without considering leakage power minimization in the other parts of the FPGA, including the pass-transistor multiplexers.

In FPGAs, input signal forcing is a substantial leakage power reduction technique, since FPGAs depend on pass-transistor logic in their design, where power dissipation is strongly state dependent. In this Chapter, a complete new methodology, based on *input pin reordering*, is developed to reduce the total leakage power dissipation in all components of FPGAs, unlike [53] that focuses only on the inverter, without incurring any area or performance penalties in the final design. In the proposed methodology, the logic and routing resources are handled differently to achieve maximum leakage savings. Moreover, a modified version of the proposed methodology is implemented to improve the performance along the critical path, and still achieves significant leakage power savings in the design. Moreover, the impact of technology scaling on the lowest leakage states is investigated in this work.

This Chapter is organized as follows; the state dependency of leakage power in FPGAs is discussed in Section 6.1. The proposed input pin reordering algorithm is introduced in Section 6.2. Finally, the results of applying the proposed algorithm on the generic FPGA architecture are discussed in Section 6.3.

6.1 Leakage Power and Input State Dependency in FPGAs

Leakage current in nanometer CMOS technologies has two main components; subthreshold and gate leakage currents. Subthreshold leakage current flows from the drain to the source of an OFF CMOS device. On the other hand, gate leakage current flows through the gate of the device to or from one or both the diffusion terminals. Gate leakage has both an ON and OFF component, with its OFF component almost ignorable relative to the ON part [46]. Both components of leakage power exhibit strong dependency on the state of the input signals as discussed in this Section.

6.1.1 Subthreshold Leakage Current

The subthreshold leakage current I_{sub} is defined as the current that flows between the drain and source of a MOS device when V_{GS} is less than V_{TH}. I_{sub} is formulated as

$$I_{sub} = \mu_o C_{ox} \frac{W}{L}(m-1) \times v_T^2 \times e^{(V_{GS}-V_{TH})/mv_T} \times (1 - e^{-V_{DS}/v_T}) \,, \tag{6.1}$$

where μ_o is the device mobility, C_{ox} is the oxide capacitance, W and L are the device dimensions, v_T is the thermal voltage (kT/q), and m is the subthreshold swing coefficient, which is given by

$$m = 1 + \frac{3t_{ox}}{W_{dm}} , \tag{6.2}$$

where t_{ox} is the oxide thickness and W_{dm} is the maximum depletion layer width. The contribution of the subthreshold leakage current to the total power dissipation increases with technology scaling due to the continuous reduction in V_{TH} to improve the device performance.

The input state dependency on the subthreshold leakage current can be readily seen in Eq. (6.1) in the dependence of I_{sub} on V_{DS} and V_{GS}. Two dominant factors affect the input dependency of subthreshold leakage current: *Drain Induced Barrier Lowering* (DIBL) and *Body Effect*. Subthreshold leakage current is also a strong function of the temperature, increasing significantly with increasing the chip temperature.

6.1.1.1 Drain Induced Barrier Lowering (DIBL)

In nanometer CMOS devices with short channels, the drain-source potential has a strong impact on the band bending over a significant part of the CMOS device. As a result, the threshold voltage of the CMOS devices becomes a function of the drain source voltage. Applying a large drain to source voltage to the CMOS device results in decreasing the threshold voltage, hence, increasing the subthreshold current. Figure 6.1 plots the change in V_{TH} and the subthreshold leakage current I_{sub} of a minimum size 90nm NMOS device with the change in V_{DS} from 0V to 1.2V. It should be noted that this 90nm CMOS process has a supply voltage of 1.2V, thus the change in the drain to source voltage plotted in Figure 6.1 can be readily experienced during operation. Figure 6.1 shows that for two equal-sized transistors, their V_{TH} can differ by almost 25% and their leakage current can vary by 4.5X, due to the DIBL effect, because of a difference in V_{DS} equal to the supply voltage.

Pass-transistor multiplexers used in FPGAs can experience four different values of V_{DS}, as shown in Figure 6.2. The transistors in the first and last stages of the multiplexer are the only ones that can experience the worst case V_{DS} of V_{DD}. The middle stages can experience a maximum of $V_{DD} - V_{TH}$ because of the weak '1' passed by the NMOS pass-transistors. From Figure 6.2, it can be deduced that the maximum leakage current occurs when the signals at both diffusion terminals of a multiplexer transistor are different, *i.e.*, to have the largest value of V_{DS}.

From the above discussion, it can be concluded that a means of reducing subthreshold leakage in pass-transistors multiplexers is to ensure that the majority of

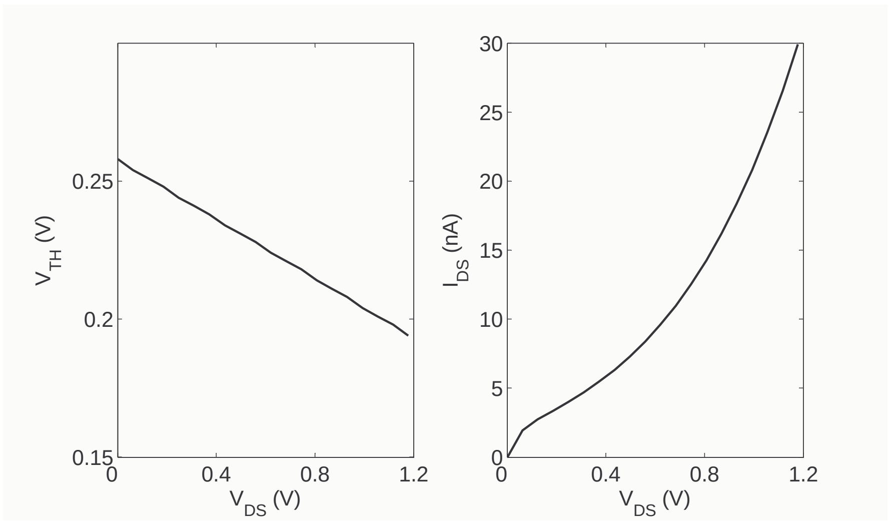


Figure 6.1: DIBL effect in a 90nm CMOS process.

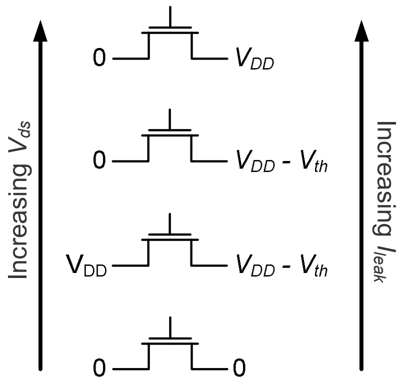


Figure 6.2: DIBL impact on subthreshold leakage in FPGA pass-transistor devices.

the pass-transistors experiences the smallest V_{DS}. It should be noted that if the first stage of the multiplexer is designed to have the smallest V_{DS}, the total multiplexer subthreshold current will be limited significantly, since the total subthreshold current has to flow through them.

6.1.1.2 Body Effect

The impact of the body to source voltage V_{BS} on V_{TH} has been witnessed in CMOS devices for several technology generations. The effect of body bias is formulated as

$$V_{TH} = V_{TH0} + \gamma\Big(\sqrt{|\Phi_s| - V_{BS}} - \sqrt{|\Phi_s|}\Big) , \tag{6.3}$$

where V_{TH0} is the ideal V_{TH} at zero V_{BS}, γ is the body bias coefficient, and Φ_s is the surface potential. Having a negative V_{BS} would result in increasing the threshold voltage, which in turn will reduce the subthreshold leakage current.

It should be noted that CMOS devices in pass-transistor multiplexers will never experience a positive V_{BS}, since the body of the pass-transistors is always connected to GND. Pass-transistors with logic '0' at one or both of the diffusion terminals will not experience body effect as V_{BS} would be zero. However, those devices with logic '1' at both their diffusion terminals will experience subthreshold leakage current reduction due to body effect because their $|V_{BS}|$ would be maximum, either V_{DD} or $V_{DD} - V_{TH}$.

6.1.2 Gate Leakage

Gate leakage exists in both the ON and OFF states of the CMOS devices [46]. However, the off component of the gate leakage is ignorable with respect to the ON component. The value of gate leakage is a strong function of both V_{GS} and V_{DS}. Large values of $|V_{GS}|$ and small values of V_{DS} generate a large gate leakage current. Figure 6.3 shows the two dominant gate leakage current configurations and how they depend on the input state. The gate leakage resulting from the other input configurations is much smaller than these two configurations and can be safely assumed zero, at least for the 90nm CMOS process used in this thesis. It should be noted that the gate leakage current is not a function of the temperature, and thus, stays constant with the change in the chip temperature.

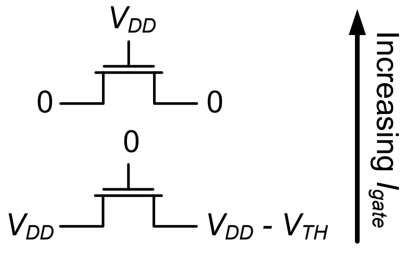
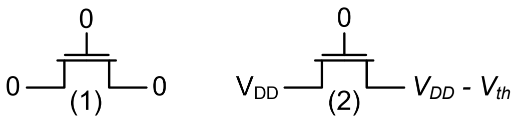


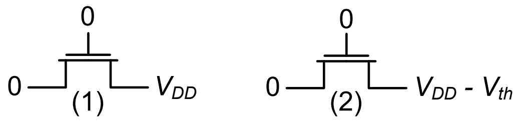
Figure 6.3: Gate leakage dominant states in FPGA pass-transistor devices.

6.1.3 Low-Leakage States in Pass-Transistor Multiplexers

From the above discussion, it can be concluded that there is one or more input states where the leakage power will be minimum in pass-transistor multiplexers. The input state depends on the relative magnitude of the subthreshold leakage current to that of the gate leakage current. In [96] it was reported that the gate leakage power dissipation is less than 1/20 of the subthreshold leakage power dissipation in 90nm FPGAs at room temperature. In the experiments done in this work, the maximum gate leakage current in a 90mn minimum-sized device is in the order of 300pA, which is much less than than the smallest subthreshold leakage current measured which is in the order of 1nA. Moreover, the contribution of the gate leakage power decreases with the increase in temperature due to the strong dependence on subthreshold leakage power on the temperature. Consequently, in this work, the most dominant leakage states are considered to be those of the subthreshold leakage current. Figure 6.16 shows the input states that result in the lowest and highest leakage current that can be experienced in FPGAs pass-transistors.



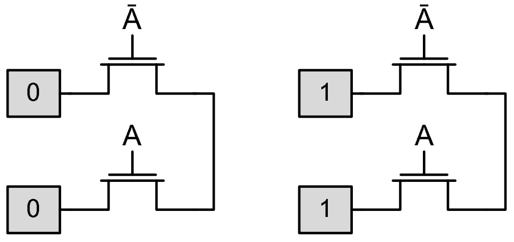
(a) Lowest leakage states.



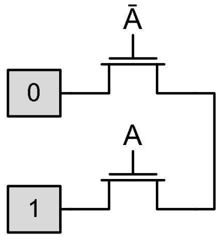
(b) Highest leakage states.

Figure 6.4: Total leakage dominant states in FPGA pass-transistor devices.

The configuration labeled (1) in Figure 6.4(a) results in the lowest leakage current. The highest leakage state is the one labeled (1) in Figure 6.4(b), which experiences the highest V_{DS}, hence, the maximum DIBL effect and no body effect. By looking at the low leakage states shown in Figure 6.4(b), it can be deduced that the lowest leakage states occur if every pair of pass-transistors in the multiplexer have inputs with similar value, as shown in Figure 6.5(a). The highest leakage state occur whenever the inputs to the multiplexer pair are different, as shown in Figure 6.5(b).



(a) Pass-transistor pair with similar inputs.



(b) Pass-transistor pair with different inputs.

Figure 6.5: Inputs to pass-transistors pairs.

6.1.4 Leakage Power in Inverters/Buffers

In this experiment a minimum-sized inverter is designed to have equal rise and fall times to minimize short circuit power dissipation. This is achieved by increasing the width of the PMOS device while keeping the NMOS device to minimum width to balance the difference in mobility of the two devices. The inverter is then simulated using HSpice and the total leakage power is recorded in both cases when the output of the inverter is '1' and '0'. The values of the measured leakage current are recorded in Table 6.1. As seen in Table 6.1, even if the PMOS and NMOS devices of the inverter are designed to have equal driving capabilities, the NMOS still leaks more than the PMOS device. The ratio between the NMOS total leakage current to that of the PMOS is almost 2X. This is mainly due to the inverse narrow width effect experienced by trench isolated CMOS devices. PMOS devices in trench isolated devices experience an increase in V_{TH} with the initial increase in the width of the device. Afterwards, the leakage current starts increasing with the device width. As a result, PMOS devices in the inverters sink a smaller subthreshold leakage current than the NMOS devices because of their larger width.

Table 6.1: Leakage current in a minimum-sized inverter.

Inverter Output	Total Leakage Current
'0'	17.03nA
'1'	31.12nA

This is an interesting phenomenon as it can be used to further reduce leakage inside LUTs pass-transistors multiplexers. Multiplexers need inverters to generate the complement of the control signals, which are generated inside the LUT using minimum-sized inverters similar to the one simulated above, as shown in Figure 6.6. If the input control signal are all zeros, A and B in Figure 6.6, then all the inverters would have a high output, thus, sinking the largest leakage current. Hence, it is more leakage efficient to avoid having the most probable input state being all zeros, where all the inverters would generate the highest leakage current.

6.2 Proposed Input Pin Reordering Algorithm

The proposed pin reordering algorithm for leakage power reduction utilizes the conclusions developed in Section 6.1 to reduce total leakage power in the pass-transistor multiplexers. The algorithm consists of two phases; the first one targets

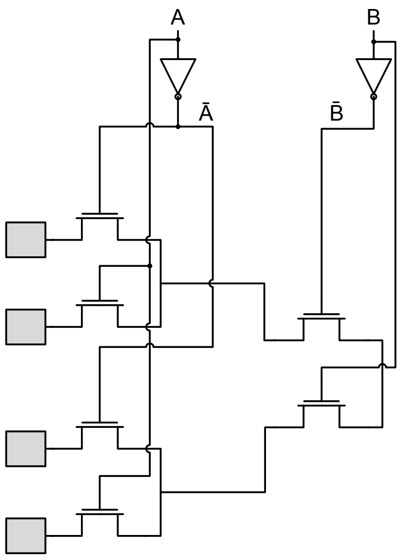


Figure 6.6: Gate leakage dominant states in FPGA pass-transistor devices.

leakage reduction in the FPGA logic blocks, *Logic Pin Reordering* (LPR), and the second phase targets the routing switches, *Routing Pin Reordering* (RPR). The LPR stage is performed right after synthesis and before the packing stage, while RPR is performed after the routing stage, as shown in Figure 6.7. Again, the CAD flow used in this thesis is based on the VPR CAD flow [28] where the packing is performed using T-VPack and placement and routing is performed using the VPR CAD tool.

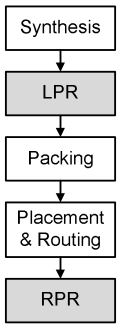


Figure 6.7: VPR CAD flow with the proposed pin reordering algorithms.

6.2.1 Logic Pin Reordering (LPR) Algorithm

The LPR algorithm reorders the input pins in such a way to have the maximum number of signals with similar polarities at the inputs of any multiplexer pair, as shown in Figure 6.5(a). The algorithm also avoids having the input configuration with the highest probability to be the one with all zeros to further minimize leakage power in the LUT, as explained in Section 6.2. Furthermore, the LPR algorithm handles logic blocks differently according to the number of inputs of each logic block. The LPR algorithm is divided into four separate phases as discussed in the next subsections.

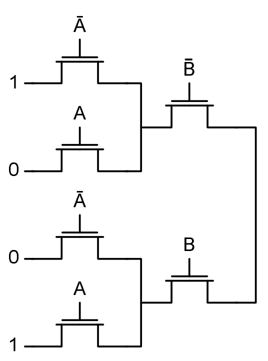
6.2.1.1 Input Pins Padding

In most FPGA CAD tools, whenever a logic block has inputs less than the maximum number of allowable inputs to a logic block, the unused inputs are either left floating or connected to either V_{DD} or GND. The choice whether to connect the unused input pin to either rail does not follow a certain reasoning, but rather it is an architecture choice. In this thesis, a padding methodology is proposed to make use of the fact that multiplexer pairs with similar inputs at both diffusion ends sink less leakage current than those with different inputs, as shown in Figure 6.5.

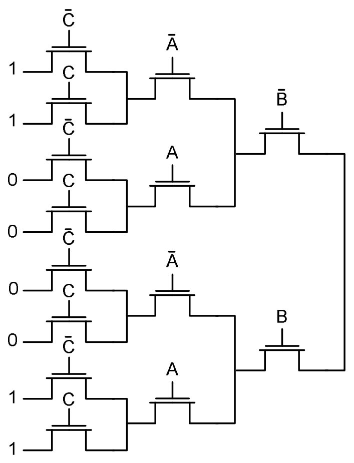
If a logic block has inputs less than the maximum number of allowable inputs, the extra inputs are padded in such a way to create the largest number of low-leakage multiplexer pairs. As an example, assume that the maximum number of inputs for every logic block is three and the logic block shown in Figure 6.8(a) has only two inputs, A and B. The LPR algorithm then pads the extra input C to the circuit, as shown in Figure 6.8(b). C is a fixed signal that can be set to either '1' or '0'. It should be noted that all modern FPGAs have the ability to generate a constant signal from within the logic block with no need to use any extra resources. As a result, the circuit with the padded inputs will have all of its first level multiplexer pairs with identical signals at both inputs, as shown in Figure 6.8(b), hence, maximum leakage reduction can be achieved in this case. It should be noted that the inputs padded into the circuit always go into the least significant bits of the multiplexer.

6.2.1.2 Input Pins Swapping

The second phase of the LPR algorithm is involved with the swapping of the input pins to have the maximum number of multiplexer pairs with similar signals at



(a) Logic block with inputs less than the maximum.



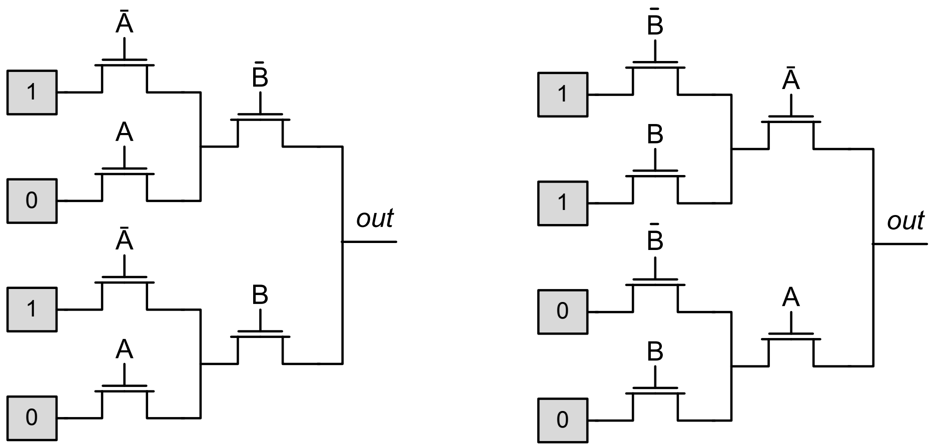
(b) Resulting circuit after input padding.

Figure 6.8: Input padding for logic blocks with inputs less than the maximum.

their inputs. Assume that the inputs to a 4-input logic block are $A_0A_1A_2A_3$. The algorithm picks the first input signal from the synthesized circuit A_0 and looks at the outputs of the logic function implemented when $A_1A_2A_3$ are given by '000', while A_0 is both '0' and '1'. If the two outputs are equal, the counter for A_0 is incremented by 1. Afterwards, the algorithm looks at the outputs of the function when $A_1A_2A_3$ are given by '001', and so on until all the 2^3 different combinations are considered. The same procedure is repeated for the other 3 inputs A_1, A_2, and A_3. The input with the highest count of equal signals is selected as the least significant input pin. The computational complexity of this phase is $O(m \times 2^{m-1})$, where m is the maximum number of inputs to the logic block. A conventional value of m is 4 [28].

As an example of the algorithm operation, consider the 2-input logic block shown in Figure 6.9(a). The count of signals with similar input multiplexers would be zero for A and two for B. Hence, the algorithm would move B to be the least significant bit of the multiplexer instead of A. The resulting logic block with configuration SRAM bits is shown in Figure 6.9(b), where it can be seen that now the first stage multiplexers have signals with similar polarity at their inputs.

After identifying the least significant input, the algorithm tries to find the order of the remaining inputs using the same methodology. However, in finding the least significant input pin, the inputs to the multiplexer are known since they are the contents of the configuration SRAM cells. In the following multiplexer stages,



(a) Logic block before input pins swapping.

(b) Resulting circuit after input swapping.

Figure 6.9: Input pin swapping for logic blocks to minimize leakage power dissipation.

static probability is used to find the most probable value expected at their inputs. If the static probability of the control input to the least significant multiplexer is higher than 0.5, the algorithm assumes that all the values connected to the transistor controlled by the least significant pin will pass to the second stage. On the other hand, if the static probability is less than 0.5, the inputs controlled by the compliment of the least significant input pin are assumed to go through. The same procedure used to select the least significant pin is used to order the remaining input pins.

It should be noted that most of the leakage savings is achieved from the selection of the least significant input pin. This is mainly because the leakage current in the multiplexer will be limited to the smallest leakage current on the path. However, rearranging the pins of the latter stages adds an extra amount of leakage savings by adding more resistance in the leakage current path. It is worth mentioning that this phase of LPR does not add any physical overhead to the design, since it merely rearranges the input pins and the configuration SRAM contents.

6.2.1.3 Most Probable States

As presented in Section 6.1.4, CMOS inverters dissipate almost 2X more leakage power when their output is ‘1’. Since the multiplexers use several inverters to generate the needed input signals to control the multiplexer, then it might be wise to avoid having the most probable input being given by all zeros. Such an input will generate the highest leakage power dissipation in the inverters inside the multiplexer. As a matter of fact, it is more desirable to have the most probable input being all ones.

To make use of this property, the LPR algorithm looks at the static probabilities of the different input combinations to each multiplexer, if the highest one contains a large number of zeros, *i.e.*, more than half the maximum number of inputs, the algorithm tries to avoid that by inverting one or more of the input signals. This can be easily done in FPGAs as it only implies changing the contents of the SRAM cells in the logic blocks that are connected to this signal.

The LPR algorithm looks at the set of inputs that need to be inverted and tries to invert only a small number of them that is needed to counterbalance the effect of the large number of zero inputs. As an example, consider a 4-input logic block that has the most probable input being '1000'. Then the algorithm would need to toggle two of the three least significant inputs to counterbalance the effect of the majority of zeros in the most probable inputs. The toggling is performed by toggling the contents of the configuration SRAM cells in the logic block that generates these signals. The choice of which signals to toggle is based on the probabilities of the next most probable inputs. If a certain signal appears '0' more than once in the first five most probable inputs states, that it has a higher probability of being selected by LPR to be toggled. If the five most probable input states have static probabilities given by $P_{1,2,3,4,5,}$, and the value of the input j in input state i is given by $I_{j,i}$. The inputs selected for toggling are those the maximize

$$\max_{\forall j} \sum_{i=0}^{5} I_{j,i} \times P_i \ . \tag{6.4}$$

6.2.1.4 Unutilized Logic Resources

The unutilized logic resources should always be placed in a low-leakage state to avoid wasting leakage current. From the discussion presented in Section 6.1 and summarized in Figure 6.4(b) and Table 6.1, the lowest leakage state occurs when all the SRAM cells store '0'. In addition, the input to the inverters that generate the control signals inside the logic blocks should be connected to V_{DD}. Since FPGAs have the flexibility to connect any input inside the logic blocks to either of the supply rails, then this phase of LPR guarantees placing the unutilized logic resources in a low-leakage mode without incurring any physical costs.

6.2.2 Routing Switches Pin Reordering (RPR) Algorithm

The routing architecture assumed in this work is the disjoint architecture with a flexibility of three. The RPR algorithm is composed of three phases.

6.2.2.1 Input Pins Padding

The input padding phase of the RPR algorithm is similar to that of the LPR algorithm. If one of the routing multiplexers has one of its inputs left floating, that input is connected to a constant signal in such a way to reduce the leakage power in that multiplexer, according to the guidelines in Section 6.1. Similar to the logic phase, these multiplexers have the flexibility to generate constant signals from within, thus reducing the need for extra hardware. However, the inputs to the drains of the pass-transistors of the routing switches are not known beforehand, unlike the logic blocks which are dictated by the contents of the configuration SRAM cells. This work depends on the static input probabilities of the routed signals to estimate the most probable value of the signal and based on that pad the vacant input signals accordingly to minimize the leakage power dissipation.

6.2.2.2 Most Probable States

Some of the routing resources employed in FPGAs are buffered routing switches, where the output of the multiplexer is connected to a buffer to transmit the signal for a long distance across the FPGA fabric. However, these buffered switches are prone to the input dependency of the leakage power dissipation in the buffers, especially since they are designed with large dimensions. This observation is the core of the earlier work proposed in [53].

The leakage power dissipation in the buffers can be minimized by avoiding the state with the highest leakage. Inside the buffer, both inverters dissipate leakage power dissipation, however, the second inverter dissipates more leakage power because it is designed with a larger size. Consequently, the desired low-leakage state is that of the second inverter. According to that, the low-leakage state is when the input to the buffer is ‘1’. Since the input to the buffer is not previously known, static probability is used to estimate the most probable input to all the buffered routing switches. If the most probable input is not ‘1’, then the input is inverted simply by inverting the values of the logic blocks connected to that net. It is noteworthy that this phase of the RPR algorithm is applied first since it might affect the actions taken by input padding phase. The algorithm used in this work is similar to the one presented in [53].

6.2.2.3 Unutilized Routing Resources

Similar to the LPR algorithm, the RPR is applied to place all the unutilized routing resources in a low-leakage state. This is performed by generating constant signals within the unused routing multiplexers to manipulate them into the lowest leakage state.

6.3 Experimental Results

The proposed leakage power reduction pin reordering algorithm is tested on a 90nm CMOS process using several FPGA benchmarks [28]. The algorithm takes as an input a readily synthesized circuit and then rearranges the inputs to each LUT to result in the minimum leakage power dissipation. The benchmark circuits are synthesized using the SIS sequential circuit synthesis tool [126]. Both the LPR and RPR algorithms are integrated into the VPR CAD flow [28] according to the flowchart shown in Figure 6.7. The leakage power modeling is performed using the proposed power modeling approach presented in Chapter 4 to take the state dependency of leakage power into consideration.

The proposed pin reordering algorithm is applied to several FPGA benchmarks and the percentage leakage savings are listed in Table 6.2 for a 4-input LUT compared to the control case. Moreover, Table 6.2 also lists the total leakage savings achieved by [53]. The total average leakage savings is around 50% across all the benchmarks tested, while that achieved by [53] is around 24.72%. It can be easily shown that the proposed algorithm outperforms that of [53] in terms of leakage savings. In addition, Table 6.2 shows how the leakage savings due to the LPR algorithm vary with the number of inputs to the logic blocks. The larger the number logic blocks with inputs less than 4, the higher the leakage savings due to LPR due to its padding phase.

Figure 6.10 shows a breakdown of the leakage savings due to the LPR phase. It can be shown that the maximum savings are generated by the input swapping phase. Figure 6.10 shows that very small leakage savings originate from placing the unutilized logic into a low-leakage state because the benchmarks tested were mapped into the smallest FPGA square array that can hold them, thus resulting

Table 6.2: Leakage savings by the proposed pin reordering algorithm across several FPGA benchmarks.

Benchmark	Percentage of logic blocks with			Avg Leakage Savings in Logic (%)	Avg Leakage Savings in Routing (%)	Tot. Leakage Savings (%)	Avg Leakage Savings (%) [53]
	2 inputs	3 inputs	4 inputs				
alu4	0.07	0.29	0.62	60.92	40.76	50.3	20.7
apex4	0.01	0.42	0.55	57.34	41.35	48.9	26.5
des	0.05	20	0.74	57.28	45.74	50.96	29.99
dsip	0	0	0.98	20.13	45.72	33.16	20.86
frisc	0.07	0.27	0.64	68.34	42.18	54.61	26.16
pdc	0.01	0.21	0.76	52.07	42.28	47.18	24.5
s298	0.08	0.22	0.68	58.84	42.27	50.12	26.85
s38584.1	0.25	0.18	0.53	79.48	43.79	61.49	16.48
spla	0.01	0.24	0.74	53.78	42.8	48.02	21.2
apex2	0.06	0.31	0.62	56.68	41.51	48.78	30.36
bigkey	0.2	0	0.79	64.08	43.43	53.28	28.69
clma	0.06	0.24	0.69	49.91	42.6	46	17.79
diffeq	0.09	0.29	0.61	67.87	41.32	53.8	27.02
elliptic	0.12	0.28	0.59	60.01	44.07	51.27	27.19
ex5p	0.04	0.21	0.74	40.80	41.27	41.04	36.7
misex3	0.04	0.35	0.59	66.08	39.29	52.2	18.7
s38417	0.04	0.41	0.52	57.10	43.74	50.27	18.83
seq	0.07	0.33	0.59	68.44	41.72	54.2	24.7
tseng	0.12	0.27	0.6	70.58	43.47	56.05	19.18
ex1010	0.04	0.42	53	69.84	41.09	54.09	31.9
			Average	58.98	42.38	50.29	24.72

in the absolute minimum unutilized logic resources. It should be noted that if the benchmarks were mapped into more practical FPGA sizes, the percentage leakage savings in the unutilized resources would increase notably.

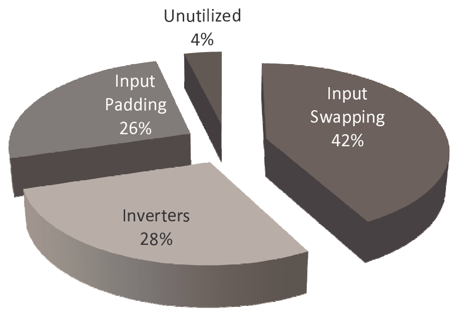


Figure 6.10: Leakage savings breakdown in logic blocks.

The leakage power savings achieved using RPR breakdown is shown in Figure 6.11. It can be noticed that the average savings in the unutilized routing resources is larger than that resulting for the logic resource. This is mainly because the percentage of unutilized routing resources is usually larger than that of the logic resources. Another observation is that the percentage leakage savings in the inverters is larger in the routing resources, which can be justified by the fact that the inverters used in the routing resources are of larger sizes than those used in the logic resources, thus they consume larger leakage.

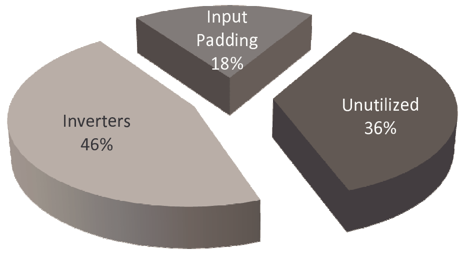


Figure 6.11: Leakage savings breakdown in the routing resources.

6.3.1 Pin Reordering and Performance

The pin swapping algorithm results in changing the V_{TH} of the transistors in the pass transistor multiplexers due to the DIBL and body effects previously explained. Its main aim is to have a net increase in V_{TH} to result in subthreshold leakage savings. As a result, the delay through these multiplexers ends up increasing. In order to investigate the impact of changing the input ordering on the design performance, several SPICE simulations were performed to calculate the average delay through the pass transistor multiplexer for all the possible input combinations. A look-up table is then structured for all the possible delays. To quantify the increase in delay due to the proposed algorithms, the delay across the critical path of the resulting designs are compared to those designed using the regular VPR and the performance penalty is plotted in Figure 6.12. It can be seen that the performance penalty is always less 3% across all benchmarks tested.

In another experiment, the algorithm was applied in such a way to avoid changing the logic blocks and routing resources along the benchmark critical path. The critical path is identified using the state dependent delay look-up table explained above. Logic blocks and routing resources along the critical path are marked as do not touch for the LPR and RPR algorithms not to change them. This version of the algorithm is called No-change for the Critical Path (NCP). The NCP version results in less leakage savings than those recorded in Table 6.2. The leakage savings resulting from the NCP versions are plotted in Figure 6.13 for all the benchmarks

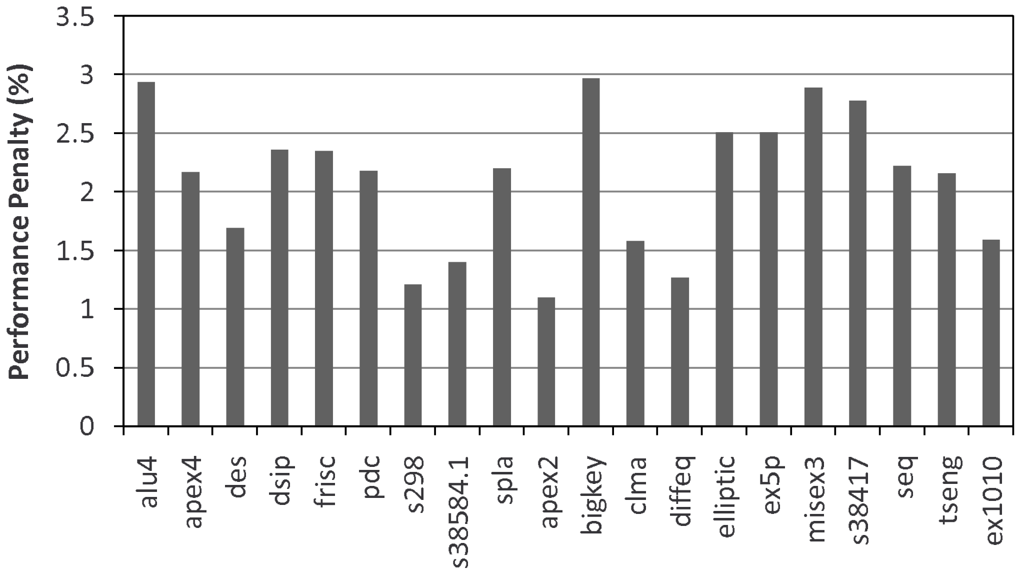


Figure 6.12: Performance penalty due to the proposed algorithm.

tested. The horizontal line represents the average of the leakage reduction, which is around 49.14%, which is slightly less than the average savings in Table 6.2 (50.29%).

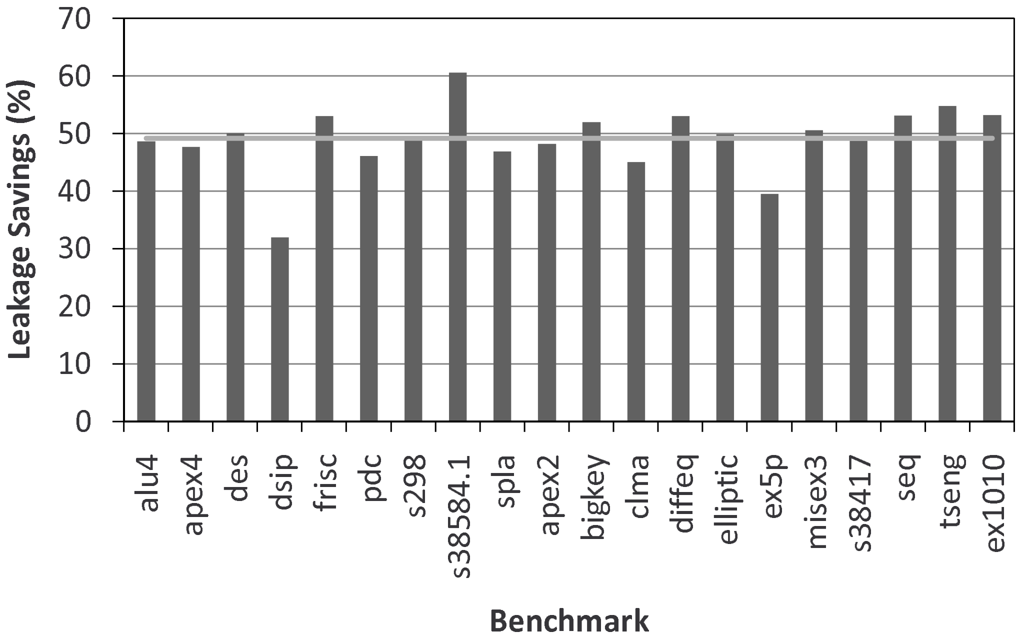


Figure 6.13: Leakage savings to avoid affecting the performance.

In another experiment, instead of leaving the logic and routing resources along the critical path unchanged, the pin swapping algorithm is applied to actually increase the multiplexers V_{th}. As a result, the leakage power along the critical path increases, while the delay along it is reduced. This provides a means of trading off some of the leakage savings to improve the circuit performance. This version of the algorithm is called Reduce the Critical Path (RCP). Figure 6.14 plots the percentage leakage savings as well as the improvement in the design performance for each benchmark achieved by the RCP version. The average performance improvement is 2.4% while the average reduction in leakage savings is 47.7%.

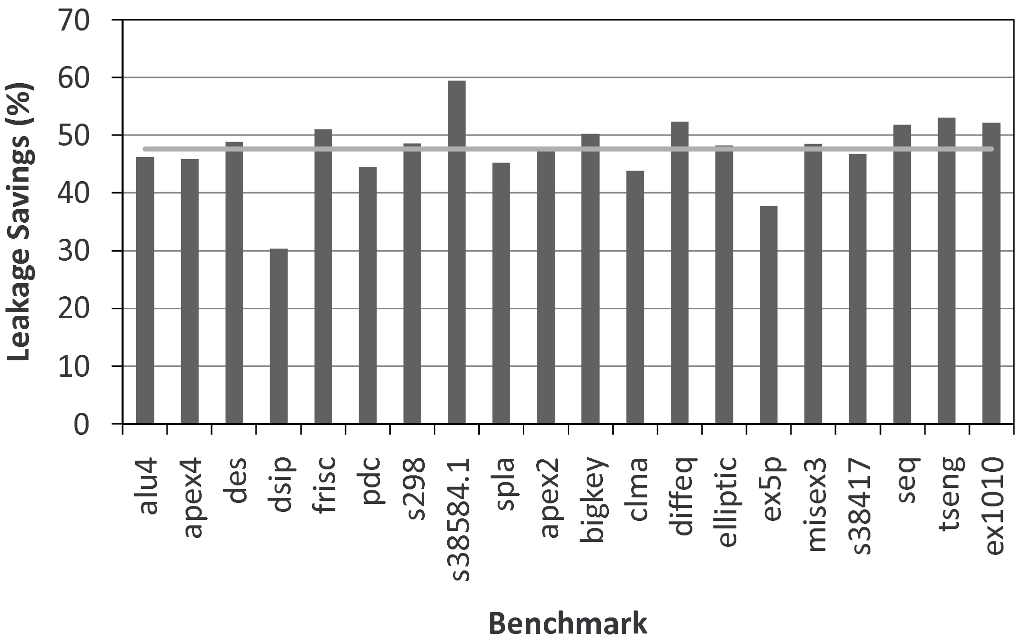
6.3.2 Pin Reordering and Technology Scaling

Based on the ITRS report [1], both types of leakage currents are expected to increase significantly with the technology scaling as shown in Figure 6.15. However, the increase in the gate leakage current is expected to be steeper resulting in gate leakage current exceeding subthreshold leakage current in future CMOS technologies.

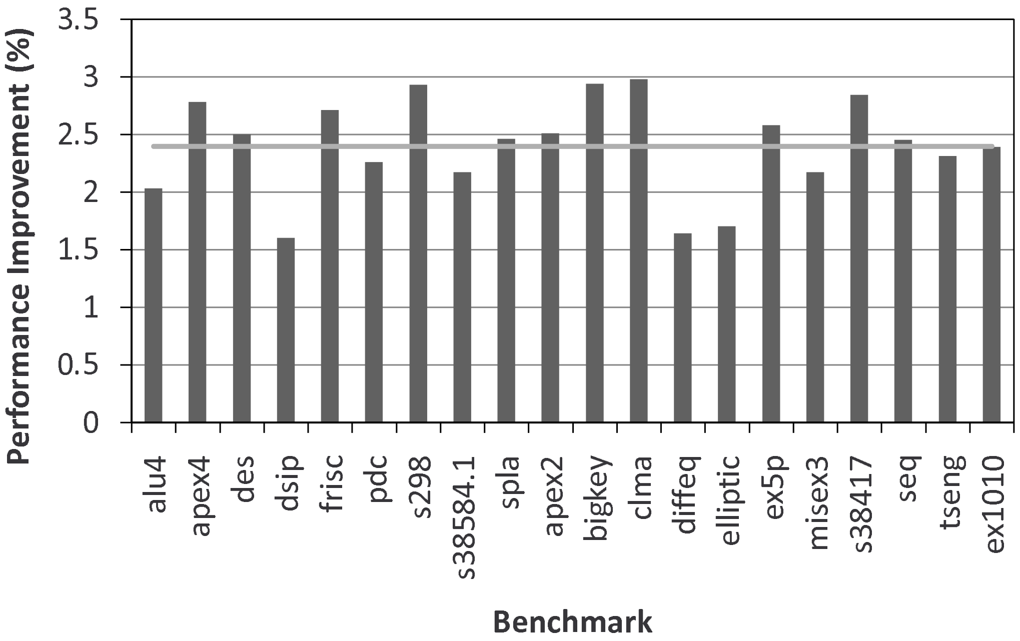
The minimum leakage state depends heavily on the relative magnitude of the subthreshold leakage and gate leakage currents. Hence, it is expected that the minimum leakage states will change for each technology. In the next set of experiments, the minimum leakage state is evaluated for several future technology nodes [124] using the BSIM4 leakage model. The results are presented in Figure 6.16. It can be noticed that the state where the input and the output of the multiplexer are given by ‘0’ is no longer the minimum leakage states for technologies beyond the 90nm. This is because this state has the maximum gate leakage current, so once the contribution of gate leakage current starts to dominate the total leakage current, the total leakage of that state will increase. The next observation is that as the technology is scaled down, the body effect starts to have a smaller effect on the device V_{th}, hence, the total leakage of the state where the input and output are given by ‘1’ starts to increase. Moreover, the gate leakage of that state is the second highest gate leakage current achievable.

6.4 Conclusion

In this Chapter, a leakage reduction algorithm is proposed for FPGAs without any physical or performance penalties by employing the input dependency of leakage power. Input reordering is used to place the logic and routing resources in their lowest leakage state. The proposed methodology targets both the logic and routing



(a) Leakage savings for the RCP version.



(b) Performance improvement for the RCP version.

Figure 6.14: Trading leakage savings to reduce critical path delay in RCP.

resources using the LPR and RPR algorithms, respectively. The newly developed algorithm achieves an average leakage savings of 50%. Another version of the proposed algorithm is also developed that results in a performance improvement of 2.4%, while achieving an average leakage reduction of 47.7%.

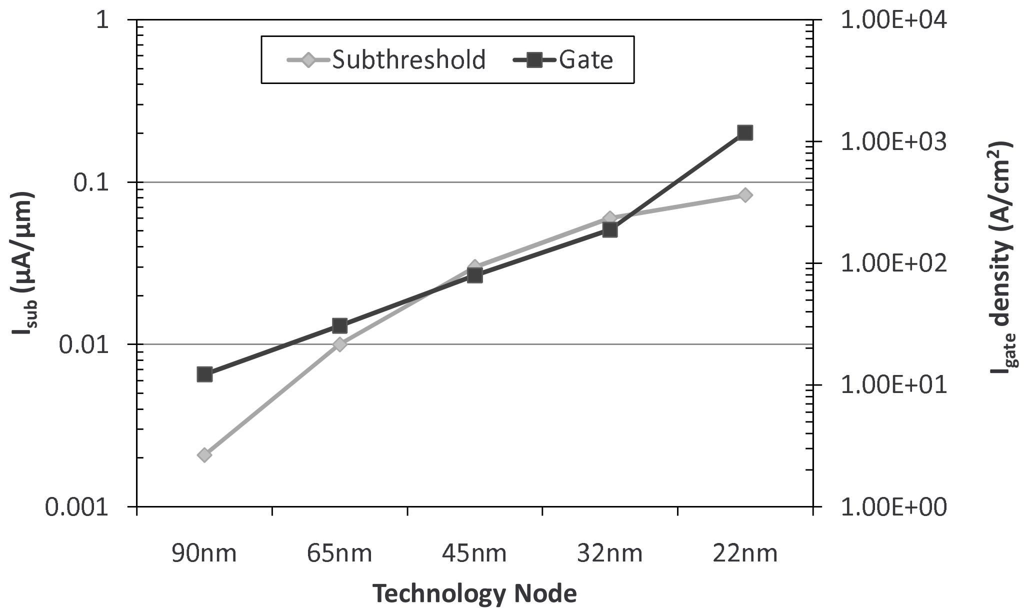
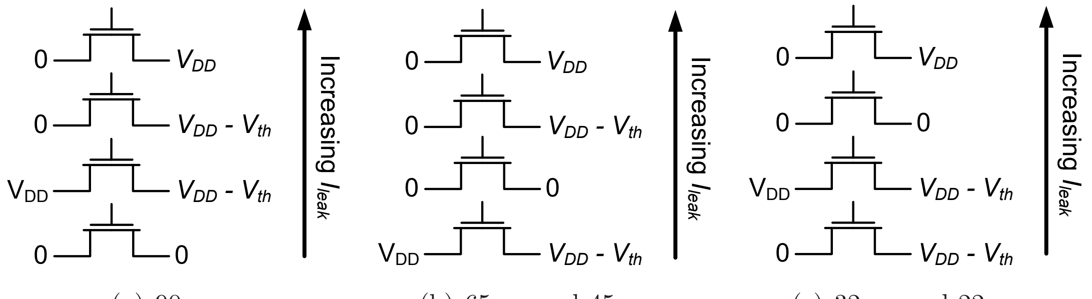


Figure 6.15: Leakage current vs. technology.



(a) 90nm. (b) 65nm and 45nm. (c) 32nm and 22nm.

Figure 6.16: Total leakage dominant states in FPGA pass transistor devices.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis proposed several methodologies for leakage power reduction in modern nanometer FPGAs. The use of supply gating using Multi-Threshold CMOS (MTCMOS) techniques was proposed to enable turning OFF the unused resources of the FPGA, which are estimated to be close to 30% of the total FPGA area. Moreover, the utilized resources are allowed to enter a sleep mode dynamically during run-time depending on certain circuit conditions. Several new activity profiling techniques (CAP, LAP, and R-LAP) were proposed to identify the FPGA resources that will share common idleness periods, such that these resources will be turned OFF together. To increase the benefits from the MTCMOS FPGA architecture, new packing techniques (AT-VPack, FAT-VPack, and T-MTCMOS) were proposed to include the logic blocks activities and pack those with similar activity profile together. The proposed techniques were applied to several FPGA benchmarks, and it was found out that the combination of R-LAP and T-MTCMOS for activity and packing algorithms, respectively, yields the most leakage savings while incurring the minimum performance penalty. On average, the R-LAP and T-MTCMOS combination yields about 52% leakage savings, while the performance loss affecting the critical paths was kept at 3%.

Another techniques proposed in this thesis for leakage power reduction in FPGAs is the pin reordering algorithm. The pin reordering algorithm makes use of the input state dependency of leakage power to place as much as possible of the FPGA circuits in a low leakage mode without incurring and physical or performance penalties. The guidelines for finding the lowest leakage power dissipation mode were derived and it was shown how they vary with every process node depending on the relative magnitude of subthreshold and gate leakage power components. The proposed pin reordering technique was applied to several FPGA benchmarks and resulted in an average of 50% leakage power savings. Furthermore, another version of the proposed algorithm is also developed that results in a performance improvement of 2.4%, while achieving an average leakage reduction of 47.7%.

In order to quantify the amount of power savings achieved by the proposed leakage reduction techniques, an accurate power model for FPGAs is developed. The new power model considers both dynamic and leakage power dissipation. Moreover, glitching power of FPGA designs is also calculated. Furthermore, spatial correlation between design signals is also accounted for to calculate the total power dissipation in FPGAs. The accuracy of the model is within 10% of HSpice simulations while spatial independence approaches are within 25% of HSpice.

7.2 Future Work

While this thesis attempted to provide practical methods for leakage power reduction in modern FPGAs, it provides a road to a bigger project that requires further research. The first goal of any future work would be to apply the proposed MTCMOS design methodology to existing FPGA macros. FPGA vendors provide their users with ready-made designed macros that implement specific functionality, such as adders and multipliers. These macros have pre-known functionality and they can be designed easily using the proposed MTCMOS techniques to provide a low-leakage alternative for power sensitive applications. Along the same line of thought, some new FPGA macros can also be designed that serve a certain consumer market. Examples of these macros include MPEG and JPEG encoders/decoders. Finally, the MTCMOS techniques proposed in this thesis can be modified to be implemented in the ASIC design domain.

A second area for future research is to investigate the impact of parameter variations on the proposed leakage reduction techniques. Parameter variations affect CMOS designs by reducing the final yield. The impact of parameter variations increases as the device feature size is scaled down. For future CMOS technologies, there is a need to design low power design techniques that are immune to parameter variations. It should be noted that leakage power is very sensitive to any small variations in the device parameters. Moreover, future FPGA power models should be able to calculate total power dissipation under the impact of parameter variations.

A third area for possible future research is to develop an analytical methodology to calculate leakage power dissipation inside the FPGA logic blocks under all input configurations. In this thesis, lookup tables resulting from HSpice simulations were used both in the developed power model and pin reordering algorithm. These lookup tables can be replaced by the analytical method to speed up the setup phases of these algorithms while providing portability over as much CMOS technologies as possible.

Bibliography

[1] The International Technology Roadmap for Semiconductors website. [Online]. Available: http://public.itrs.net

[2] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, 1999.

[3] B. Zahiri, "Structured ASICs: Opportunities and Challenges," in *Proc. of Intl. Conf. on Computer Design*, 2003, pp. 404–409.

[4] R. R. Taylor and H. Schmit, "Creating a Power-Aware Structured ASIC," in *Proc. of Intl. Symp. on Low Power Electronics and Design*, 2004, pp. 74–77.

[5] K. J. Han, N. Chan, S. Kim, B. Leung, V. Hecht, B. Cronquist, D. Shum, A. Tilke, L. Pescini, M. Stiftinger, and R. Kakoschke, "Flash-based Field Programmable Gate Array Technology With Deep Trench Isolation," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 2007, pp. 89–91.

[6] S. D. Brown, "An Overview of Technology, Architecture and CAD Tools for Programmable Logic Devices," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 1994, pp. 69–76.

[7] J. Greene, E. Hamdy, and S. Beal, "Antifuse Field Programmable Gate Arrays," *Proc. IEEE*, vol. 81, no. 7, pp. 1042–1056, July 1993.

[8] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2000, pp. 3–12.

[9] J. Rose, R. J. Francis, D. Lewis, and P. Chow, "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," *IEEE J. Solid-State Circuits*, vol. 25, no. 5, pp. 1217–1225, Oct. 1990.

[10] Altera Corp. Stratix III Device Handbook. [Online]. Available: http://www.altera.com/literature/hb/stx3/stratix3_handbook.pdf

[11] Xilinx Inc. Vertix-5 FPGA User Guide. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug190.pdf

[12] Actel Corp. ProASIC3 Handbook. [Online]. Available: http://www.actel.com/documents/PA3_HB.pdf

[13] J. Rose and S. Brown, "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays," *IEEE J. Solid-State Circuits*, vol. 26, no. 3, pp. 277–282, 1991.

[14] J. Cong and M. Smith, "A Parallel Bottom-Up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design," in *Proc. of IEEE/ACM Design Automation Conf.*, 1993, pp. 755–760.

[15] J. Cong, J. Peck, and Y. Ding, "Rasp: A general logic synthesis system for sram-based fpgas," in *Proc. of IEEE/ACM Design Automation Conf.*, 1996, pp. 137–143.

[16] J. Cong, C. Wu, and Y. Ding, "Cut ranking and pruning: Enabling a general and efficient fpga mapping solution," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 1999, pp. 29–35.

[17] A. Ling, D. P. Singh, and S. D. Brown, "Fpga technology mapping: A study of optimality," in *Proc. of IEEE/ACM Design Automation Conf.*, 2005, pp. 427–432.

[18] A. Marquardt, V. Betz, and J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 1999, pp. 37–46.

[19] J. Cong, L. Hargen, and A. B. Kahng, "Random Walks for Circuit Clustering," in *Proc. of IEEE Intl. Conf. on Application Specific Integrated Circuits*, 1991, pp. 14–21.

[20] J. Cong and S. K. Lim, "Edge Separability Based Circuit Clustering with Application to Circuit Partitioning," in *Proc. of IEEE/ACM Asia South Pacific Design Automation Conf.*, 2000, pp. 429–434.

[21] L. W. Hagen and A. B. Kahng, “Combining Problem Reduction and Adaptive Multi-Start: a New Technique for Superior Iterative Partitioning,” *IEEE Trans. Computer-Aided Design*, vol. 16, no. 7, pp. 709–717, July 1997.

[22] D. J.-H. Huang and A. B. Kahng, “When Clusters Meet Partitions: New Density-Based Methods for Circuit Decomposition,” in *Proc. of European Design and Test Conf.*, 1995, pp. 60–64.

[23] A. E. Dunlop and B. W. Kernighan, “A Procedure for Placement of Standard Cell VLSI Circuits,” *IEEE Trans. Computer-Aided Design*, vol. 4, no. 1, pp. 92–98, 1985.

[24] D. J.-H. Huang and A. B. Kahng, “Partitioning-Based Standard-Cell Global Placement with an Exact Objective,” in *Proc. of ACM Intl. Symp. on Physical Design*, 1997, pp. 18–25.

[25] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, “GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization,” *IEEE Trans. Computer-Aided Design*, vol. 10, no. 3, pp. 356–365, Mar. 1991.

[26] A. Srinivasan, K. Chaudhary, and E. S. Kuh, “Ritual : A Performance Driven Placement Algorithm for Small Cell ICs,” in *Proc. of Intl. Conf. on Computer Aided Design*, 1991, pp. 48–51.

[27] A. Marquardt, V. Betz, and J. Rose, “Timing-Driven Placement for FPGAs,” in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2000, pp. 203–213.

[28] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer Academic Publishers, 1999.

[29] C. Sechen and A. Sangiovanni-Vincentelli, “The TimberWolf Placement and Routing Package,” *IEEE J. Solid-State Circuits*, vol. 20, no. 4, pp. 510–522, Apr. 1985.

[30] S. Kirpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, vol. 220, 4598, pp. 671–680, 1983.

[31] R. B. Hitchcock, “Timing Verification and the Timing Analysis Program,” in *Proc. of IEEE/ACM Design Automation Conf.*, 1982, pp. 594–604.

[32] D. Lewis, E. Ahmed, G. Baeckler, V. Betz, M. Bourgeault, D. Cashman, D. Galloway, M. Hutton, C. Lane, A. Lee, P. Leventis, S. Marquardt, C. McClintock, K. Padalia, B. Pedersen, G. Powell, B. Ratchev, S. Reddy, J. Schleicher, K. Stevens, R. Yuan, R. Cliff, and J. Rose, "The Stratix II logic and routing architecture ," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2005, pp. 14–20.

[33] D. Lewis, V. Betz, D. Jefferson, A. Lee, C. Lane, P. Leventis, S. Marquardt, C. McClintock, B. Pedersen, G. Powell, S. Reddy, C. Wysocki, R. Cliff, and J. Rose, "The Stratix^{TM} Routing and Logic Architecture," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2003, pp. 12–20.

[34] G. G. Lemieux and S. D. Brown, "A Detailed Routing Algorithm for Allocating Wire Segments in Field-Programmable Gate Arrays," in *Proc. of ACM Physical Design Workshop*, 1993, pp. 215–226.

[35] Y.-W. Chang, D. F. Wong, and C. K. Wong, "Universal Switch Modules for FPGA Design," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 1, no. 1, pp. 80–101, Jan. 1996.

[36] S. J. Wilton, "Architectures and algorithms for field-programmable gate arrays with embedded memory," Ph.D. dissertation, Univ. of Toronto, Toronto, 1997. [Online]. Available: http://www.eecg.toronto.edu/ jayar/pubs/theses/Wilton/StevenWilton.pdf

[37] M. I. Masud and S. J. Wilton, "A New Switch Block for Segmented FPGAs," in *Proc. of Intl. Workshop on Field Programmable Logic and Applications*, 1999, pp. 274–281.

[38] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns, "Placement And Routing Tools For The Triptych FPGA," *IEEE Trans. VLSI Syst.*, vol. 3, no. 4, pp. 473–482, Dec. 1995.

[39] G. E. Moore, "Lithography and the Future of Moore's Law," in *Proc. Optical/Laser Microlithography VIII*, vol. 2440, 1995, pp. 2–17.

[40] ——, "No Exponential is Forever: But "Forever" Can Be Delayed," in *Dig. of Tech. Papers IEEE Intl. Solid-State Circuits Conf.*, 2003, pp. 2–17.

[41] E. Kusse and J. Rabaey, "Low-Energy Embedded FPGA Structures," in *Proc. of Intl. Symp. on Low Power Electronics and Design*, 1998, pp. 155–160.

[42] V. George, H. Zhang, and J. Rabaey, "The Design of a Low Energy FPGA," in *Proc. of Intl. Symp. on Low Power Electronics and Design*, 1999, pp. 188–193.

[43] J. Lamoureux and S. Wilton, "On the Interaction Between Power-Aware FPGA CAD Algorithms," in *Proc. of Intl. Conf. on Computer Aided Design*, 2003, pp. 701–708.

[44] J. Anderson and F. N. Najm, "Power-Aware Technology Mapping for LUT-Based FPGAs," in *Proc. of IEEE Intl. Conf. on Field-Programmable Technology*, 2002, pp. 211–218.

[45] J. Lamoureux, G. G. Lemieux, and S. J. E. Wilton, "GlitchLess: An Active Glitch Minimization Technique for FPGAs," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2007, pp. 156–165.

[46] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," *Proc. IEEE*, vol. 91, no. 2, pp. 305–327, Feb 2003.

[47] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2006, pp. 21–30.

[48] T. Tuan and B. Lai, "Leakage Power Analysis of a 90nm FPGA," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 2003, pp. 57–60.

[49] F. Li, Y. Lin, L. He, and J. Cong, "FPGA Power Reduction Using Configurable Dual-Vdd," in *Proc. of IEEE/ACM Design Automation Conf.*, 2004, pp. 735–740.

[50] F. Li, Y. Lin, and L. He, "Vdd Programmability to Reduce FPGA Interconnect Power," in *Proc. of Intl. Conf. on Computer Aided Design*, 2004, pp. 760–765.

[51] F. Li, Y. Lin, L. He, and J. Cong, "Low-Power FPGA Using Pre-defined Dual-Vdd/Dual-Vt Fabrics," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2004, pp. 42–50.

[52] N. Azizi and F. Najm, "Look-Up Table Leakage Power Reduction for FPGAs," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 2005, pp. 187–190.

[53] J. Anderson, F. N. Najm, and T. Tuan, "Active Leakage Power Optimization for FPGAs," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2004, pp. 33–41.

[54] J. H. Anderson and F. N. Najm, "Active Leakage Power Optimization For FPGAs," *IEEE Trans. Computer-Aided Design*, vol. 25, no. 3, pp. 423–437, 2006.

[55] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "Reducing leakage energy in fpgas using region-constrianed placement," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2004, pp. 51–58.

[56] S. Gupta, J. Anderson, L. Farragher, and Q. Wang, "CAD Techniques for Power Optimization in Virex-5 FPGAs," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 2007, pp. 85–88.

[57] A. Rahman and V. Polavarapuv, "Evaluation of Low-Leakage Design Techniques for Field Programmable Gate Arrays," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2004, pp. 23–30.

[58] J. Anderson and F. N. Najm, "Low-Power Programmable Routing Circuitry for FPGAs," in *Proc. of Intl. Conf. on Computer Aided Design*, 2004, pp. 602–609.

[59] A. Rahman, S. Das, T. Tuan, and A. Rahut, "Heterogeneous Routing Architecture for LowPower FPGA Fabric," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 2005, pp. 183–186.

[60] A. Kumar and M. Anis, "A Leakage-Tolerant Dual-Vt CAD Flow for FPGAs," *IEEE Trans. Computer-Aided Design*, vol. 26, no. 1, pp. 53–66, Jan. 2007.

[61] F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Trans. VLSI Syst.*, vol. 2, no. 4, pp. 446–455, Dec. 1994.

[62] S. R. Vemuru and N. Scheinberg, "Short-circuit Power Dissipation Estimation For CMOS Logic Gates," *IEEE Trans. Circuits Syst. I*, vol. 41, no. 11, pp. 762–765, Nov. 1994.

[63] Q. Wang and S. B. Vrudhula, "On Short Circuit Power Estimation Of CMOS Inverters," *Proc. of Intl. Conf. on Computer Design*, pp. 70–75, 1998.

[64] E. Acar, R. Arunachalam, and S. R. Nassif, "Predicting Short Circuit Power from Timing Models," in *Proc. of IEEE/ACM Asia South Pacific Design Automation Conf.*, 2003, pp. 277–282.

[65] H. Fatemi, S. Nazarian, and M. Pedram, "A Current-based Method for Short Circuit Power Calculation under Noisy Input Waveforms," in *Proc. of IEEE/ACM Asia South Pacific Design Automation Conf.*, 2007, pp. 774–779.

[66] K. Poon, S. 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, Apr. 2005.

[67] S. M. Kang, "Accurate Simulation of Power Dissipation in VLSI Circuits," *IEEE J. Solid-State Circuits*, vol. 21, pp. 889–891, Oct. 1986.

[68] G. Y. Yacoub and W. H. Ku, "An Accurate Simulation Technique for Short-Circuit Power Dissipation Based on Current Component Isolation," in *Proc. of Intl. Symp. on Circuits and Systems*, 1989, pp. 1157–1161.

[69] T. H. Krodel, "PowerPlay-Fast Dynamic Power Estimation Based on Logic Simulation," in *Proc. of Intl. Conf. on Computer Design*, 1991, pp. 96–100.

[70] F. Rouatbi, B. Haroun, and A. J. Al-Khalili, "Power Estimation Tool for Sub-Micron CMOS VLSI Circuits," in *Proc. of Intl. Conf. on Computer Aided Design*, 1992, pp. 204–209.

[71] Tony D. Givargis, Frank Vahid, and Jörg Henkel, "Trace-driven System-level Power Evaluation of System-on-a-chip Peripheral Cores," in *Proc. of IEEE/ACM Asia South Pacific Design Automation Conf.*, 2001, pp. 306–312.

[72] A. K. Murugavel and N. Ranganathan, "Petri Net Modeling of Gate and Interconnect Delays for Power Estimation," in *Proc. of IEEE/ACM Design Automation Conf.*, 2002, pp. 455–460.

[73] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo Approach for Power Estimation," *IEEE Trans. VLSI Syst.*, vol. 1, no. 1, pp. 63–71, Mar. 1993.

[74] M. G. Xakellis and F. N. Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," in *Proc. of IEEE/ACM Design Automation Conf.*, 1994, pp. 728–733.

[75] A. M. Hill and S.-M. S. Kang, "Determining Accuracy Bounds for Simulation-Based Switching Activity Estimation," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 6, pp. 611–618, June 1996.

[76] R. Marculescu, D. Marculescu, and M. Pedram, "Sequence Compaction for Power Estimation: Theory and Practice," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 7, pp. 973–993, July 1999.

[77] A. Murugavel, N. Ranganathan, R. Chandramouli, and S. Chavali, "Least-Square Estimation of Average Power in Digital CMOS Circuits," *IEEE Trans. VLSI Syst.*, vol. 10, no. 1, pp. 55–58, Feb. 2002.

[78] X. Liu and M. C. Papaefthymiou, "A Markov Chain Sequence Generator for Power Macromodeling," *IEEE Trans. Computer-Aided Design*, vol. 23, no. 7, pp. 1048–1062, July 2004.

[79] M. A. Cirit, "Estimating Dynamic Power Consumption of CMOS Circuits," in *Proc. of Intl. Conf. on Computer Aided Design*, 1987, pp. 534–537.

[80] F. N. Najm, "Transition Density: A New Measure of Activity in Digital Circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, no. 2, pp. 310–323, Feb. 1993.

[81] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," in *Proc. of IEEE/ACM Design Automation Conf.*, 1992, pp. 253–259.

[82] R. Marculescu, D. Marculescu, and M. Pedram, "Efficient Power Estimation for Highly Correlated Input Streams," in *Proc. of IEEE/ACM Design Automation Conf.*, 1995, pp. 628–634.

[83] ——, "Probabilistic Modeling of Dependencies During Switching Activity Analysis," *IEEE Trans. Computer-Aided Design*, vol. 17, no. 2, pp. 73–83, Feb. 1998.

[84] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Efficient Estimation of Dynamic Power Consumption Under a Real Delay Model," in *Proc. of Intl. Conf. on Computer Aided Design*, 1993, pp. 224–228.

[85] R. Marculescu, D. Marculescu, and M. Pedram, "Switching Activity Analysis Considering Spatiotemporal Correlations," in *Proc. of Intl. Conf. on Computer Aided Design*, 1994, pp. 294–299.

[86] J. C. Costa, J. C. Monteiro, and S. Devadas, "Switching Activity Estimation Using Limited Depth Reconvergent Path Analysis," in *Proc. of Intl. Symp. on Low Power Electronics and Design*, 1997, pp. 184–189.

[87] S. Bhanja and N. Ranganathan, "Switching Activity Estimation of VLSI Circuits Using Bayesian Networks," *IEEE Trans. VLSI Syst.*, vol. 11, no. 4, pp. 558–567, Aug. 2003.

[88] K. Weiß, C. Oetker, I. Katchan, T. Steckstor, and W. Rosenstiel, "Power Estimation Approach for SRAM-Based FPGAs," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2000, pp. 195–202.

[89] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic Power Consumption in Virtex^{TM}-II FPGA Family," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2002, pp. 157–164.

[90] V. Degalahal and T. Tuan, "Methodlogy for High Level Estimation of FPGA Power Consumption," in *Proc. of IEEE/ACM Asia South Pacific Design Automation Conf.*, 2005, pp. 657–660.

[91] J. H. Anderson and F. N. Najm, "Power Estimation Techniques in FPGAs," *IEEE Trans. VLSI Syst.*, vol. 12, no. 10, pp. 1015–1027, 2004.

[92] K. Poon, A. Yan, and S. Wilton, "A Flexible Power Model for FPGAs," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2002, pp. 312–321.

[93] F. Li, D. Chen, L. He, and J. Cong, "Architecture Evaluation for Power-Effecient FPGAs," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2003, pp. 175–184.

[94] F. Li, Y. Lin, H. Lei, D. Chen, and J. Cong, "Power Modeling and Characteristics of Field Programmable Gate Arrays," *IEEE Trans. VLSI Syst.*, vol. 24, no. 11, pp. 1712–1724, 2005.

[95] Y. Lin, F. Li, and L. He, "Power Modeling and Architecture Evaluation for FPGAs with Novel Circuits for Vdd Programmability," in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2005, pp. 199–207.

[96] A. Kumar and M. Anis, "An Analytical State Dependent Leakage Power Model for FPGAs," in *Proc. of Design, Automation, and Test in Europe*, 2006, pp. 612–617.

[97] T.-L. Chou and K. Roy, "Statistical Estimation of Sequential Circuit Activity," in *Proc. of Intl. Conf. on Computer Aided Design*, 1995, pp. 34–37.

[98] H. Y. Lui, C. H. Lee, and R. H. Patel, "Power Estimation and Thermal Budgeting Methodlogy for FPGAs," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 2004, pp. 711–714.

[99] Altera Corp. PowerPlay Early Power Estimator. [Online]. Available: http://www.altera.com/literature/ug/ug_stx3_epe.pdf

[100] Xilinx Inc. Xilinx Power Estimator User Guide. [Online]. Available: http://www.xilinx.com/products/design_resources/power_central/ug440.pdf

[101] Actel Corp. IGLOO Power Calculator. [Online]. Available: http://www.actel.com/documents/IGLOOpowercalculator.zip

[102] Altera Corp. PowerPlay Power Analysis. [Online]. Available: http://www.altera.com/literature/hb/qts/qts_qii53013.pdf

[103] Actel Corp. SmartPower v8.2 Users Guide. [Online]. Available: http://www.actel.com/documents/smartpower_ug.pdf

[104] H.-J. Wunderlich, "PROTEST: A Tool for Probabilistic Testability Analysis," in *Proc. of IEEE/ACM Design Automation Conf.*, 1985.

[105] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT - Probabilistic Estimation of Digital Circuit Testability," in *Dig. of Tech. Papers IEEE Intl. Symp. on Fault-Tolerant Comp.*, 1985.

[106] F. Maamari and J. Rajski, "A Reconvergent Fanout Analysis for Efficient Exact Fault Simulation of Combinational Circuits," in *Dig. of Tech. Papers IEEE Intl. Symp. on Fault-Tolerant Comp.*, 1988.

[107] S. Chakravarty and H. B. Hunt, "On Computing Signal Probability and Detection Probability of Stuck-At Faults," *IEEE Trans. Comput.*, vol. 39, no. 11, pp. 1369–1377, Nov. 1990.

[108] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs," in *Proc. of IEEE/ACM Design Automation Conf.*, 1994, pp. 18–23.

[109] Stratix III Device Handbook, Altera Corp. [Online]. Available: http://www.altera.com/literature/hb/stx3/stratix3_handbook.pdf

[110] P. Schneider and S. Krishnamoorthy, "Effects of Correlations on Accuracy of Power Analysis - An Experimental Study," in *Proc. of Intl. Symp. on Low Power Electronics and Design*, 1996, pp. 113–116.

[111] J. Kao and A. Chandrakasan, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits," *IEEE J. Solid-State Circuits*, vol. 35, no. 7, pp. 1009–1018, July 2000.

[112] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V Power Supply High-speed Digital Circuit Technology With Multithreshold-voltage CMOS," *IEEE J. Solid-State Circuits*, vol. 30, no. 8, pp. 847–854, Aug. 1995.

[113] S. Shigematsu, S. Mutoh, Y. M. Y. Tanabe, and J. Yamada, "A 1-V High-Speed MTCMOS Circuit Scheme for Power-Down Application Circuits," *IEEE J. Solid-State Circuits*, vol. 32, no. 6, pp. 861–869, June 1997.

[114] M. Anis, S. Areibi, and M. Elmasry, "Design and Optimization of Multi-threshold CMOS (MTCMOS) Circuits," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 10, pp. 1324–1342, Oct. 2003.

[115] Altera. Stratix Device Handbook, Volume 1. [Online]. Available: http://www.altera.com/literature/hb/stx/stratix_handbook.pdf

[116] Xilinx. Two Flows for Partial Reconfiguration: Module Based or Difference Based. [Online]. Available: http://direct.xilinx.com/bvdocs/publications/xapp290.pdf

[117] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units," in *Proc. of Intl. Symp. on Low Power Electronics and Design*, 2004, pp. 32–37.

[118] A. Rahman, S. Das, T. Tuan, and S. Trimberger, "Determination of Power Gating Granularity for FPGA Fabric," in *Proc. of IEEE Custom Integrated Circuits Conf.*, 2006, pp. 9–12.

[119] S. V. Kosonocky, M. Immediato, P. Cottrell, T. Hook, R. Mann, and J. Brown, "Enchanced Multi-Threshold (MTCMOS) Circuits using Variable Well Bias," in *Proc. of Intl. Symp. on Low Power Electronics and Design*, 2001, pp. 165–169.

[120] H.-O. Kim, Y. Shin, H. Kim, and I. Eo, “Physical Design Methodology of Power Gating Circuits for Standard-Cell-Based Design,” in *Proc. of IEEE/ACM Design Automation Conf.*, 2006, pp. 109–112.

[121] B. Calhoun, F. Honoré, and A. Chandrakasan, “A Leakage Reduction Methodology for Distributed MTCMOS,” *IEEE J. Solid-State Circuits*, vol. 39, no. 5, pp. 818–826, May 2004.

[122] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, “A 90nm Low-Power FPGA for Battery-Powered Applications,” in *Proc. of ACM Intl. Symp. on Field Programmable Gate Arrays*, 2006, pp. 3–11.

[123] D. A. James Kao, Anantha Chandrakasan, “Transistor Sizing Issues and Tool for Multi-threshold CMOS Technology,” in *Proc. of IEEE/ACM Design Automation Conf.*, 1997, pp. 409–414.

[124] The Berkeley Predictive Technology Model website. [Online]. Available: http://www-device.eecs.berkeley.edu/∼ptm/

[125] R. S. Guindi and F. N. Najm, “Design Techniques for Gate-Leakage Reduction in CMOS Circuits,” in *Proc. of IEEE Intl. Symp. on Quality of Electronic Design*, 2003, pp. 61–65.

[126] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, “SIS: A System for Sequential Circuit Synthesis,” Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, Tech. Rep. UCB/ERL M92/41, May 1992.

List of Publications Resulting from this Work

1. H. Hassan, M. Anis, and M. Elmasry, "CAD Techniques for Leakage-Tolerant MTCMOS FPGAs," *IEEE Transactions on VLSI*, submitted for publication.

2. H. Hassan, M. Anis, and M. Elmasry, "Total Power Modeling in FPGAs Under Spatial Correlation," *IEEE Transactions on VLSI*, accepted for publication.

3. H. Hassan, M. Anis, and M. Elmasry, "Input Vector Reordering for Leakage Power Reduction in FPGAs," *IEEE Transactions on CAD*, accepted for publication.

4. H. Hassan, M. Anis, and M. Elmasry, "A Timing Driven Algorithm for Leakage Reduction in MTCMOS FPGAs," in *Proc. ACM Asia and South Pacific Design Automation Conference*, 2007, pp. 678–683.

5. H. Hassan, M. Anis, A. El Daher, and M. Elmasry, "Activity Packing in FPGAs for Leakage Power Reduction," in *Proc. ACM/IEEE Design Automation and Test in Europe Conference*, 2005, pp. 212–217.

6. H. Hassan, M. Anis, and M. Elmasry, "A leakage-aware CAD flow for MTCMOS FPGA architectures," in *Proc. ACM/IEEE International Symposium on Low Power Electronics and Design*, 2005, pp. 257–262.

7. H. Hassan, M. Anis, and M. Elmasry, "Design Methodology and CAD Tools for Nanometer FPGAs: Optimization for Leakage Power," in *SIGDA Ph.D. Forum at the ACM/IEEE Design Automation Conference*, 2008, (abstract - poster).

8. H. Hassan, M. Anis, and M. Elmasry, "Leakage-aware Placement for FPGAs," in *Proc. ACM International Symposium on FPGAs*, 2005, pp. 267 (abstract - poster).