Leakage Power Modeling and Reduction Techniques for Field Programmable Gate Arrays

by

Akhilesh Kumar

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2006

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy the thesis including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

FPGAs have become quite popular for implementing digital circuits and systems because of reduced costs and fast design cycles. This has led to increased complexity of FPGAs, and with technology scaling, many new challenges have come up for the FPGA industry, leakage power being one of the key challenges. The current generation FPGAs are being implemented in 90nm technology, therefore, managing leakage power in deep-submicron FPGAs has become critical for the FPGA industry to remain competitive in the semiconductor market and to enter the mobile applications domain.

In this work an analytical state dependent leakage power model for FPGAs is developed, followed by dual-Vt based designs of the FPGA architecture for reducing leakage power.

The leakage power model computes subthreshold and gate leakage in FPGAs, since these are the two dominant components of total leakage power in the scaled nanometer technologies. The leakage power model takes into account the dependency of gate and subthreshold leakage on the state of the circuit inputs. The leakage power model has two main components, one which computes the probability of a state for a particular FPGA circuit element, and the other which computes the leakage of the FPGA circuit element for a given input using analytical equations. This FPGA power model is particularly important for rapidly analyzing various FPGA architectures across different technology nodes.

Dual-Vt based designs of the FPGA architecture are proposed, developed, and evaluated, for reducing the leakage power using a CAD framework. The logic and the routing resources of the FPGA are considered for dual-Vt assignment . The number of the logic elements that can be assigned high-Vt in the *ideal* case by using a dual-Vt assignment algorithm in the CAD framework is estimated. Based upon this estimate two kinds of architectures are developed and evaluated, homogeneous and heterogeneous architectures. Results indicate that leakage power savings of up to 50% can be obtained from these architectures. The analytical state dependent leakage power model developed has been used for estimating the leakage power savings from the dual-Vt FPGA architectures. The CAD framework that has been developed can also be used for developing and evaluating different dual-Vt FPGA architectures, other than the ones proposed in this work.

Acknowledgments

I would like to acknowledge the invaluable guidance and encouragement received from my research advisor Professor Mohab Anis for the work done in this thesis. I would also like to thank my group members for their suggestions and help.

I am greatly thankful to the readers of my thesis, Professor Mark Aagaard and Professor Shawki Areibi.

Thanks are also due to the technical and administrative staff of the Department of Electrical and Computer Engineering, University of Waterloo.

To my mother, father, sisters, cousins, uncles and aunts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Field Programmable Gate Arrays: Leakage Power Challenge

Digital systems have grown immensely complex with the scaling of technology. The custom VLSI designs have led the growth of high performance digital systems. However, with increasing complexity of designs, the cost and design cycles of custom VLSI designs have increased significantly. FPGAs offer an efficient and cost effective option for implementing digital systems for medium to low volume production. Earlier, FPGAs were being used only for ASIC prototyping, however with increasing logic density and performance the FPGAs are getting embedded in the end user products. Digital system designers can now get the advantages of low time-to-market of the programmable logic in addition to almost ASIC-like logic density. Commercial FPGAs, such as *Stratix* from Altera, and *Virtex* from Xilinx have on-chip memory blocks and DSP resources, apart from the programmable logic, making it even more attractive for implementing complete systems on chip.

Leakage power has been recently recognized as a major challenge in the FPGA industry. This was primarily because other design challenges, such as performance and area, were given more attention in the past. With technology scaling, leakage power has emerged as a key design challenge. The current generation of FPGAs are being implemented in the 90nm CMOS technology, which necessitates devising techniques for leakage power reduction, because leakage power increases with small geometries. For the FPGAs to continue to retain its semiconductor market and competitive advantages over the high performance custom VLSI designs, the FPGA industry must adopt new techniques for leakage power reduction. The work in [4] showed that a 90nm FPGA con-

sumes too much leakage power to be successfully used in mobile applications. Therefore, for FPGAs to gain popularity in the domains such as wireless personal communication system, or in the biomedical applications, the FPGAs need to implement techniques for reducing leakage power.

The increase in complexity of the current generation FPGAs has resulted in more number of transistors in the FPGAs which directly translate into increased leakage power. The resource utilization of FPGAs is just over 60%, and the unutilized parts also consume leakage power, which means that reducing leakage power is important both in the used and the unused parts of the FPGA. The problem of leakage power in FPGAs is more difficult to handle than in ASICs because of the very nature of programmability of FPGAs, which means that the final application which would run on the FPGA is unknown. Motivated by the above challenges, this work contributes to leakage power management in FPGAs as outlined in the next section.

1.2 Contributions of this Work

1. **Analytical Models for Total Leakage Power for FPGAs:** In this work analytical models for leakage power calculation for FPGAs have been developed. The leakage power models incorporate BSIM4 models to compute the subthreshold leakage and gate leakage. These leakage power models have been used for computing leakage currents through the various FPGA circuit elements.

2. **State Dependency for Leakage Power Calculation in FPGAs:** The leakage power model takes state dependency of subthreshold and gate leakage during the computation of these leakage currents.

3. **CAD framework for dual-Vt FPGA designs:** A dual-Vt FPGA CAD framework for designing, developing, and evaluating dual-Vt FPGAs has been proposed. VPR and T-Vpack [9], the widely used academic research tools for FPGA, have been used for developing the dual-Vt FPGA CAD framework.

4. **Dual-Vt FPGA Architectures:** Based on the dual-Vt FPGA CAD framework, dual-Vt FPGA architectures are proposed, developed and evaluated using the dual-Vt FPGA CAD framework. These architectures are intended for reducing leakage power consumption without severe delay penalties.

1.3 Organization of the Thesis

This thesis has been organized as follows.

Chapter 2 gives an overview of the FPGA architecture. It discusses the logic block structure and the routing resources and outlines a general SRAM based FPGA architecture that has been used in this work. It also gives a brief description of the CAD tools used for implementing digital circuits on FPGAs.

Chapter 3 discusses the leakage power in FPGAs. It gives an overview of power dissipation in FPGAs and discusses previous work on leakage power reduction in FPGAs.

Chapter 4 discusses the work on the analytical, state dependent leakage power model for FPGAs. It describes the BSIM4 subthreshold and gate leakage equations and explains the state dependency of the subthreshold and gate leakage in the FPGA circuit elements. It describes the overall framework used for computing the total leakage power in FPGAs, using a Leakage Computation Engine (LCE). Finally results are presented for various MCNC benchmarks.

Chapter 5 explains the dual-Vt FPGA CAD framework that has been developed, followed by a description of the proposed dual-Vt FPGA architectures. It explains various algorithms modified, developed and used in the CAD framework. The different stages of the CAD framework have been discussed in detail and compared with the traditional CAD framework for implementing a digital circuit on FPGA. It presents the results of leakage power savings and design tradeoffs for various dual-Vt FPGA architectures using the dual-Vt FPGA CAD framework. Guidelines for designing dual-Vt FPGA architectures are also presented.

Chapter 6 concludes the thesis and outlines the future work for leakage power management in FPGAs.

Chapter 2

Overview of FPGA Architecture and CAD Tools

2.1 FPGA Architecture

A basic FPGA is shown in Fig. 2.1. The FPGA architecture is very regular in structure. It is made up of two main components - logic blocks (CLBs) and routing resources. The logic blocks implement the functionality of the given circuit while the routing resources provide the connectivity for implementing the logic. The logic blocks have the flexibility to connect to the routing resources surrounding them. The logic blocks and the routing resources are configurable, so that they can be programmed to implement any logic. Though many types of architectures have been experimented with, the most popular one is the SRAM based architecture which is described below and has been used in this work [9].

2.1.1 Logic Block

The logic block of the SRAM based FPGA is LUT (look-up-table) based and are composed of basic logic elements (BLE). LUT is an array of SRAM cells to implement a truth table. Fig. 2.3 shows a two input LUT. It has 4 SRAM cells and a multiplexer to select one of the SRAM cells. The selection is done by the two select signals to the multiplexer, which serve as inputs to the truth-table. Each BLE consists of a k-input LUT, flip-flop and a multiplexer for selecting the output either directly from the output of LUT or the registered output value of the LUT stored in the flip-flop. Fig. 2.4 shows the basic logic element. Previous works have shown that the 4-input LUT is the most

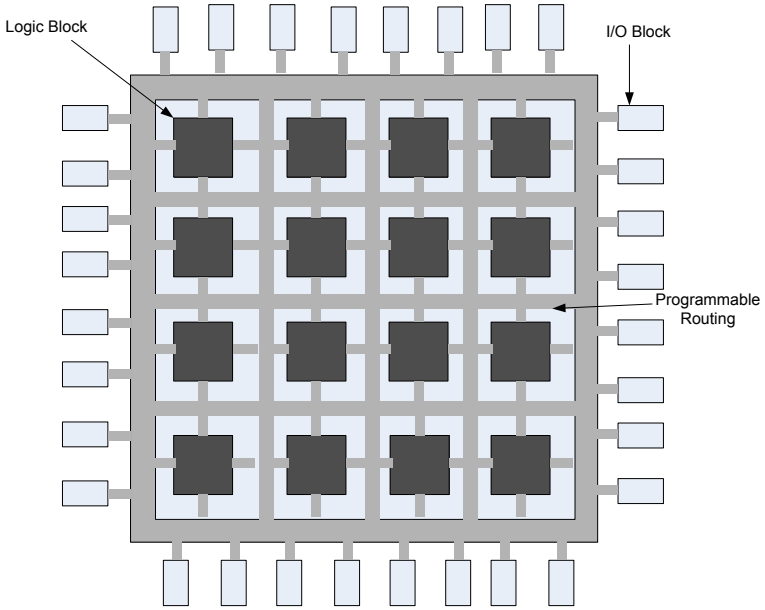

Figure 2.1: A basic FPGA

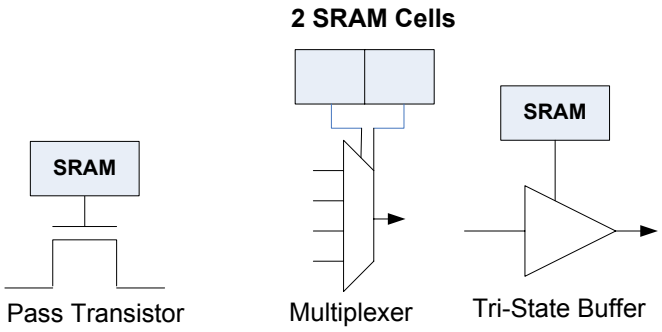

Figure 2.2: Programmable switches used in SRAM-based FPGAs

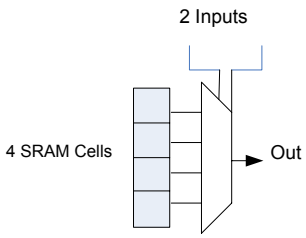

Figure 2.3: A 2-input LUT

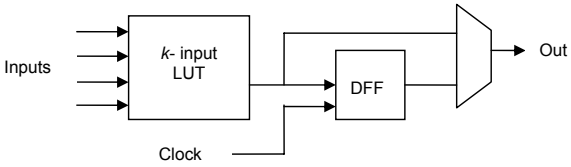

Figure 2.4: Basic Logic Element [9]

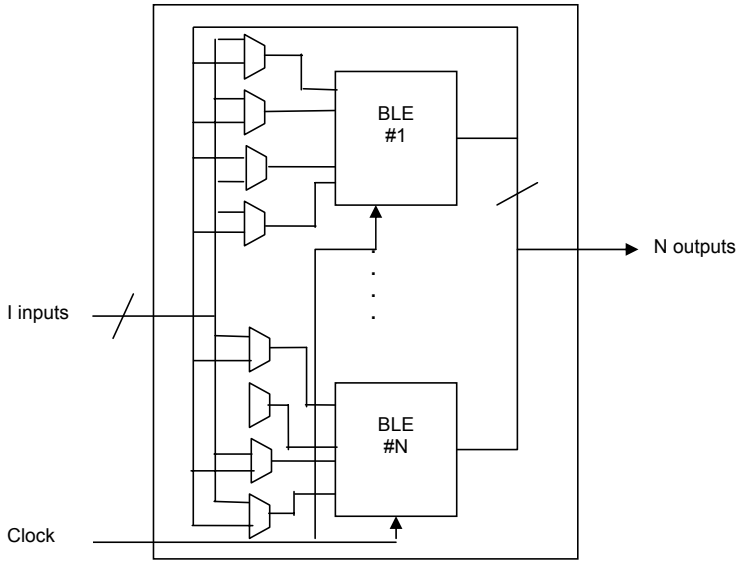

Figure 2.5: Cluster based logic block [9]

optimum size as far as logic density, and utilization of resources are concerned, and this has been widely used. Cluster based logic blocks were investigated in [9] and it was shown that the cluster based logic blocks are better in speed and area. The structure of a cluster based logic block is shown in Fig. 2.5. In the cluster based logic block, the logic block is made up of N BLEs. There are I inputs to the logic block such that each input can connect to all the BLEs. Also the output of each BLE can drive one of the inputs of each of the BLEs. The clock feeds all the BLEs. The work in [9] showed that the logic clusters containing 4 to 10 BLEs achieve good performance. Each subblock is made up of a BLE and the corresponding LUT input multiplexers.

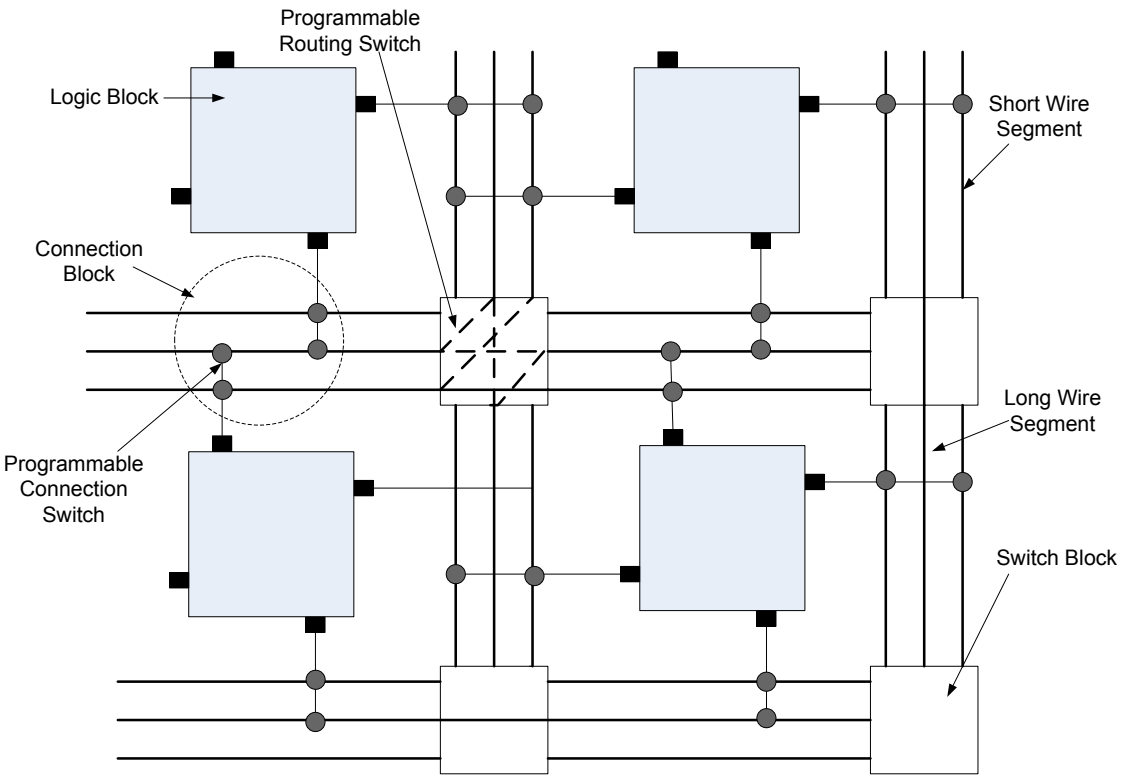

Figure 2.6: Island style routing architecture [9]

2.1.2 Routing Resources

The routing resources are of various types, but the one used in this work is the island-based architecture. In the island based architecture, the routing resources form a mesh like structure with the horizontal and vertical routing channels. The horizontal and vertical routing channels are connected by switch boxes which are programmable and thus provide the flexibility in making the connections. The logic blocks are connected to the routing channels through the connection boxes which are also programmable. Fig. 2.6 shows the island style routing architecture [9]. The programmable switches used for implementing the interconnections are shown in Fig. 2.2. These programmable switches have SRAM cells which can be programmed to either turn on or turn off the switch. Apart from the logic blocks and the routing resources, the clock distribution is assumed to have a dedicated network. Most of the commercial FPGAs have a structure similar to the one described above or some variant of the above architecture.

2.1.3 I/O Blocks

The I/O blocks are also programmable so that they can be configured either as input or as output, or can be tri-stated.

2.2 CAD Tools

To implement a circuit on the current generation FPGAs, CAD tools are needed which can generate the configuration bits for the SRAM cells of the FPGAs. Usually the circuit description is provided using Verilog, VHDL, SystemC, or other higher level descriptions. The CAD tools for the FPGAs read this input and output a configuration file for programming the FPGA. Fig. 2.7 shows the basic CAD flow for implementing a digital circuit/system on FPGAs [9]. The CAD flow has three main tasks: Synthesis, placement and routing. In the following sections synthesis, placement and routing for FPGAs are explained. Since VPR and T-Vpack have been used in this work, the discussion will be kept in context of these CAD tools. Almost all of the commercial FPGA CAD flows perform the same basic functions of synthesis, placement and routing.

2.2.1 Synthesis

The synthesis of a netlist involves conversion of a circuit description, usually in hardware description language (HDL), into a netlist of basic gates. This netlist of basic gates is

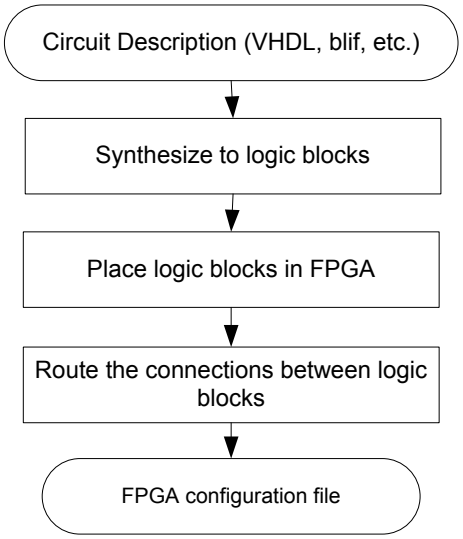

Figure 2.7: Basic CAD flow for FPGAs

then converted into a netlist of FPGA logic blocks. Fig. 2.8 shows the steps involved in the synthesis of a circuit description into a netlist of logic blocks.

Technology independent logic optimization involves the removal of redundant logic and simplification of the logic [27] [28]. The optimized netlist is then mapped to look-up tables, which is a process of identifying the logic gates that would go into a LUT [21]. The final step of the synthesis procedure is the clustering of the LUTs and flip-flops (for sequential logic) into logic blocks. The goal here is usually to minimize the number of logic blocks and/or minimize the delay. The work in [29] used a measure of closeness of LUTs to pack them into a cluster to form a logic block.

The work in [9] uses a timing driven logic block packing tool, called T-VPack. The tool targets packing the BLEs into a cluster shown in Fig. 2.5. It needs the parameters such as number of BLEs per cluster, number of inputs per cluster, size of the LUTs, and number of clocks per cluster. The first stage of the packing procedure simply forms the BLEs by packing a register and a LUT together. Initially the packing procedure packs the BLEs greedily into a cluster, followed by a hill climbing phase if the greedy phase is not able to fill the cluster completely.

To enable a timing driven packing, it is necessary to get an estimate of delays through various paths of the netlist. To enable this computation three types of delays are modeled: delay through a BLE, *LogicDelay*, delay between blocks in the same cluster, *IntraClusterConnectionDelay*, and the delay between blocks in different clusters, *InterClusterConnectionDelay*. The values for these are set as 0.1, 0.1 and 1.0 for *LogicDelay*,

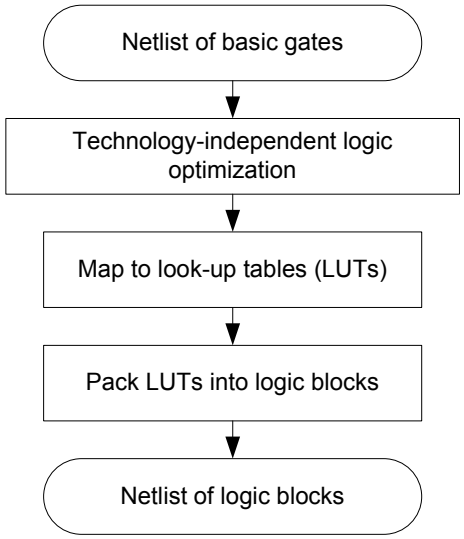

Figure 2.8: Synthesis procedure for FPGAs

IntraClusterConnectionDelay and *InterClusterConnectionDelay*, respectively. The *InterClusterConnectionDelay* cannot be determined until the circuit has been implemented on the FPGA. However, these values represent the correct trend of values, and the performance of T-Vpack is not very sensitive to the exact values. The criticality of a connection is defined as

$$ConnectionCriticality(i) = 1 - \frac{slack(i)}{MaxSlack} \tag{2.1}$$

where $MaxSlack$ is the largest slack amongst all the connections in the circuit. A new cluster is created by selecting a seed BLE having the highest criticality amongst the un-clustered BLEs. After the seed BLE has been selected, an attraction function is used to determine the next un-clustered BLE, B, to be added to the current cluster C. The attraction function is given by:

$$Attraction(B) = \alpha.Criticality(B) + (1 - \alpha)\left[\frac{Nets(B) \cap Nets(C)}{MaxNets}\right] \tag{2.2}$$

where the first term represents the timing part, and the second term represents the cost of nets shared between the current cluster and the BLE under consideration. Any value of α between 0.4 and 0.8 gives good results. The computation of *Criticality* of a

BLE is explained in [9] and also the tie-breaker mechanism used for the case when two or more BLEs have the same criticality. Essentially, the tie-breaker mechanism selects that BLE which reduces the length of the largest number of critical paths.

The hill-climbing phase tries to add more BLEs to the cluster in case it is not full. In this phase adding a BLE to a cluster is allowed even if it leads to more inputs required for the cluster than the maximum allowable. This is done because in some cases the BLE being added might have all its inputs from the BLEs in the current cluster and also might drive the inputs of some of the BLEs in the current cluster. In this case the number of inputs required for the cluster decreases by one. However, this hill climbing phase increases the logic utilization only by 1 - 2% in some of the circuits.

2.2.2 Placement

The work in [9] developed the tool VPR for placement and routing. For placement the FPGA is considered as a set of legal discrete positions at which the logic blocks of the synthesized netlist can be placed. For placement, the architecture descriptions needed by VPR are:

1. The number of logic block input and output pins.
2. The number of I/O pads that fit into one row or column of the FPGA.
3. The routing channel width (number of tracks in a routing channel).

The placement technique used in VPR is based on simulated annealing [30], which imitates the annealing process used to gradually cool a molten metal to produce high quality metal objects. The simulated annealing works by first starting with an initial random placement by placing the logic blocks randomly on the available locations in the FPGA. The placement then proceeds by making a large number of moves to improve the placement. This is done by selecting a logic block randomly and its new location also randomly. This would produce a change in the cost function, and if the cost function improves, the move is always accepted. However, if the cost function worsens, there is still some probability of acceptance of the move. The probability of acceptance is given by $e^{-\triangle C/T}$, where $\triangle C$ is the change in the cost function and the goal is to decrease the cost function. The T is the temperature parameter and controls the probability of acceptance of the moves which worsen the placement. The temperature is initally set to a high value so that at the beginning of the annealing, virtually all the moves are accepted. The temperature is gradually decreased as the placement improves, such that finally the probability of accepting a bad move is almost negligible. The flexibility of accepting bad

moves allows the simulated annealing schedule to overcome the local minima in the cost function.

The VPR sets the initial temperature in the same way as in [31]. The number of moves attempted at each temperature is done as in [32]. It is set to

$$MovesPerTemperature = InnerNum.(N_{blocks})^{4/3} \tag{2.3}$$

where the default value of $InnerNum$ is 10, and N_{blocks} is the number of logic blocks in the netlist. The fraction of moves accepted is kept close to 0.44 for as long as possible, as it yields better results [32]. However, VPR uses a new method of updating the temperature. The VPR computes the new temperature as $T_{new} = \gamma.T_{old}$, where the value of γ depends on the fraction of moves accepted at T_{old}. The idea is to spend maximum time near the temperatures at which large improvements in placement occur. The annealing procedure is not very sensitive to the exact value of γ, if it has the right form, γ is close to 1 if the fraction of moves accepted is close to 0.44, whereas γ is small if the fraction of moves accepted is near 1 or 0. VPR has a timing driven placement and uses a cost function to optimize both the timing and the delay. The complete timing driven placement algorithm is explained in detail in [33]. The cost function for the timing driven placement developed in [33] is given by

$$\triangle C = \lambda.\frac{\triangle TimingCost}{PreviousTimingCost} + (1-\lambda).\frac{\triangle WiringCost}{PreviousWiringCost} \tag{2.4}$$

where $\triangle TimingCost$ and $\triangle WiringCost$ represent the change in the timing cost and the change in the wiring cost, respectively, due to a move. The simulated annealing procedure is terminated when

$$T < \epsilon.\frac{Cost}{N_{nets}} \tag{2.5}$$

where N_{nets} is the total number of nets in the circuit and the value of ϵ is set as 0.005.

2.2.3 Routing

The routing of the placed netlist, essentially, determines the switches that need to be turned on in the routing resources of the FPGA. The routing algorithm in VPR is based on the Pathfinder algorithm proposed in [34]. The Pathfinder repeatedly rips-up and re-routes every net in the circuit until all congestion is resolved. One routing iteration involves ripping-up and re-routing every net in the circuit. The first routing iteration routes for minimum delay, even if it leads to congestion, or overuse of routing resources.

To remove this overuse another routing iteration is performed. The cost of overusing a routing resource is increased after every iteration. This improves the chance of resolving the congestion. At the end of each routing iteration all the nets are completely routed, although with some congestion. Based on this routing, timing analysis can be done to compute the critical path and also the slack of each source sink connection. The timing driven router uses an Elmore delay model to compute the delays of all the connections. The criticality of a connection beteen source of net i and one of its sink j is computed as follows:

$$Crit(i,j) = max\left(\left[MaxCrit - \frac{slack(i,j)}{D_{max}}\right]^{\eta}, 0\right) \tag{2.6}$$

where $slack(i,j)$ is the the slack available to the connection and D_{max} is the delay of the critical path. $MaxCrit$ and η are the parameters which determine how the slack impacts the congestion delay tradeoff in the cost function. In VPR η is set to 1 and $MaxCrit$ is set to 0.99.

The VPR creates a routing resource graph to describe the FPGA architecture and connectivity information. The wire and the logic block pins of the FPGA are represented as nodes in the routing resource graph, and the switches are represented as directed edges in the graph. This routing resource graph is used to perform the routing.

The routing of a net is done by starting with a single node in the routing resource graph, corresponding to the source of the net. A wave expansion algorithm is invoked (k-1) times to connect the source to each of the net's (k-1) sinks, in order of the criticality of the sinks, the most critical sink being the first. The cost for using a node n during this expansion is given by:

$$Cost(n) = Crit(i,j).delay(n, topology) + [1 - Crit(i,j)].b(n).h(n).p(n) \tag{2.7}$$

where $b(n)$, $h(n)$ and $p(n)$ are the base cost, historical congestion, and present congestion as explained in [9]. This procedure is repeated for each of the nets to get the complete routing.

2.3 VPR and T-VPack

This section describes the FPGA CAD tools used in this work. The CAD tools used in this work are VPR, for placement and routing, and T-VPack for clustering of the BLEs [9]. VPR is invoked on the command line as follows [40]

$$vpr\ netlist.net\ architecture.arch\ placement.p\ routing.r\ [-options] \tag{2.8}$$

where $netlist.net$ is the circuit description providing the information about the connectivity of the logic blocks, $architecture.arch$ is the architecture file which describes the architectural parameters of the FPGA. The output of the final placement is written in $placement.p$, or, if the circuit is only being routed, the placement information is read from the file $placement.p$. The final routing information is written in $routing.p$. VPR has two basic modes of operation. In the first mode, VPR places a circuit on the FPGA and routes it for minimum routing channel width. In the other mode, when the user specifies the routing channel width, VPR attempts to route the circuit only once and if it is un-routable it simply aborts, reporting that the circuit is un-routable. The VPR also provides graphics which shows the actual placement and routing of the logic blocks, along with the routing switches.

T-VPack reads a netlist in the *blif* (Berkeley Logic Interchange Format) format having look-up tables (LUTs) and flip-flops (FFs) and packs these into logic blocks. The output of the T-Vpack is in the $.net$ format, which is a netlist of logic blocks. T-VPack is invoked on the command line by

$$t - vpack\ input.blif\ output.net\ [-options] \tag{2.9}$$

where options are used to specify the size of the LUTs, cluster size, inputs per cluster and various optimization options.

The complete VPR CAD flow is shown in Fig. 2.9. SIS [20] is used for logic optimization of the circuit. FlowMap [21] is used for technology-mapping to 4-LUTs and flip-flops. FlowMap produces an output in the $.blif$ format. T-VPack packs the netlist into logic blocks and produces an output in the $.net$ format. VPR is then used for the placement and routing of the netlist. Other logic optimizers and technology mappers, instead of SIS and FlowMap can also be used in this CAD flow.

2.4 Summary

This chapter discussed the island based FPGA architecture used in this work. The logic blocks, programmable switches and routing resources were described. An overview of the CAD tool based on VPR was given. This CAD tool is used for implementing a circuit on the FPGA. Synthesis, placement and routing techniques were discussed.

The next chapter discusses the leakage power mechanisms and the related work in leakage power modeling and reduction techniques for FPGAs.

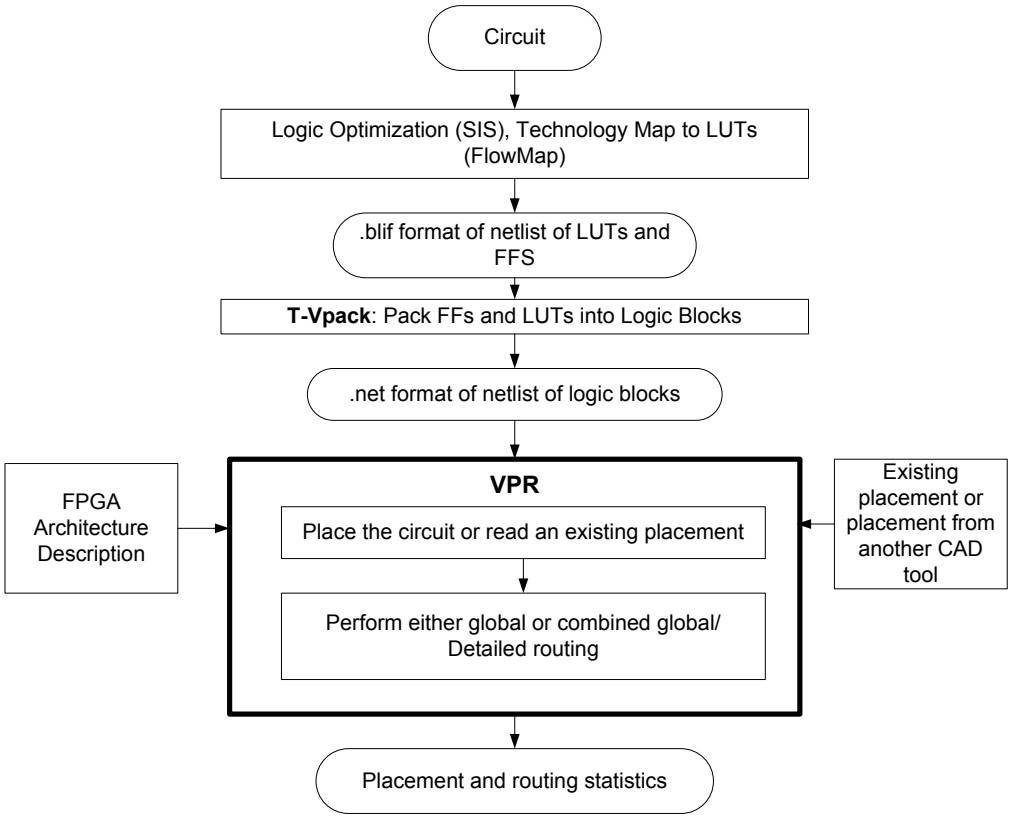

Figure 2.9: VPR CAD flow

Chapter 3

Leakage Power in FPGAs: Background and Related Work

3.1 Introduction

There are two sources of power dissipation in a CMOS circuit: dynamic power and static power. The dynamic power dissipation has three components: Switching power, short circuit power and glitching power. The switching power is due to the charging and discharging of the node capacitances in the circuit. The average switching power dissipation is given by

$$P_{dyn} = \frac{1}{2}.V_{dd}^2.\sum_i C_i.A_i \tag{3.1}$$

where V_{dd} is the supply voltage, A_i is the activity of the node i, C_i is the capacitance of the node i. The short circuit power is due to the transient current between V_{dd} and ground during logic transition. It is around 10% of the switching power. The glitching power is due to spurious transitions during the logic evaluation in the circuit and is caused primarily because of unbalanced path delays.

The static power dissipation in a CMOS circuit is due to leakage current. The static power dissipation is given by

$$P_{leak} = V_{dd}.I_{leak} \tag{3.2}$$

where I_{leak} is the leakage current in the circuit. The leakage power is discussed in detail in the next section.

Dynamic power management in FPGAs was given more importance earlier , because the dominant component of total power was dynamic power. The work in [8] evaluated

different architectural parameters for designing a power efficient FPGA for reducing both the dynamic and leakage power. The work in [38] used a clustering technique for reducing the dynamic power. The work in [36] reduced dynamic power by optimizing the interconnect architecture and circuit design, and by reducing the voltage swing for the interconnects. The work in [37], used a dual-voltage scheme for operating pass transistor networks at low voltage for reducing the dynamic power. The work in [39] used a programmable dual-Vdd technique for reducing the dynamic and leakage power. The work in [43] develops a power-aware technology mapping for LUT based FPGAs to keep the high switching activity nets out of the FPGA routing network, because the routing network has a high capacitance and leads to increased switching power. The work in [42] reduces switching power by optimizing the technology-mapping, clustering, placement and routing stages of the VPR CAD flow, by taking into account the activity of the nets in each stage of the CAD flow.

3.2 Leakage Power

In this section, leakage mechanisms and the impact of technology scaling on leakage power is discussed. The leakage power of a state of the art 90nm FPGA is also discussed.

3.2.1 Technology Scaling and Leakage Power

Rapid scaling of technology was targeted to increase the performance and logic density. Fig. 3.1 shows the technology scaling trend projected by ITRS [46]. There has been an improvement of more than 30% in the delays of the transistors per technology generation. With this, the supply voltage has been scaling and also the threshold voltage (V_{th})of the transistor, so that sufficient gate overdrive is maintained. This has resulted in significant increase in subthreshold leakage as shown by the following equation

$$I_{sub} = I_0 \left[1 - exp\left[\frac{-V_{ds}}{V_T}\right] \right] . exp\left[\frac{V_{gs} - V_{th} - V_{off}}{nV_T} \right] \tag{3.3}$$

where V_T is the thermal voltage, V_{off} is the offset voltage which determines the channel current at $V_{gs} = 0$, n is the subthreshold swing coefficient, $W, L, \mu, q, \phi_s, \epsilon_{si}$, are the width, length, mobility of charge carriers, electron charge, surface potential and permittivity of silicon, respectively, for the transistor. It can be seen that the subthreshold leakage is exponentially dependent on the threshold voltage, V_{th}, of the transistor.

Fig. 3.2 shows the active and leakage power trends for the Intel's process technologies [41]. The leakage power for the 0.25μm technology is 0.1% of the total power,

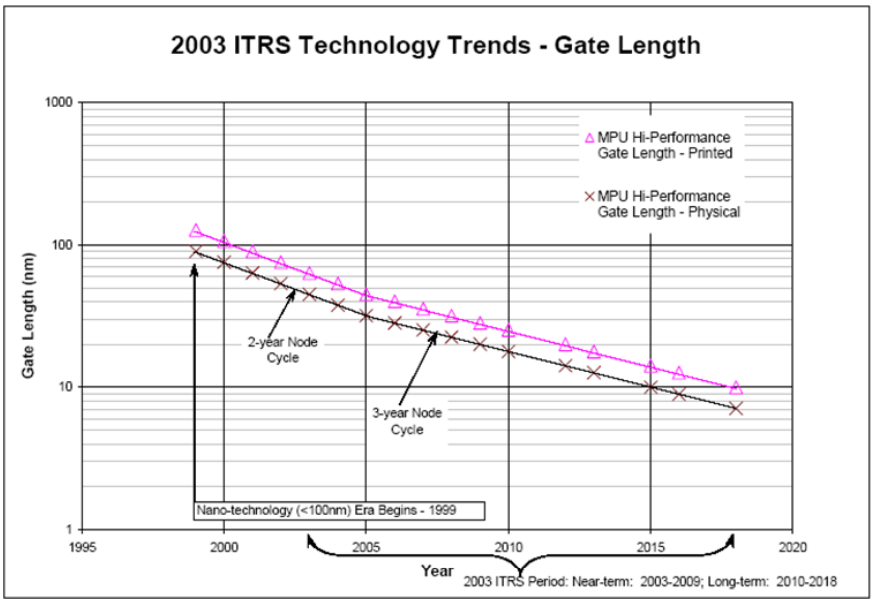

Figure 3.1: Technology Scaling

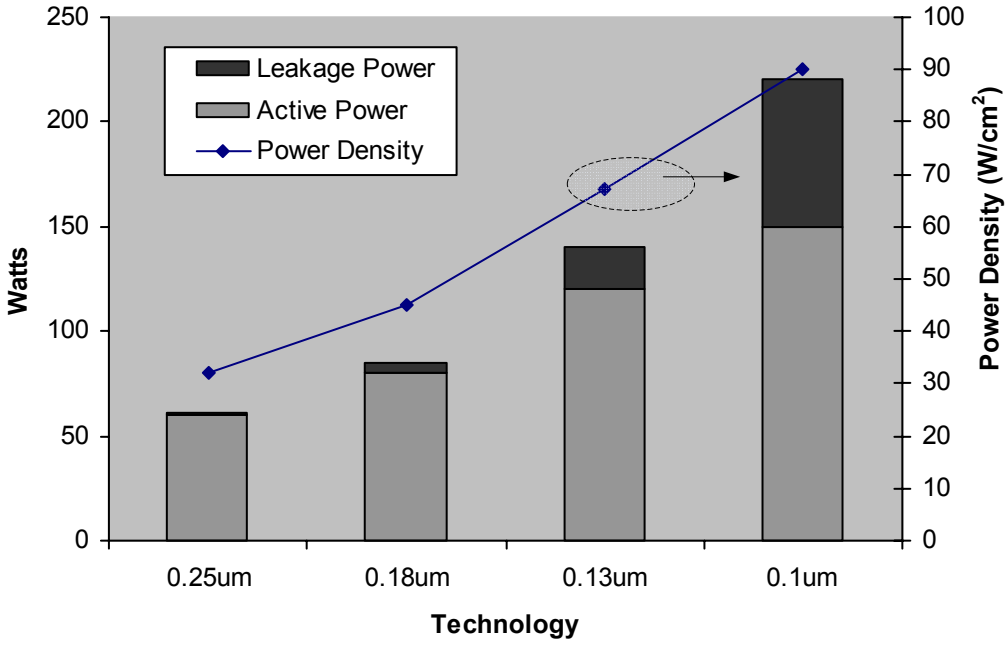

Figure 3.2: Leakage power contribution to total power with technology scaling

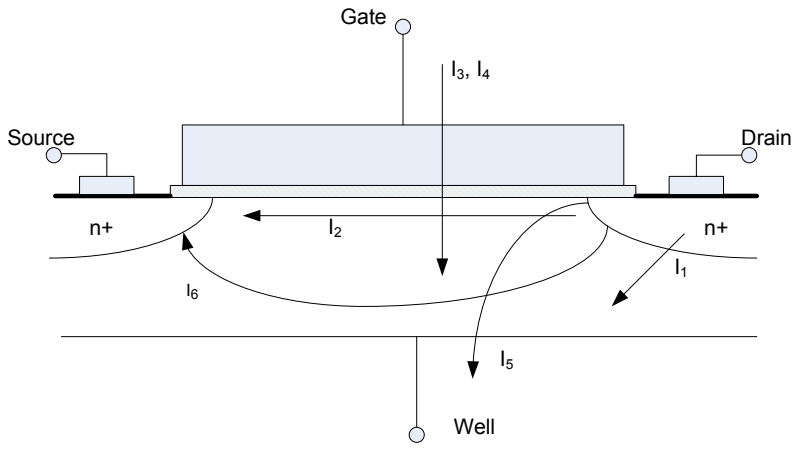

Figure 3.3: Leakage currents in a short channel transistor

whereas for the 0.1μm technology, it is almost 25% of the total power. It is projected that for 65nm technology, the leakage power would be as high as 50% of the total power. Therefore leakage power management is very important in scaled technologies.

The shrinking geometries has led to other sources of leakage current. Fig. 3.3 shows the leakage currents through a short channel transistor [13].

I_1 is the reverse-bias pn junction leakage current. This current is caused because of the minority carrier diffusion/drift near the edge of the depletion region, and due to electron-hole pair generation in the depletion region of the reverse biased junction [13]. In case the n and p regions are heavily doped then band-to-band tunneling (BTBT) starts to dominate the leakage current in the pn junction. The BTBT leakage current flows under high electric field conditions, when the electrons from the valence band of p tunnel into the conduction band of n.

I_2 is the subthreshold leakage current. This current occurs between the source and drain of the transistor due to weak inversion in the subthreshold region, when the gate voltage is below the threshold voltage V_{th} [13].

I_3 is the gate tunneling current through the gate oxide of the transistor. With the reduction of gate oxide thickness the electric field across the gate oxide increases. These lead to tunneling of electrons from the substrate to gate and from gate to substrate resulting in the gate oxide tunneling current [13].

I_4 is the current due to injection of hot carriers from substrate to gate oxide. The short-channel transistors have high electric field near the silicon and gate oxide interface. This results in holes and electrons gaining sufficient energy to cross interface barrier to enter the oxide layer, resulting in hot-carrier injection [13].

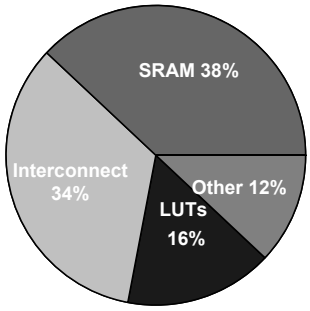

Figure 3.4: Leakage breakdown among different FPGA elements [4]

I_5 is the gate induced drain leakage current (GIDL). This phenomenon occurs because of high electric field effect in the drain junction of the transistor [13].

I_6 is the punch-through current. This occurs in short channel devices because the source-substrate and drain-substrate depletion regions tend to come closer. When these depletion regions merge, punch-through occurs [13].

3.2.2 Leakage Power in FPGAs

The work in [4] analyzed leakage power in a state of the art 90nm FPGA using SPICE simulations with BSIM4 models. The leakage power was reported to be 4.2μW per CLB at $25\,^{\circ}\mathrm{C}$. With a GSM cell phone's standby current budget of 300μA, the upper bound on leakage power would imply only 86 CLBs [4], which is too small for any significant processing. Hence leakage power is a very big obstacle for FPGAs to enter into mobile applications domain. The leakage power breakdown among different elements of the FPGA reported in [4] is shown in Fig. 3.4. It shows that leakage power from the configuration SRAM cells and the routing interconnects form a major part of the total leakage. Further, it shows that leakage from the unused parts of the FPGA can be as high as 56% for a small design using 50% of the available CLBs, whereas for a design which uses all the CLBs, the unused leakage is 35%, still a significant portion of total leakage. Therefore reducing leakage power in the unused parts of FPGAs is also very important.

3.2.3 Estimating Power Savings

The total power consumption can be divided into 2 parts, active power and standby power. The total average power can be written as

$$P_{avg} = \frac{t_{act} \times P_{act} + t_{off} \times P_{off}}{t_{act} + t_{off}} \tag{3.4}$$

$$T = t_{act} + t_{off} \tag{3.5}$$

where P_{avg}, P_{act}, and P_{off} are the average, active and standby (off) power. For personal wireless communication systems, typically the standby time or off time (t_{off}) is 90% of the total time (T) and active time (t_{act})is 10% of the total time. During the active time the components of power dissipation can be written as

$$P_{act} = [P_{dyn} + P_{sckt} + P_{actleak}]_{used} + [P_{actleak}]_{unused} \tag{3.6}$$

where P_{dyn},P_{sckt}, and $P_{actleak}$ are the dynamic, short circuit and active leakage power consumptions respectively. P_{off} is the standby leakage power consumption of the FPGA, because during the standby mode, only leakage power is dissipated. Hence, reducing leakage power during the standby mode for mobile applications would increase the battery life significantly.

3.3 Leakage Power Modeling for FPGAs

Analytical equations for leakage computation have been studied and developed in detail, which can model the complex behavior of various components of leakage current in a MOS transistor. These models are based on physical and empirical parameters [25]. Typically, the leakage power consumption of any circuit is not only dependent on the physical parameters of the circuit, but is also heavily dependent on the inputs to the circuit. The work in [5] showed that the leakage current can vary by an order of magnitude depending on the input to the circuit and demonstrated that certain input vectors are the dominant leakage states for a logic gate.

There have been very few works targeted at modeling leakage power for the FPGAs. The work in [11] modeled the dynamic and the leakage power in FPGAs. The power model was integrated into the VPR framework. The power model framework is shown in Fig. 3.5. It developed an activity estimation tool, using a transition density model to estimate the activities of the internal nodes of the FPGA for dynamic power computation.

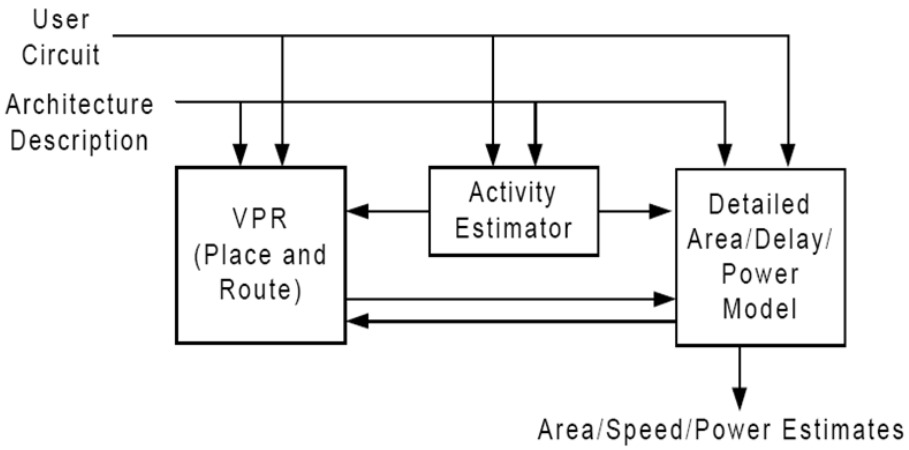

Figure 3.5: Power model framework developed in [11]

It used the concept of boolean difference to propagate the signal probabilities which is given by the following equation for a given boolean function $f(x)$.

$$\frac{df(x)}{dx_i} = f(x_i) \oplus f(\overline{x_i}) \tag{3.7}$$

The probability of boolean difference, $P(\frac{df(x)}{dx_i})$ represents the static probability that a change in x_i would produce a change at the output. With an input transition density (number of transitions per second) of $D(x_i)$, the total transition density at the output is given by [11]:

$$D(y) = \sum_i \frac{df(x)}{dx_i}.D(x_i) \tag{3.8}$$

The primary inputs were considered to be uncorrelated having a static probability of 0.5 and transition density of 0.5. The dynamic power is then given by

$$P_{dyn} = \sum_{all\ nodes} \frac{1}{2}.{V_{dd}}^2.C_y.D(y).f_{clk} \tag{3.9}$$

where C_y, and f_{clk} are the capacitance of the node y, and clock frequency, respectively. The short circuit power was assumed to be 10% of the dynamic power. This work modeled only subthreshold leakage. The subthreshold leakage was modeled using the following equation for a transistor,

$$I_{sub} = I_{on}.exp\left[\frac{(V_{gs} - V_{on}).q}{n.k.T}\right] \tag{3.10}$$

where $V_{on} = V_{th} + n.K.T/q$, n is the subthreshold swing coefficient, k is the Boltzman's constant, q is the electron charge and V_{gs} is the gate to source voltage of the transistor, I_{on} is the drain current when $V_{gs} = V_{on}$. For the inactive transistors, V_{gs} was considered as half of the threshold voltage.

However, this work has some major drawbacks in leakage power modeling. This work considered only subthreshold leakage and did not consider the dependency of subthreshold leakage on the state of the circuit, rather it calculated an average leakage considering that all the transistors were leaking and the V_{gs} was considered as half of V_{th} for leakage computation, which is not accurate. The short channel effects have not been taken into account in this work. These produce inaccurate estimation of leakage current. Further, for scaled technologies it is important to model the gate leakage.

The work in [8] and [4] calculated total power using look-up table based approach based on SPICE simulations to characterize the power of the FPGA circuit elements. The look-up table stores the leakage power of the circuit elements for different inputs or an average leakage for each circuit element. The total leakage is computed by adding the leakage of all the circuit elements. However, this methodology is not accurate as the leakage power is strongly dependent on the state of the inputs and considering an average leakage for the circuit elements of the FPGA leads to inaccurate leakage estimation.

Motivated by the above mentioned limitations of the previous works, this work develops an analytical model for leakage power calculation for FPGAs, that takes into account the dependency of the leakage power on the state of the circuit. The contribution of this work on leakage power modeling for FPGAs can be summarized as:

1. Developing analytical models and methodology to compute subthreshold and gate leakage power for FPGAs, independent of the technology node.
2. Computation of state dependent subthreshold and gate leakage.
3. Analysis of sources of leakage in FPGAs.

3.4 Leakage Power Reduction in FPGAs

In this work reduction of subthreshold leakage reduction is targeted, because the gate leakage is still orders of magnitude smaller than subthreshold leakage for current generation technologies. The subthreshold leakage current through a MOSFET can be modeled as shown in equation 4.1. Since the subthreshold leakage current is exponentially dependent on the threshold voltage, increasing the threshold voltage would decrease the

leakage current substantially. However, the high threshold voltage devices have larger switching delays.

The various leakage current mechanisms and some leakage reduction techniques for CMOS circuits were discussed in [13][2]. The techniques for reducing leakage power involve static and dynamic approaches. The dynamic approach involves run time decision making for leakage reduction. One such popular technique is the use of sleep transistors in MTCMOS circuits for controlling the leakage power [16]. Several optimizations for MTCMOS circuits involving the use of sleep transistors have been proposed, such as in [14][15]. However, this technique can reduce only standby leakage power and leads to performance degradation. The static technique does not involve run time decision making for leakage control. The dual threshold voltage design technique, which is a static approach, has been widely used in the custom VLSI designs for reducing leakage power. The dual-Vt implementation reduces both, the active leakage and the standby leakage. Further, there is no performance degradation in a dual-Vt implementation for custom VLSI designs.

The dual threshold voltage design technique uses two kinds of transistors in the same circuit. Some transistors have a high threshold voltage, while other transistors have a low threshold voltage. The high threshold voltage transistors have less subthreshold leakage power dissipation but also have a larger delay as compared to the low threshold voltage transistors. Fig. 3.6 shows the concept of dual threshold voltage implementation in custom VLSI designs. Here, gates on non-critical paths are assigned high-Vt and the gates on the critical path are assigned low-Vt. The objective is to maximize the number of transistors having high threshold voltage without sacrificing the performance of the circuit. Several prior works have proposed algorithms which assign high-Vt and low-Vt to the logic gates of the given circuit [1][5][17][18]. However the dual threshold voltage design technique proposed in the literature for custom VLSI designs cannot be used for FPGAs. This is because the FPGAs are programmable and the circuit that would be eventually implemented on it is unknown and hence the delays through various paths of the circuit are not known. In this work a dual-Vt FPGA CAD flow for designing and evaluating different dual-Vt FPGA architectures is proposed and developed. No published work has proposed such a CAD flow.

There have been very few works targeted at reducing the leakage power in FPGAs. The leakage reduction techniques can be broadly divided into techniques that target reduction of leakage in logic, routing or both.

The work in [3] used a technique based on the property that the leakage power consumed by a CMOS circuit is dependent on the state of its inputs and used the signal statistics to alter the state of the inputs in order to reduce the leakage power in such a way that the functionality of the circuit does not change. It explains that FPGA circuit

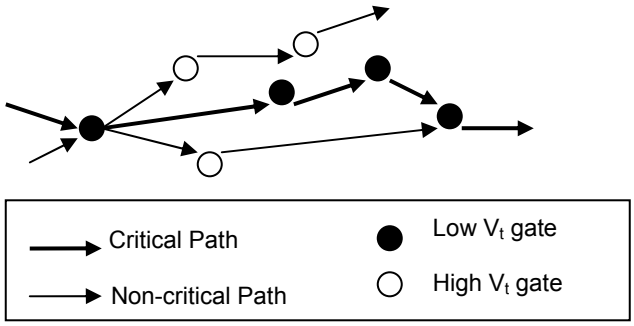

Figure 3.6: Dual-Vt design implementation

elements such as multiplexers and pass transistors should have their inputs and outputs at logic level 1 to reduce active leakage. For example, the gate leakage of a turned on pass transistor is more when a logic 0 is being driven as compared to the case when logic 1 is passed through it. The methodology tries to maximize the time that a signal spends in logic level 1. It uses the concept of static probability of signals to alter the state of the signal. If the static probability of the signal is less than 0.5, then it is a candidate for signal inversion. The signal is inverted if it is possible to do so without changing the functionality of the circuit. It reports an average active leakage savings of 25%. However this work addressed only active leakage. Standby leakage reduction is very important for mobile application because mobile devices typically spend almost 90% of the time in standby mode. Further, this work addressed leakage reduction only in the used parts of the FPGA, and it was shown in [4] that leakage from the unused parts of FPGA can be as high as 56% of the total leakage.

The work in [6] approached the problem of reducing leakage power by dividing the FPGA fabric into small regions with each of the regions being controlled by a sleep transistor which would be turned on/off depending upon whether that region is being used or not, thus reducing the leakage power. It uses a region constrained placement technique to maximize the sleep time. It reports leakage savings of around 20%-30%, with performance loss of 8%. However, this technique requires extra sleep transistors and control circuitry for configuration of the control bits, leading to area overhead. Further, it uses dynamic reconfiguration of SRAM cells, i.e. during the actual run time of the application, which leads to problems associated with the wake up time for the system, and an extra overhead. With the extra overhead the leakage energy savings is reported as 19%. Finally, this technique leads to reduction of only standby leakage power.

The work in [7] explored the dual-Vt, body biasing and gate biasing of nMOS pass transistors for reducing leakage power. The dual-Vt technique used in [7] was based on

varying the percentage of high threshold voltage elements in the routing resources of the FPGA and studying its effect for a number of benchmarks. It did not specify the leakage savings obtained, and no detailed evaluation of several possible routing architectures was presented. Body-biasing technique requires control circuitry and generation of voltages for body biasing. Further, this technique leads to reduction in leakage savings with technology scaling [19]. The negative gate biasing of nMOS pass transistors has implementation issues and also leads to additional leakage current component, called gate induced drain leakage.

The work in [22] used a dual-Vdd/Vt architecture to reduce leakage power and dynamic power. Only the configuration SRAM cells were assigned high-Vt to reduce leakage power. Two types of logic blocks are used, logic blocks with high-Vdd and logic blocks with low-Vdd. The logic blocks have a fixed pattern on the FPGA. An algorithm is used to assign high-Vdd/low-Vdd to the logic blocks based on slack available and power sensitivity. It uses a level converter to transfer the logic between high-Vdd and low-Vdd regions. An overall power savings of around 14% is reported for the dual-Vdd architectures with non-negligible delay penalties. However, the dual-Vdd approach requires design of power supply network with two voltage levels and level converters which lead to area overhead and increased complexity. Further it is difficult to increase the granularity of the approach because that would imply increasing the complexity of the power supply network.

The work in [44] uses logic blocks with dual-Vdd supplies which can be configured so that the logic blocks can be assigned high-Vdd/low-Vdd during the configuration phase for power and performance tradeoffs and leakage power is reduced only in the logic blocks. This technique reduces both the dynamic and leakage power. It uses three kinds of logic blocks, H-block having a fixed high-Vdd, L-block having a low-Vdd and P-block having configurable V_{dd}. The P-logic blocks are allowed to have a 5% performance degradation and the power switch leads to area overhead of 24% for these blocks. An architecture, arch-DV, is used with H/L/P blocks having the ratio 1/1/3 with an overall area overhead of 14%. An overall power savings of 9.04% is reported for the arch-DV. For the architecture having all P-blocks with an area overhead of 24%, an overall power savings of 14.3% is reported. This work has problems with area overhead management, and design of supply network with dual-Vdd. No analysis of delay penalties was presented, rather the comparison with the baseline architecture was done for a number of fixed clock frequencies. A similar work was done in [39] using configurable dual-Vdd blocks with a different algorithm to assign high-Vdd/low-Vdd to logic blocks. An average overall power savings of 61% was reported with leakage savings of 71% with delay penalty of around 20%. This work has similar design issues as above.

The work in [45] proposes low power routing switch design for reducing the leakage

and switching power. The low power switch is designed by having sleep transistors for the buffer at the output of the multiplexer switch. Two sleep transistors are used in parallel, NMOS sleep transistor called MNX, and PMOS sleep transistor called MPX, providing virtual Vdd to the output buffer. In the high speed mode MPX is turned on and MNX is turned off. In the low power mode MNX is turned on and MPX is turned off. In the low power mode the virtual Vdd is equal to $Vdd - V_{th}$, leading to lower output swing and reduced subthreshold leakage. In an alternate design the body of the PMOS switches in the buffer is tied to virtual Vdd, leading to reduced threshold voltage and consequently increased drive strength. This leads to increased leakage, but the buffer can be smaller in size leading to area reduction. The switches were designed such that the performance loss is within 5%. It also took advantage of the fact that for most of the routing switches in FPGAs, significant slacks are available, so that a significant fraction of the routing switches can be of this type. A leakage power savings of 36% was reported for the switch in the low power mode versus the high speed mode. For the alternate switch design, a leakage power savings of 28% was reported. The switching energy was reduced by 29%. In the sleep mode, a leakage savings of 61% was observed. The proposed switch is 1.3 times larger in area as compared to the traditional switch, whereas the alternate switch is 1.2 times larger in area as compared to the traditional switch. The area overhead for the complete FPGA is estimated to be around 20%. The main drawback of this work is that it leads to non-negligible area overhead.

Motivated by the limitations of the leakage power reduction techniques for FPGAs, a dual-Vt technique and CAD flow that has the following advantages, is proposed in this work:

1. Reduces both active leakage power and standby leakage power.
2. Provides a CAD framework for developing and evaluating a dual-Vt FPGA implementation.
3. The inherent area penalty in using sleep transistors is not present in this design technique.
4. The dual-Vt architecture does not require any modification in the existing placement and routing tools from the perspective of the users.

3.5 Summary

This chapter presented work that has been done on leakage power modeling and reduction for FPGAs. The methodologies were described and the results were discussed. The

drawbacks of these techniques were discussed, which served as the motivation for this work.

In the next chapter, the leakage power model for FPGAs developed in this work is explained and results obtained from the leakage power model are discussed. Chapter 5 discusses the dual-Vt FPGA architecture and CAD flow for leakage power reduction.

Chapter 4

Analytical State Dependent Leakage Power Model for FPGAs

4.1 Introduction

In this chapter the leakage power model for FPGAs is explained and discussed. Analytical models are used for computing leakage current through each of the FPGA circuit elements. A library of functions is used for this purpose. This library of funtions takes the input as the probability of state of the FPGA circuit element and the technology parameters and computes the leakage for the FPGA circuit element. This is repeated for all the elements in the FPGA and the total leakage current is computed as the sum of the individual leakage currents.

Analytical models for leakage currents in a transistor are explained in the next section and leakage models to account for short-channel effects are developed. The leakage currents in various FPGA circuit elements and their state dependency are discussed in section 4.3. Section 4.4 discusses the overall framework for computing the leakage power for FPGAs. Finally, in section 4.5 results obtained from the leakage power model are discussed.

4.2 Analytical Models for Leakage Currents

The leakage power model considers the subthreshold and the gate leakage. The following are the subthreshold and gate leakage equations used in the power model [25].

$$I_{sub} = I_0\left[1 - exp\big[\frac{-V_{ds}}{V_T}\big]\right].exp\left[\frac{V_{gs} - V_{th} - V_{off}}{nV_T}\right] \tag{4.1}$$

$$I_0 = \mu\frac{W}{L}\sqrt{\frac{q\epsilon_{si}NDEP}{2\phi_s}V_T^2} \tag{4.2}$$

$$I_{gc0} = \frac{W.L.A.V_{gs}.V_{aux}}{T_{ox}^2}.exp\Big[- B.T_{ox}.$$
$$\big(AIGC - BIGC.V_{oxdepinv}\big).\big(1 + CIGC.V_{oxdepinv}\big)\Big] \tag{4.3}$$

$$V_{aux} = NIGC.V_T.log\big[1 + exp\big(\frac{Vgs - V_{th0}}{NIGC.V_T}\big)\big] \tag{4.4}$$

$$V_{oxdepinv} = K1.\sqrt{\phi_s} + V_{gs} - V_{th} \tag{4.5}$$

$$I_{gcs} = \frac{PIGCD.V_{ds} + exp(-PIGCD.V_{ds})}{PIGCD^2.V_{ds}^2 + 2e - 4} - \frac{1 + 1e - 4}{PIGCD^2.V_{ds}^2 + 2e - 4} \tag{4.6}$$

$$I_{gcd} = \frac{1 - (PIGCD.V_{ds} + 1).exp(-PIGCD.V_{ds})}{PIGCD^2.V_{ds}^2 + 2e - 4} + \frac{1e - 4}{PIGCD^2.V_{ds}^2 + 2e - 4} \tag{4.7}$$

The subthreshold leakage (I_{sub}) equations [25] are given by equations (4.1) and (4.2), where V_T is the thermal voltage, V_{off} is the offset voltage which determines the channel current at $V_{gs} = 0$, n is the subthreshold swing coefficient, $W, L, \mu, q, \phi_s, \epsilon_{si}$, are the width, length, mobility of charge carriers, electron charge, surface potential and permittivity of silicon, respectively, for the transistor. Since only the gate to channel current (I_{gc0})is the dominant gate leakage current, and the gate current for the PMOS is significantly smaller than the gate current for the NMOS, only gate to channel current for the NMOS is modeled [26]. However, the proposed model can be easily extended to incorporate other gate leakage components and the gate leakage for the PMOS. The gate leakage equations are given by (4.3)- (4.7), where A, B are physical constants, T_{ox} is the gate oxide thickness, $AIGC$, $BIGC$, $CIGC$, and $NIGC$ are the empirical parameters, $K1$ is the first order body bias coefficient. Equation (4.3) is used for computing I_{gc0} and

equations (4.6), and (4.7) are used for partitioning I_{gc0} into the source current I_{gcs} and drain current I_{gcd}, where $PIGCD$ is a parameter for the partitioning.

In this work, industrial CMOS 130nm and CMOS 90nm processes were used for the leakage analysis of the FPGA using the leakage power model. The deep-submicron MOSFETs have various short channel effects (SCE) which were not present in long channel devices. For the CMOS 130nm process that was used, it was observed that the threshold voltage (V_{th}) of the NMOS was affected by the reverse narrow width effect (RNWE) [13], i.e., the threshold voltage of the transistor increased as the width of the transistor increased from the minimum width, which consequently reduced the leakage of the transistor. Further, the threshold voltage of the transistors are also affected by the drain to source voltage (V_{ds}). The threshold voltage of the transistor decreases when the drain to source voltage is increased. To incorporate these effects into the models, the experimental data from the SPICE simulation was curve fitted to empirical equations as follows:

$$V_{th}|_{(V_{ds}=0)} = V_0\big(1 - a.exp(-b_1.W - b_2.W^2)\big) \tag{4.8}$$

$$V_{th} = V_{th}|_{(V_{ds}=0)} - m.V_{ds} \tag{4.9}$$

where W is the width of the transistor, equation 4.8 models the RNWE, and equation 4.9 models the impact of V_{ds} on V_{th}. For CMOS 130nm NMOS, $V_0 = 0.412V$, $a = 0.345003$, $b_1 = 1.01194$, $b_2 = -0.0568004$, and $m = 0.02125$. For CMOS 130nm PMOS, the RNWE was not too significant, so only the dependence of V_{th} on V_{ds} was modeled, with $m = 0.02388$. These values were determined from curve fitting of the simulation data. For the CMOS 90nm process a similar RNWE for the NMOS was observed, and the dependence of V_{th} on V_{ds} was observed for both NMOS and PMOS. However, for the CMOS 90nm PMOS a narrow width effect (NWE)[13] was observed, which results in increasing V_{th} as the width of the transistor is decreased. The RNWE and V_{ds} dependence for CMOS 90nm NMOS were modeled using equations (4.8) and (4.9) with the constants as $V_0 = 0.320812$, $a = 0.437178$, $b_1 = 1.2$, $b_2 = -0.068$, and $m = 0.0668$. For the PMOS, a model using curve fitting to account for the NWE was developed, as follows:

$$V_{th} = \frac{f_1 + f_2.W + f_3.W^2}{g_1 + g_2.W + g_3.W^2} \tag{4.10}$$

where $f_1 = 0.49$, $f_2 = 1.16679$, $f_3 = -1.51$, $g_1 = 0.318$, $g_2 = 4.3$ and $g_1 = -0.533$. The impact V_{ds} was modeled using equation (4.9), with $m = -0.0468$. These equations have been used in our leakage power model to model the reverse narrow width effect and the dependence of V_{th} on V_{ds}. Although the constants used in these equations make it technology dependent, the data can be easily extracted by simulating only one device under few different widths and drain to source voltages.

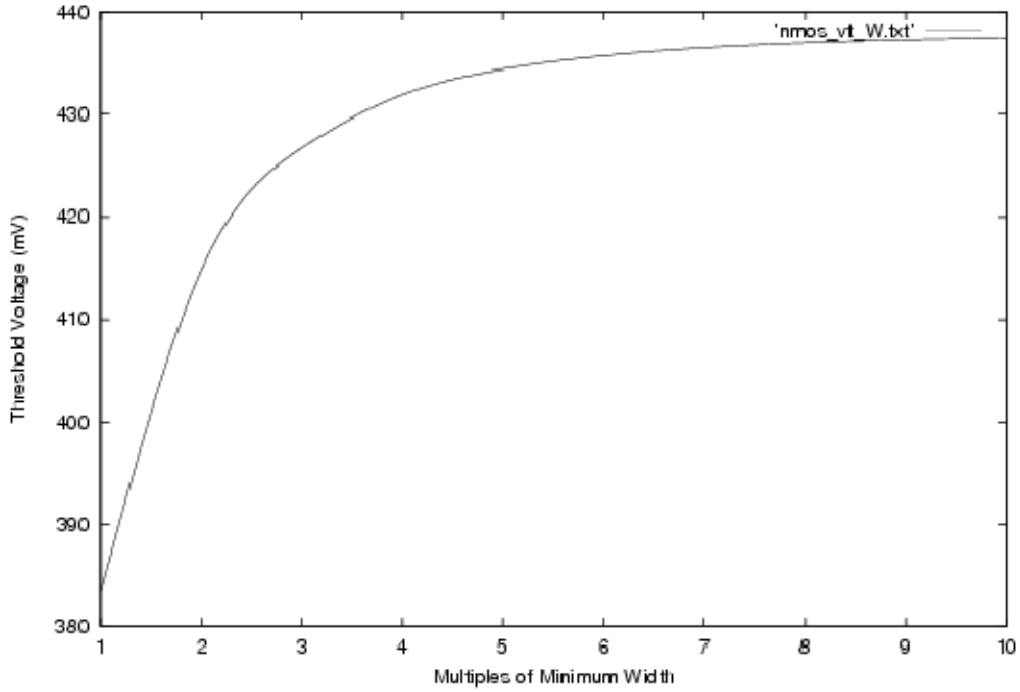

Figure 4.1: Dependence of V_{th} on the width of NMOS for CMOS 130nm

For illustrative purposes the dependence of V_{th} on the width of the transistor and the drain to source voltage are shown in Fig. 4.1 and Fig. 4.2, respectively for NMOS in CMOS 130nm. Fig. 4.1 shows the dependence of V_{th} on the width of NMOS. It shows that as the width is increased the threshold voltage increases rapidly, and flattens out after the width of the transistor is 7 times the minimum width. Fig. 4.2 shows the dependence of V_{th} on the drain to source voltage. It can be seen that the threshold voltage of the NMOS decreases linearly with the drain to source voltage.

Table 4.1 shows that the inclusion of the RNWE in the power model greatly improves the overall accuracy of the power model. The base threshold voltages of the devices were determined from the SPICE simulation so that various effects can be accounted for in the model.

4.3 Leakage in FPGA Circuit Elements

This section describes the various leakage current components that have been modeled in different circuit elements. The inverters were sized for equal rise and fall times, and

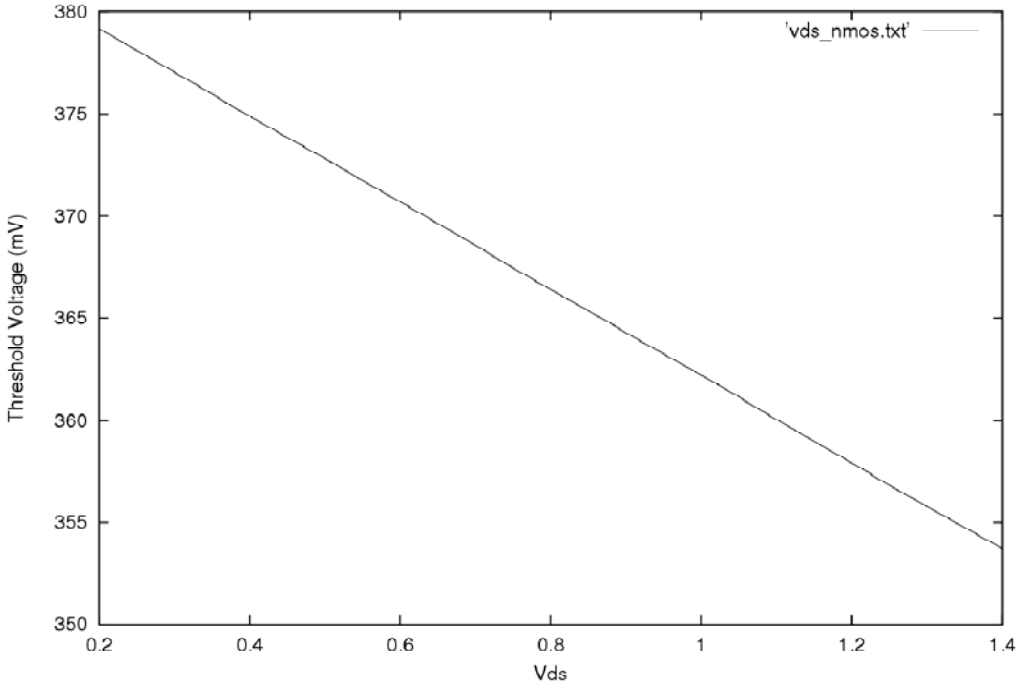

Figure 4.2: Dependence of V_{th} on drain to source voltage for NMOS in CMOS 130nm

Table 4.1: Comparison of Power Model with the SPICE simulations for CMOS 130nm

Circuit Element	SPICE (pW)	Power Model (without SCE) (pW)	Power Model (with SCE) (pW)	Error (without SCE)	Error (with SCE)
Inverter (2x)	372.7	901.2	411.8	141%	10.5%
4-Binary Tree	1156	1352	1212	16.9%	4.8%
Buffered Switch	883.2	1403	873.4	58.8%	1.1%

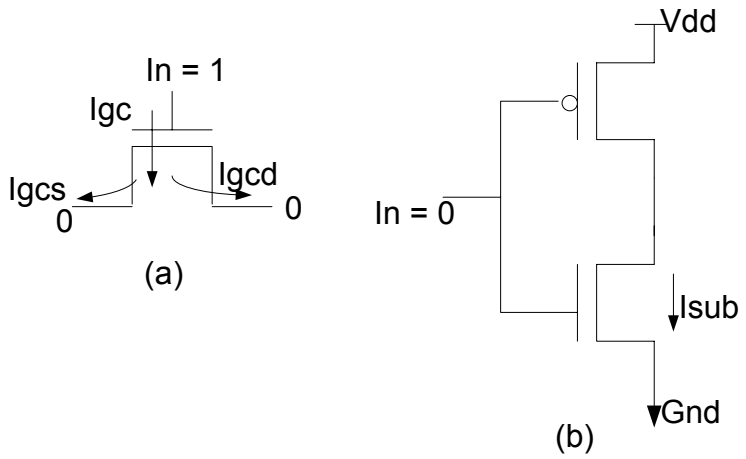

Figure 4.3: (a) Gate leakage in NMOS (b) Subthreshold leakage in Inverter

for minimizing the delay and area product [9]. All the multiplexers were implemented with minimum sized transistors, the SRAM cells are considered to have minimum sized transistors with high-V_{th} to mitigate subthreshold leakage, and the routing switches were optimized for area and delay product. Both the PMOS and NMOS in various circuit elements are considered as the candidates for subthreshold leakage, but only the NMOS transistors are considered as candidates for gate leakage because the gate leakage in PMOS is considerably smaller than NMOS [26]. Furthermore, the back gate leakage of the NMOS transistors is ignored and only the gate current from the gate to channel is considered, which then gets partitioned, and flows into the source and the drain of the transistor as shown in Fig. 4.3(a). The methodology that was adopted for computing the leakage power for each of the circuit elements in the FPGA is described below.

Inverter: The subthreshold leakage of the inverters is modeled in both the states, i.e, when the input is 0 and when the input is 1 and the gate leakage of the inverter when the input is 1. With the input at 0, only subthreshold leakage flows through the NMOS of the inverter and the gate leakage through PMOS is ignored as shown in Fig. 4.3(b). When the input to the inverter is 1, there is subthreshold leakage through the PMOS and gate leakage through the NMOS.

Multiplexer: In FPGAs, the multiplexers are implemented with NMOS pass transistor structures. The multiplexer is binary tree implemented using pass transistors. The leakage currents in the multiplexer is again strongly dependent on the state of its inputs. The multiplexer leakage under two cases are as follows.

Case1: Fig. 4.4 shows the structure of the multiplexer and the subthreshold and gate leakages for the select signal (0,0) and the input vector (0010). Only one transis-

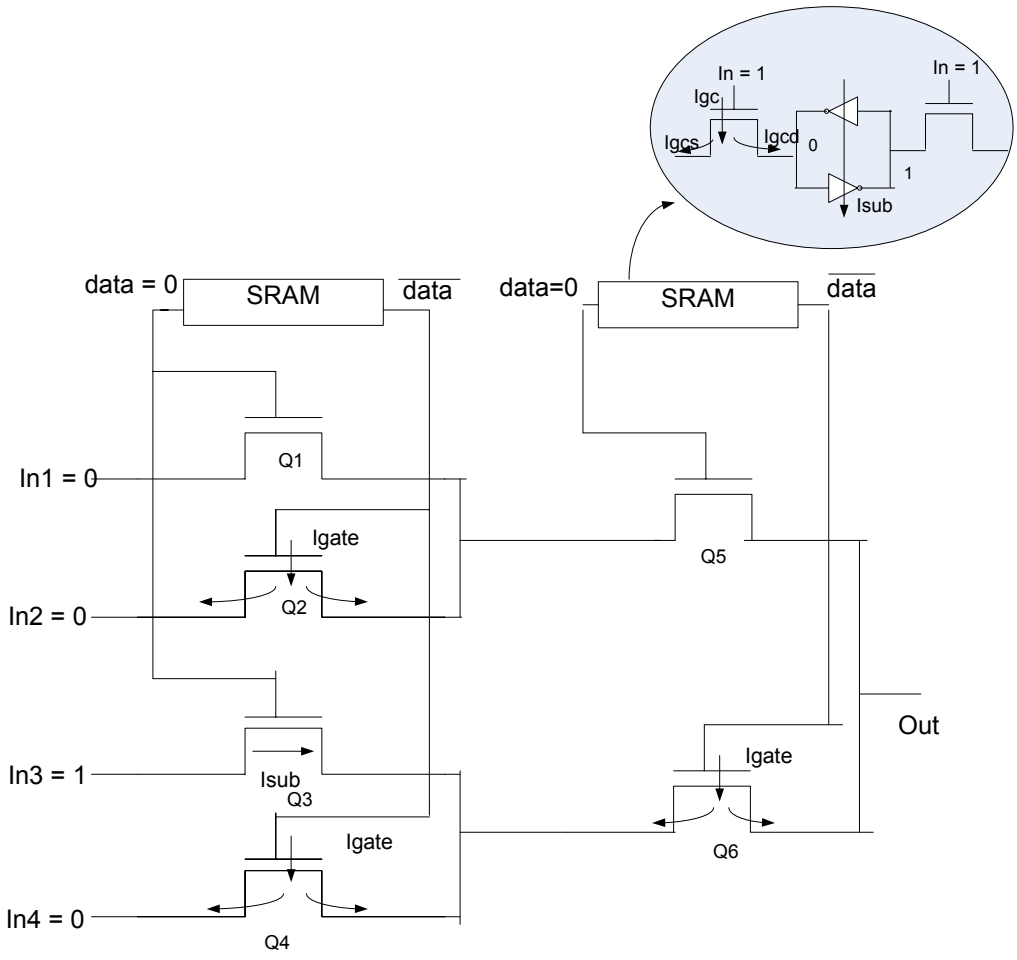

Figure 4.4: Multiplexer structure and the corresponding state dependent leakage for a particular select signal and input vector

tor (Q3) has subthreshold leakage, whereas three transistors have gate leakage currents (Q2,Q4,Q6). However, when the input vector changes to (0110), keeping the select signal same, there are three transistors which have subthreshold leakage (Q1,Q3,Q5) and two transistors have gate leakage (Q1,Q6). Therefore it is quite important to consider the state dependency of leakage currents in any circuit.

Case2: Another phenomenon that needs to be accounted for in the pass transistor structures is that of the impact of V_{ds} on the threshold voltage of the transistor. Consider the case of two cascaded pass transistors as shown in Fig. 4.5. Here, transistor Q2 has subthreshold leakage. However, the drain terminal of Q2 is not at V_{dd}, but at a smaller

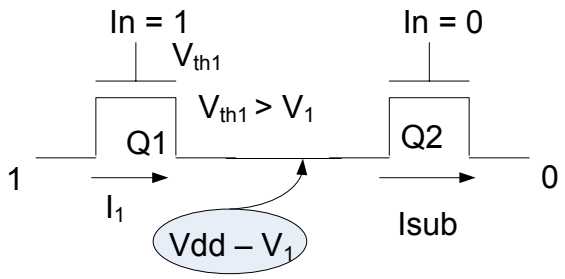

Figure 4.5: Leakage in multiplexers is affected by the voltage drop during signal propagation

value, which is $V_{dd} - V_1$, where V_1 is voltage which is smaller than the threshold voltage of Q1, (V_{th1}). This reduced drain voltage increases the threshold voltage of transistor Q2, which reduces the subthreshold leakage through Q2. It is interesting to note that $V_1 < V_{th1}$. This can be explained as follows. When Q1 tries to charge the drain node of Q2, Q1 has to be turned on, which implies that initially $V_1 > V_{th1}$ and Q1 is on and charges the node till $V_1 = V_{th1}$. After this, Q1 is turned off and subthreshold leakage current through it charges the drain node of Q2. At steady state Q1 needs to supply only the subthreshold leakage current which is flowing through Q2. Under this condition, Q1 need not be turned on fully, i.e., it can operate in the subthreshold region and still provide sufficient current for the leakage current through Q2. Hence a steady state is reached when the voltage drop across Q1 is adequate to provide the necessary current. V_1 was assumed a constant value of $0.2V$, and $0.1V$ for CMOS 130nm and CMOS 90nm respectively. These values have been arrived at, using SPICE simulations and provide sufficiently accurate results. The leakage value reported in Table 4.1 for the 4 input binary tree takes this value of V_1.

SRAM Cells: The FPGA contains many SRAM cells which are used for configuring the FPGA. These SRAM cells are configured only once and it remains constant throughout the run time of the FPGA. The standard six transistor SRAM cell is considered in this work. The SRAM cells are implemented with high-V_{th} transistors, because the SRAM cells are used only in the read mode, and is configured only once, and hence does not result in any performance penalty. This reduces the subthreshold leakage significantly and many commercial FPGAs have high-V_{th} SRAM cells. The leakage through two inverters connected back to back and gate leakage through one of the access pass transistors have been modeled.

LUTs: The look-up tables (LUTs) consist of an array of SRAM cells and a multiplexer. The array of SRAM cells implement the truth table and the multiplexer selects the SRAM cell based on the input to the LUT. The leakage for the LUTs would again be

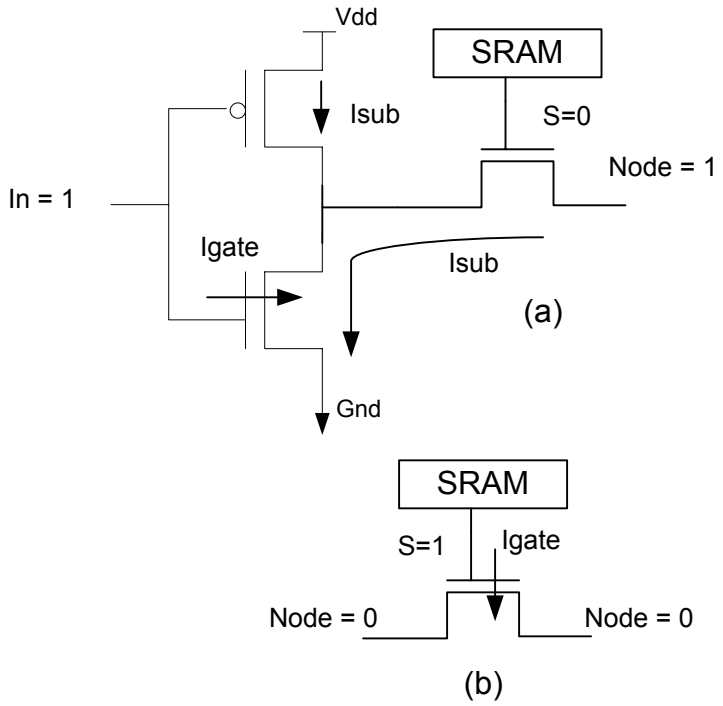

Figure 4.6: (a) Buffered routing switch. Subthreshold and gate leakage currents under certain input conditions. (b) Pass transistor routing switch. Only gate leakage is present when the switch is turned on.

state dependent as explained above, for the multiplexers and the inverters.

D Flip-flop: The D flip-flops are again made of latches and pass transistors so the leakage current for the flip-flops can also be modeled in terms of the basic inverter and pass transistors with the appropriate sizes of the transistors.

Routing Switches: There are two kinds of routing switches that are present in this FPGA architecture, namely, buffered routing switches and pass transistor based routing switches. Both switches have NMOS pass transistors. Fig. 4.6(a) shows the leakage currents through this switch when it is turned off with the input node at logic 1 and output node also at logic 1. In this case there is subthreshold leakage through the PMOS of the inverter and through the pass transistor of the switch. Fig. 4.6(b) shows the gate leakage current that flows through the pass transistor switch when the switch is turned on, and logic 0 is being passed through the switch.

The pass transistors in the routing switches have to drive buffers at the end of routing segments. When logic 1 is being driven through a NMOS pass transistor, it leads to a V_{th} drop in the voltage level of the signal. This leads to both the PMOS and NMOS

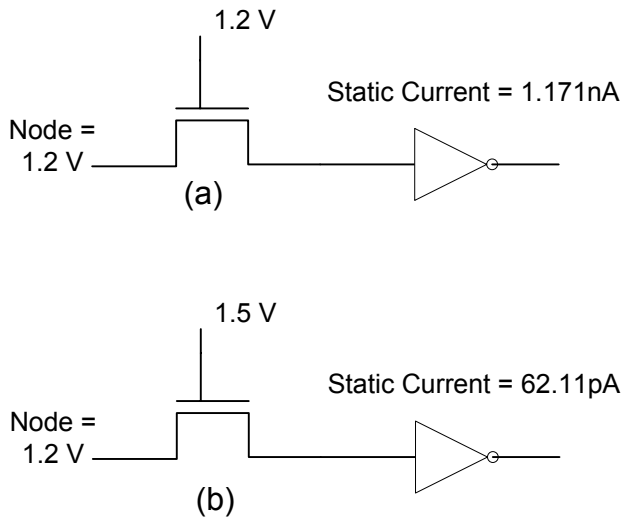

Figure 4.7: (a) Static current without gate boosting. (b) Reduced static current with gate boosting.

of the driven buffer to get partially turned on leading to large static current. To address this problem commercial FPGAs employ gate boosting of the NMOS pass transistors to decrease the static current dissipated in the buffer driven by the NMOS pass transistor as depicted in Fig. 4.7. In this case the gates of the NMOS pass transistors are driven by a higher input voltage. Fig. 4.7 shows that the static current gets reduced considerably when gate boosting is employed.

4.4 Leakage Power Model

In this section, the leakage power model framework is described. The overall architecture of the leakage power model is shown in Fig. 4.8. The widely used academic and research tool, VPR [9], has been used for placement and routing of the benchmark circuits. After the placement and routing of the given circuit, the power model computes the probability of states for each node of the circuit. For computing the probability of the nodes of the circuit, the work done in [11] has been used. The probability for each of the nodes is computed by propagating the static probability of the signals at the input, which are considered to be independent. The probability of any signal for a boolean function can be computed using the binary decision diagrams (BDD) [35]. Binary decision diagrams represent a logic function graphically. A function $f(x_1, ..., x_n)$ can be written as

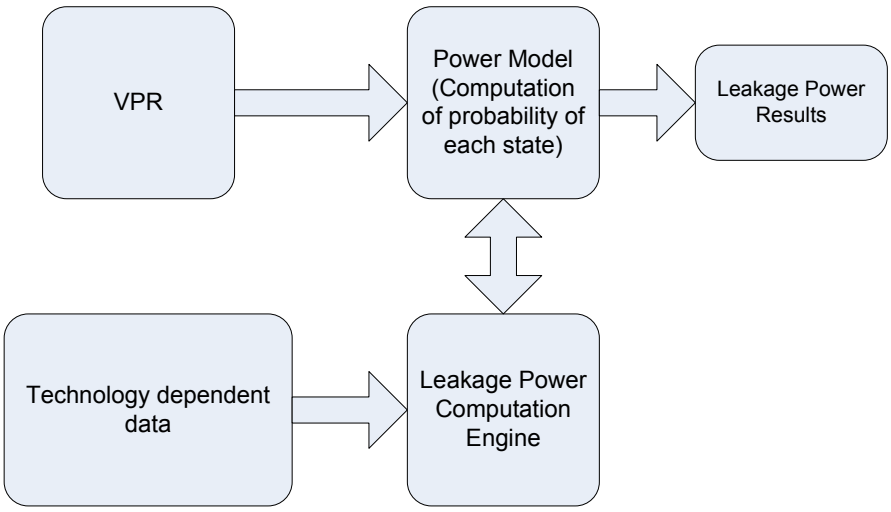

Figure 4.8: Overall architecture of the leakage power model

$$f = x_i.f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n) + \overline{x_i}.f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \tag{4.11}$$

using *Shannon expansion*, where

$$f_{xi} = f(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n) \tag{4.12}$$
$$f_{\overline{xi}} = f(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \tag{4.13}$$

are the $cofactors$ of f and are obtained by replacing x_i with logic 1 and logic 0, respectively, in f. An input x_i is represented by a node in the BDD, and the edge coming out of the node represent the value of the input x_i. The value of the function can be determined by simply traversing the BDD from its root. The calculation of probability then becomes [35]

$$P(f) = P(x_i).P(f_{xi}) + P(\overline{x_i}).P(f_{\overline{xi}}) \tag{4.14}$$

Starting with $i = 1$, a depth first traversal of BDD would yield the the probability $P(f)$. For any boolean logic function $f(x)$, the boolean difference is calculated as follows:

$$\frac{df(x)}{dx_i} = f(x_i) \oplus f(\overline{x_i}) \tag{4.15}$$

The probability of the boolean function $P\left(\frac{df(x)}{dx_i}\right)$, then gives the probability that a change in x_i would produce a change at the output. After the probability of states for each of the node is computed, the power model looks at each of the circuit elements of the FPGA and computes the leakage for each of the states of that element using the Leakage Computation Engine (LCE). The LCE has been implement to compute the leakage for each of the circuit elements of the FPGA based on given input vector (and the state of the SRAM cells, if they are present in the given circuit element). The LCE is basically a library of state dependent leakage calculation functions. This library has the basic leakage equations for subthreshold leakage and gate leakage and the computation of the associated parameters, based on equations (4.1)-(4.7). It also has the models for computing the leakage for each of the FPGA circuit elements for a given input vector. For the used part of the logic block, the actual probability of states is considered depending on the input probability. For the unused part of the logic block, it is assumed that all the SRAM cell configuration bits are programmed to zero and the probability of states is computed accordingly. In case of used pass transistor switches, they consume only gate leakage. The used buffered switches have subthreshold leakage power in the buffers and gate leakage in the pass transistor. For unused switches we consider that all the switches have different logic level at their two nodes. In this case, the buffers have both the gate and subthreshold leakage, whereas the pass transistors have only the gate leakage.

The leakage power model takes into account the state dependency of leakage power by considering the probability of states for each of the circuit element. Consider a circuit element which has n states and the probability of the states are $Prob_1$,$Prob_2$, ..., $Prob_n$, such that $\sum_{i=1}^{n} Prob_i = 1$. The leakage power for different states are given as $Pleak_1$, $Pleak_2$, ..., $Pleak_n$. The average leakage power can then be written as:

$$P_{avgleak} = \sum_{i=1}^{n} Prob_i.Pleak_i \tag{4.16}$$

4.5 Results and Discussion

For evaluating the leakage power consumption of different benchmarks, a fixed FPGA architecture for the benchmarks was taken. Smaller benchmarks had 20x20 logic blocks and a routing channel width of 100. Each logic block is made up of a cluster of 12 sub-blocks. For the larger benchmarks (bigkey, des, dsip), a square array of 35x35 logic blocks and a routing channel width of 100 was assumed.

Table 4.2 shows the leakage power consumption for different benchmarks. It can be seen that the dominant leakage for both the technologies is the subthreshold leak-

Table 4.2: Subthreshold and gate leakage for different benchmarks

Benchmark	Subthreshold Leakage		Gate Leakage		Total Leakage	
	130nm (μW)	90nm (μW)	130nm (pW)	90nm (μW)	130nm (μW)	90nm μW)
alu4	138.34	596	498	21.7	138.34	617
apex2	152.37	667	502	21.6	152.37	689.46
apex4	131.6	567	509.6	22.1	131.6	589
bigkey	342.5	1455	1445	64.1	342.5	1519
des	347.7	1482	1447	64	347.7	1546
diffeq	140.5	603	486	21.1	140.5	624
dsip	332.3	1403	14351	63.7	332.3	1467
elliptic	187.2	844	517	21.8	187.2	866
ex1010	195.8	912	584	24.2	195.8	936
ex5p	126.6	541	505	22	126.6	563
frisc	185.3	838	525	22.1	185.3	860
misex3	136.5	588	498	21.6	136.5	609
s298	156.7	679	479	20.8	156.7	700
spla	177.5	659	549	21.5	177.5	681
tseng	128.8	547	490	21.4	128.8	569

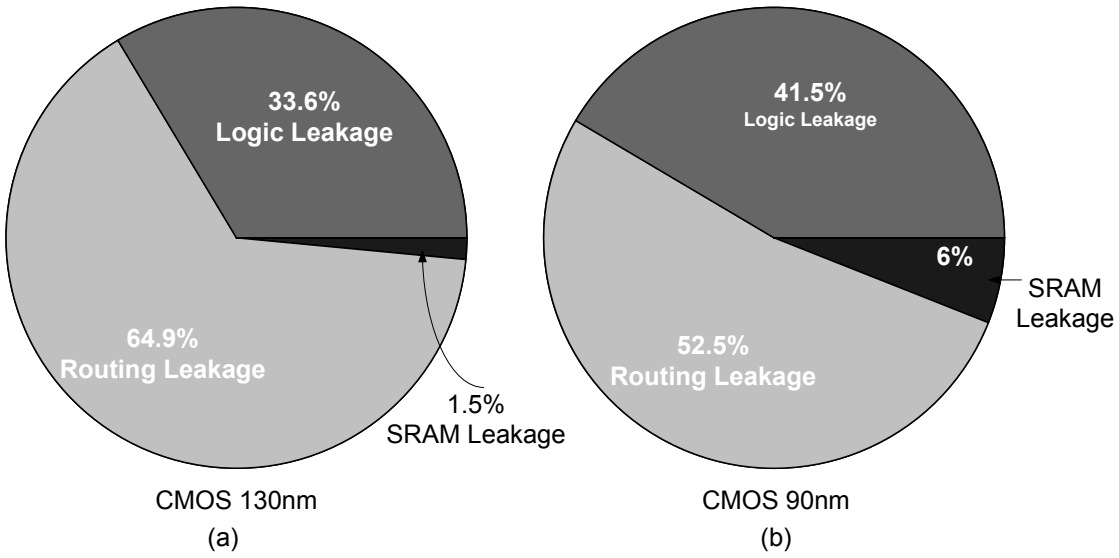

Figure 4.9: Average Leakage distribution for different parts the FPGA for CMOS 130nm and 90nm

age, the gate leakage being orders of magnitude smaller. The subthreshold leakage for the CMOS 90nm FPGA is almost 4 times greater than the subthreshold leakage for the CMOS 130nm FPGA. Further, it can be seen that the gate leakage increases exponentially with technology scaling. For the CMOS 90nm FPGA, the gate leakage is orders of magnitude greater than the gate leakage for CMOS 130nm FPGA. This result is consistent with the fact that the contribution of the gate leakage to total leakage increases with technology scaling. The state dependency of the leakage is evident from the fact that the leakage power for different benchmarks are different, even though the same size of FPGA is considered for benchmarks. It should be noted that the total SRAM leakage remains constant for all the benchmarks as it is dependent only on the total number of SRAM cells and since we used a fixed size FPGA for all the benchmarks, the SRAM leakage remains constant for all the implementations. However, the leakage, especially in the logic part is strongly state dependent. For example, the logic leakage in case of the benchmark spla is 72μW, whereas in case of ex5p is 35.72μW, almost half of the logic leakage of spla (CMOS 130nm). For the CMOS 90nm the logic leakage for spla and ex5p are 368μW, and 208μW, respectively, again showing a lot of dependency on state.

Fig. 4.9 shows the distribution of average leakage power in different parts of the FPGA for the CMOS 130nm and CMOS 90nm. The SRAM leakage is very small as compared to the logic and the routing leakage because the SRAM cells are implemented with high-V_{th} transistors. It can be seen that the dominant leakage is the routing leakage

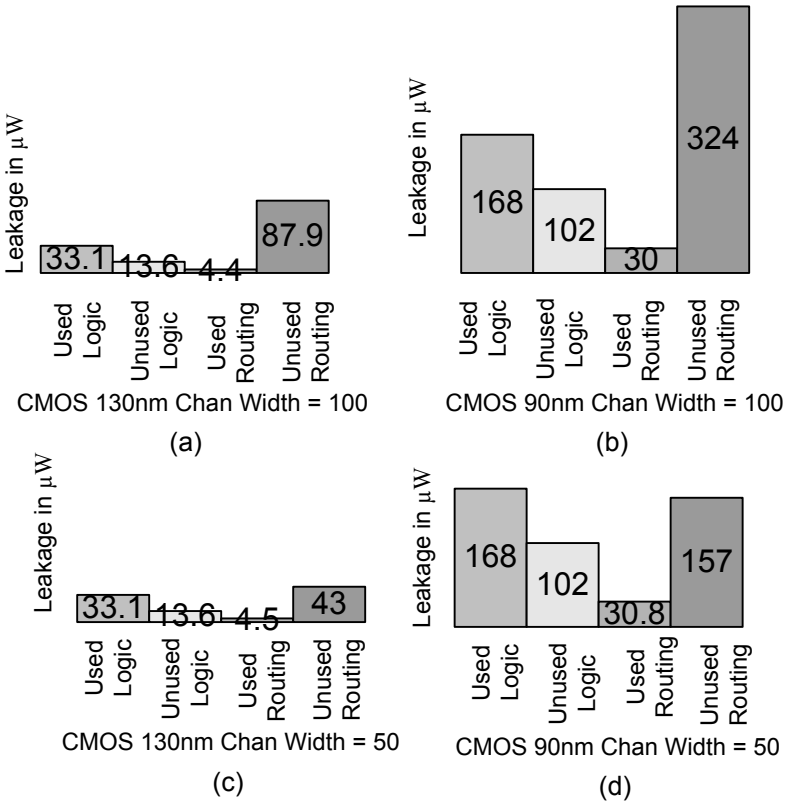

Figure 4.10: (a),(b)Used and unused leakage for different components of FPGA for the benchmark alu4 for the FPGA architecture with routing channel width of 100 (c),(d) With routing channel width of 50

for both CMOS 130nm and CMOS 90nm. However, the contribution of logic leakage to the total leakage increases from 33.6% to 41.5% when the technology is scaled from CMOS 130nm to CMOS 90nm. This is because the V_{th} of the PMOS is CMOS 90nm suffered from narrow width effect, which consequently reduced it, leading to increased contribution of leakage from the PMOS. Since most of the PMOS transistors are present in the logic part, the contribution of the logic leakage to the total leakage increased. It is evident from Fig. 4.10 that routing leakage is the dominant leakage power for the given FPGA architecture. Majority of the routing leakage power comes from the unused part of the routing resources. Further, for most FPGA designs the logic utilization is quite high, whereas the utilization of routing resources is quite low. In the initial case with the routing channel width of 100, the routing leakage was considerably larger than the logic leakage. Almost all the routing leakage comes from the unused routing resources, whereas the major part of the logic leakage is from the used logic part. However, when

the routing channel width is reduced to 50 (alu4 can be placed and routed with a channel width of 50), the leakage from the routing resources reduces to almost half, as expected. This results in significant reduction of total leakage. The contribution of the logic leakage to total leakage becomes slightly more than the routing leakage for the CMOS 130nm. For CMOS 90nm the logic leakage clearly starts dominating the routing leakage, once the routing resources is reduced to half. However, the total logic leakage remains constant, as expected. This clearly shows that FPGA CAD tools should try to increase the utilization of the routing resources, so that FPGAs can be implemented with lesser routing resources to reduce the leakage.

4.6 Summary

This chapter discussed the leakage power model developed in this work. Analytical models based on BSIM4 for computing leakage current were presented and the methodology for computing leakage through a FPGA circuit element was discusse. It was shown that the leakage current in a circuit is heavily dependent on the inputs to the circuit element. This was accounted for in the leakage power model. Finally, some of the dominant short channel effects were also accounted for, which greatly improves the accuracy of leakage computation. Results were presented for CMOS 90nm and CMOS 130nm, which indicate that the leakage through a CMOS 90nm FPGA is 4 times more than the leakage through a CMOS 130nm FPGA.

In the next chapter, dual-Vt FPGA architectures are explored, and leakage savings results are obtained using the leakage power model developed in this chapter.

Chapter 5

Dual-Vt FPGA Design for Leakage Reduction

5.1 Introduction

In this chapter, dual-Vt FPGAs are explored. Dual-Vt FPGA architectures are proposed and a dual-Vt FPGA CAD framework is proposed for designing the dual-Vt FPGA architectures. Using the dual-Vt FPGA CAD framework, various parameters for the dual-Vt FPGA architectures can be optimized and determined. The dual-Vt FPGA can then be fabricated with these parameters and user can then map the application on to the FPGA. The dual-Vt design technique has been widely studied for the ASICs and is a very popular methodology for designing low leakage VLSI circuits in the industry. However, dual-Vt FPGAs have not yet been investigated. This work proposes several dual-Vt FPGA architectures and a CAD framework for finding the architectural parameters of dual-Vt FPGAs.

5.2 Technology Used

An industrial CMOS 0.13μm technology node has been used for the technology dependent data for the FPGA architectures. The methodology proposed in [9] was used to extract the required technology dependent data. For extracting the delay data for the logic blocks, SPICE simulation was done. The delay through the subblock is the delay through the LUT input multiplexer, LUT, flip-flop, and multiplexer for selecting the output of the flip-flop or LUT. The sizing of the buffers was done for equal rise and fall

time as proposed in [9]. Metal 3 is used for the routing wires. The sizing of the routing switches was done for minimum area delay product as proposed in [9]. The regular MOS models have been used for low threshold voltage implementation. The Vdd for this technology is $1.2V$ and Vt is $0.2V$. The high-Vt transistors have threshold voltage 100 mV higher than the low-Vt transistors. SPICE simulation shows that on increasing the high-Vt value further, the delays of the subblocks increase without leading to any significant leakage savings. Since the SRAM cells do not contribute to any run time delay, it is assumed that they are implemented with high-Vt transistors to reduce the leakage power, and hence the results for leakage savings do not include any leakage savings from the SRAM cells in this work[4].

5.3 Proposed Dual Threshold FPGA Architecture

The very nature of programmability of FPGAs makes the dual-Vt design technique for FPGAs different from dual-Vt custom VLSI designs. This work proposes dual-Vt FPGA architectures, and then uses a CAD framework for determining the parameters and subsequently evaluating the proposed dual-Vt FPGA architectures. This section describes the proposed dual-Vt FPGA architectures and the dual-Vt FPGA CAD flow. This work targets the logic elements within the CLBs and the routing resources as candidates for dual-Vt assignment. The following architectures are evaluated:

1. Homogeneous Architecture with only subblocks within CLBs considered for dual-Vt assignment.($arch1$)
2. Homogeneous Architecture with subblocks and routing resources considered for dual-Vt assignment.($arch2$)
3. Heterogeneous Architecture with only subblocks considered for dual-Vt assignment. ($arch3$)
4. Heterogeneous Architecture with the subblocks and routing resources considered for dual-Vt assignment. ($arch4$)

The proposed homogeneous and the heterogeneous architectures are described below.

5.3.1 Homogeneous dual-Vt FPGA architecture

This proposed architecture has a certain number of high-Vt subblocks and low-Vt subblocks in each of the CLBs. For a cluster size of N, M ($M < N$) subblocks are assigned

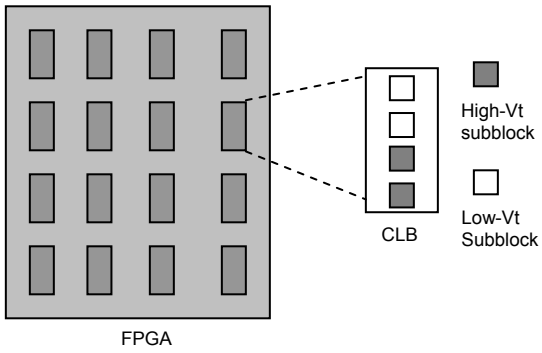

Figure 5.1: Proposed homogeneous FPGA architecture. Each CLB has a fixed ratio of high-Vt and low-Vt subblocks.

high-Vt, whereas $(N - M)$ subblocks are assigned low-Vt in each CLB. For example, Fig. 5.1 shows a homogeneous dual-Vt FPGA architecture in which each CLB has 50% high-Vt and 50% low-Vt subblocks for a cluster size of 4. In the case of $arch1$ only the subblocks are considered as candidates for dual-Vt assignment. This architecture is extended to include the routing resources as candidates for dual-Vt assignment, which leads to the architecture $arch2$. The switches in the connection blocks and the switch blocks, which drive the routing segments are considered for dual-Vt assignment. In the architecture $arch2$, certain fraction of these switches are assigned high-Vt. Fig. 5.2 shows the switch block and the associated routing switches. The connection block has similar switches for providing connections between CLBs and routing segments.

5.3.2 Heterogeneous dual-Vt FPGA architecture

This proposed architecture has two kinds of CLBs distributed uniformly throughout the array. The first type of CLBs has subblocks, all of which have high-Vt and the second type has a certain number of high-Vt subblocks. These two kinds of logic blocks are distributed uniformly throughout the array in such a way that the array is regular. If the logic block cluster size in N, then the proposed FPGA architecture would have two kinds of logic blocks distributed uniformly. One of those two kinds would have all N high-Vt subblocks ($type1$), whereas the second kind of logic blocks would have M high-Vt subblocks ($M < N$), and $N - M$ low-Vt subblocks.

Fig. 5.3 shows the distribution of the two kinds of the logic blocks in the FPGA, such that 50% of the logic blocks are of $type1$, whereas the other 50% of the logic blocks are of $type2$ for a cluster size of 4. Such an architecture was selected because the number

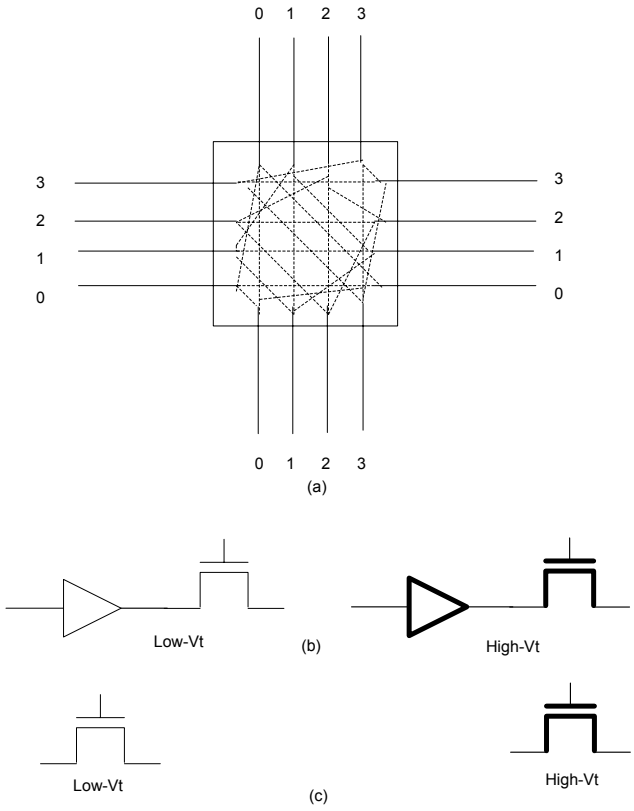

Figure 5.2: Switch block. (a) The overall architecture of a switch block (b) Buffered pass transistor switch (c) Pass transistor based switch

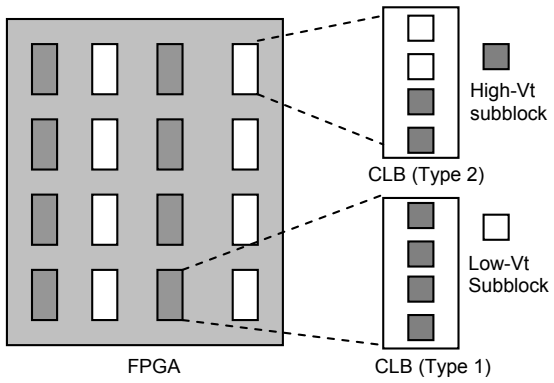

Figure 5.3: Proposed heterogeneous FPGA architecture. Two kinds of CLBs; one having all high-Vt subblocks, the other having a fixed ratio of high-Vt and low-Vt subblocks.

of subblocks assigned high-Vt was very high and so a large percentage of logic blocks can possibly have all high-Vt subblocks. This heterogeneous architecture is extended to include the routing resources, leading to the architecture $arch4$ in the same way as for $arch2$.

5.3.3 Proposed Dual-Vt FPGA CAD Framework

Developing and evaluating dual-Vt FPGA architectures as outlined above would require a CAD framework which can perform dual-Vt assignment to the subblocks and the routing resources and has a methodology for analyzing different dual-Vt FPGA architectures. The typical existing FPGA CAD flow within the framework of VPR, for placing and routing a given netlist, and the proposed generic dual-Vt FPGA CAD flow are shown in the Fig. 5.4. The proposed dual-Vt FPGA CAD flow has been developed using the widely used academic research tools VPR and T-Vpack [9], and the state dependent leakage power model for FPGAs that was developed in this work and discussed in the previous chapter [10]. The typical FPGA CAD flow involves circuit optimization using SIS [20] and technology mapping to LUTs using FlowMap [21]. T-Vpack is then used to do the packing and clustering, which generates the netlist of logic blocks. VPR is then used for placement and routing. The proposed dual-Vt FPGA CAD framework has 6 stages as shown. The next section describes various stages of the CAD flow.

5.4 CAD framework implementation

The VPR was modified to support the dual-Vt assignment to the subblocks and the T-Vpack was modified to support re-clustering. Each of the stages of the dual-Vt FPGA CAD flow (Fig. 5.4(b))is explained below.

5.4.1 Stage 1

In this stage the $.net$ file is generated, which is a netlist of logic blocks, along with the activity and function files required for the power estimation. The *.net* file is generated from the $.blif$ (Berkeley Logic Interchange Format) file based upon the specified FPGA architecture, such as cluster size, inputs per cluster, LUT size etc., by the T-Vpack. The activity file generation and function file generation is a part of the power model framework [10][11]. These files serve as inputs to the second stage.

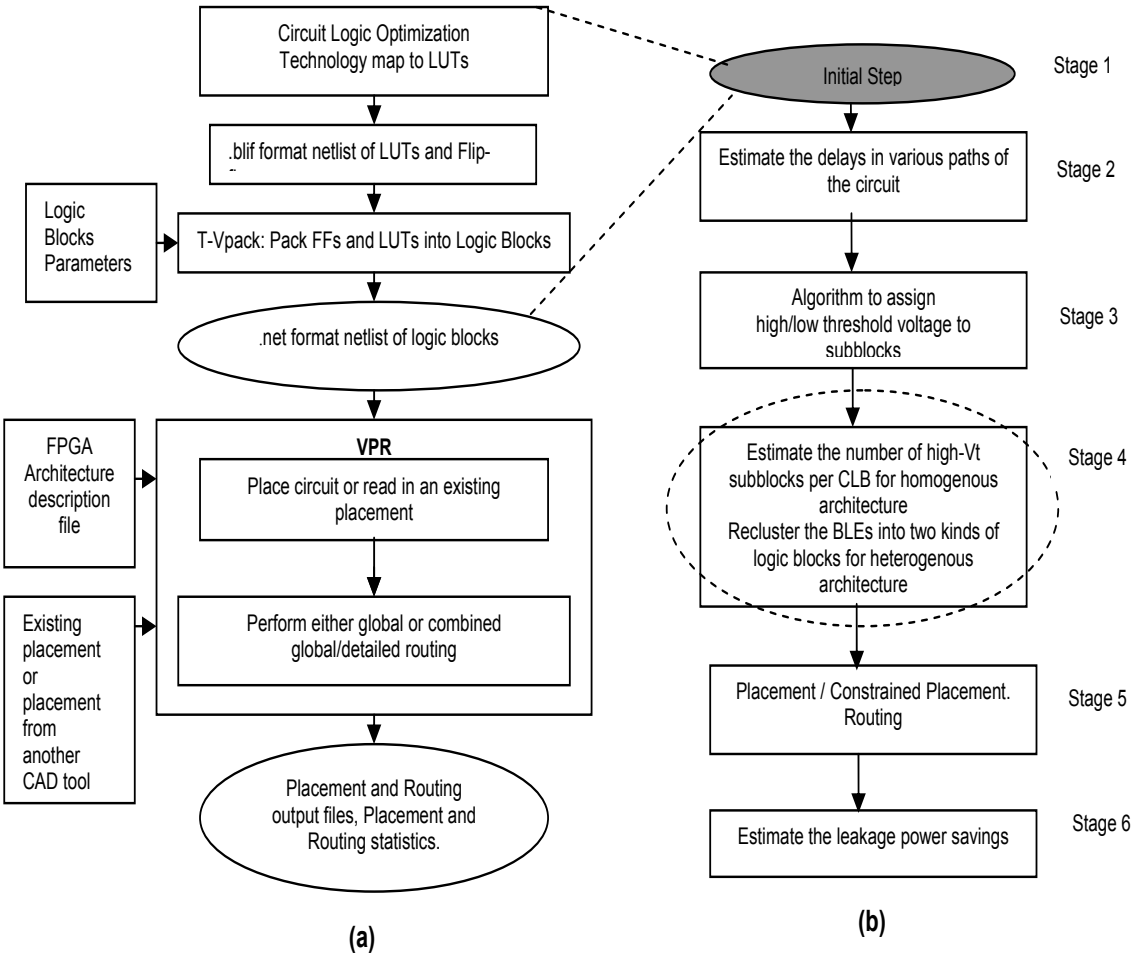

Figure 5.4: (a) Typical FPGA CAD flow within VPR and T-Vpack framework. (b) Proposed generic dual-Vt FPGA CAD flow

5.4.2 Stage 2

The second stage is a delay estimation stage, which is used to assign high-Vt and low-Vt to various subblocks based on the slacks available for each path. The delay calculation can either be accurate or can be based upon some estimation. In this work accurate delay estimation based on the actual placement and routing of the benchmark has been used. In this stage it is assumed that all the subblocks are low-Vt and the delay calculation is done based on the low-Vt delay values for the subblocks and routing resources. VPR does the placement and routing of the benchmark for delay calculation. Since the leakage power model is integrated into the VPR, the power for the single low threshold voltage implementation of the FPGA, for the benchmark, is calculated immediately after placement and routing. This leakage power consumption result is used in the final stage of the CAD flow for comparison with the leakage power consumed by the dual-Vt implementation and calculation of leakage savings.

5.4.3 Stage 3

Initially it is assumed that all the subblocks are low-Vt except the configuration SRAM cells which are all high-Vt for both, the single low-Vt implementation and dual-Vt implementation. Based on the delays of various paths of the circuit, computed in the previous stage, the dual-Vt assignment algorithm assigns high-Vt to various subblocks. The dual-Vt assignment algorithm is shown in Algorithm 1 [1]. The algorithm is a heuristic and

Algorithm 1 Dual-Vt assignment[1]

```
Using timinggraph in VPR (after Place and Route)
for each subblock do
    Compute the slack available
    if slack > 0 then
        Assign high-Vt to the subblock
        Recompute the slack
        if slack < 0 then
            Re-assign low-Vt to the subblock
        end if
    end if
end for
```

works on the timing graph created by the VPR for placement and routing. The nodes of the timing graph are levelized starting from the primary outputs and the graph is traversed level by level. This timing graph contains the information for the delays between various

pins of the circuit. For assigning high-Vt to a particular subblock the slack available at the output pin of the subblock is considered. The assignment of high-Vt would change the delays through the subblock. This delay value is updated in the timing graph and the next subblock is then considered for high-Vt assignment. It should be noted that the dual-Vt assignment is not constrained to keep the CLBs identical in terms of the number of high-Vt and low-Vt subblocks in each CLB. This kind of dual-Vt assignment, which is the ideal assignment, would result in a dual-Vt FPGA implementation which would not have any performance degradation. This dual-Vt assignment gives a theoretical upper limit for the number of subblocks high-Vt assigned. The dual-Vt assignment produces an output in which there are different numbers of high-Vt and low-Vt subblocks across different CLBs. This is, however, not a feasible solution. This information is needed by the next re-clustering stage to create a feasible netlist for the dual-Vt FPGA architectures.

5.4.4 Stage 4

This stage works within the framework of T-Vpack. For the homogeneous architectures ($arch1$ and $arch2$) the number of high-Vt subblocks per CLB is estimated based upon the results of the dual-Vt assignment to subblocks in the previous stage. Based on these results the evaluation of the homogeneous architecture is done with a fixed number of high-Vt subblocks per CLB.

The heterogeneous architectures ($arch3$ and $arch4$) are developed and evaluated in the following way based on Algorithm 2. The clustering algorithm proposed in [9] has been modified. The clustering algorithm in [9] picks up a seed BLE and then tries to build up the cluster by *attracting* the other BLEs based on an attraction function. The attraction function used in [9] to add a BLE B to a current cluster C is shown in equation (5.1)

$$Attraction = \alpha.Criticality(B) + (1 - \alpha)\left[\frac{Nets(B) \cap Nets(C)}{MaxNets}\right] \tag{5.1}$$

where the *criticality* represents that part of the attraction function which tries to reduce the delay of the circuit. For computing the delay of the various paths in the circuit during the packing stage the intra-cluster delay is assumed to be 0.1, logic delay is assumed to be 0.1 and inter-cluster delay is assumed to be 1 [9]. For the modified clustering algorithm, the delay of high-Vt BLEs is set to 0.2 to take into account for the increased delays of high-Vt BLEs. The exact values of these delays are not very important as long as they represent the correct trend [9]. The methodology for criticality computation is explained in [9], and its value lies between 0 and 1. The second part of the function tries to cluster the BLE which share maximum nets with the current cluster. To take into account the

high-Vt BLEs and increase the $type1$ (all high-Vt) clusters the modified cost function is shown in equation 5.2

$$Attraction = \alpha.Criticality(B) + (1 - \alpha - \beta)\left[\frac{Nets(B) \cap Nets(C)}{MaxNets}\right] + \beta.\left[\frac{Hvt.NumHvt + (1 - Hvt).NumLvt}{ClusterSize}\right] \quad (5.2)$$

where the third part of the equation denotes the attraction of a BLE to a cluster based on the number of high-Vt/low-Vt BLEs in the current cluster, to increase the $type1$ logic blocks. The parameter Hvt takes a value of 1 for a high-Vt BLE, and 0 for a low-Vt BLE. $NumHvt$ denotes the number of high-Vt BLEs in the current cluster, and $NumLvt$ denotes the number of low-Vt BLEs in the current cluster. The value of β depends upon the benchmark, but a value between 0.01 and 0.05 was found to give good results for all the benchmarks. The value of α was set to 0.8.

The number of low-Vt subblocks, $NumLvtBLEs$, in the $type2$ logic blocks is determined as follows

$$NumLvtBLEs = ClusterSize.\left(\frac{SumLvtCrit}{NumFracLB}\right) \quad (5.3)$$

where $SumLvtCrit$ denotes the sum of *criticality* of all low-Vt subblocks, $NumFracLB$ denotes the number of logic blocks which have at least one low-Vt subblock so that they can be categorized into $type2$ logic block. This equation has been used to determine the number of low-Vt subblocks in $type2$ CLBs because the criticality measure takes into account the importance of a subblock with regards to delay, which directly translates to the threshold voltage of the transistors of the subblock. The next step is to assign high-Vt/low-Vt to subblocks in $type2$ logic blocks so that each of the $type2$ logic blocks have the same number of low-Vt subblocks. All the subblocks in a logic block are arranged in the ascending order of *criticality*. If the logic block has more low-Vt subblocks than $NumLvtBLEs$, the BLEs are assigned high-Vt starting with the lowest critical low-Vt subblock, till the number of low-Vt subblocks is equal to $NumLvtBLEs$. If the logic block has low-Vt subblocks less than $NumLvtBLEs$, some of the subblocks are assigned low-Vt, starting with the highest critical high-Vt subblock, so that maximum gain in performance might be achieved. This method of assigning high-Vt/low-Vt to subblocks during the re-clustering stage is also used for the homogeneous architecture, such that the value $NumLvtBLEs$ is used for all the logic blocks.

This stage produces an output which gives an estimate of the number of $type1$ and $type2$ CLBs for the heterogeneous architectures ($arch3$ and $arch4$) and number of low-Vt subblocks in the $type2$ CLBs.

Algorithm 2 Recluster

Do clustering based on T-Vpack algorithm with modified attraction function
for each logic block **do**
  if $All - subblocks - high - Vt$ **then**
    Mark the logic block as $type1$
  else
    Mark the logic block as $type2$
  end if
end for
Determine the number of low-Vt subblocks in $type2$ CLB based on equation 5.3
for Each $type2$ logic block **do**
  Arrange the subblocks in ascending order of criticality
  if $No. - of - low - Vt - subblocks < NumLvtBLEs$ **then**
    while $No. - of - low - Vt - subblocks < NumLvtBLEs$ **do**
      Re-assign a high-Vt subblock to low-Vt based on criticality (Highest critical high-Vt subblock assigned low-Vt first)
    end while
  else
    while $No. - of - low - Vt - subblocks > NumLvtBLEs$ **do**
      Re-assign a low-Vt subblock to high-Vt based on criticality (Lowest critical low-Vt subblock assigned high-Vt first)
    end while
  end if
end for

5.4.5 Stage 5

Homogeneous Architectures: For the architecture $arch1$, placement and routing is done in this stage. For realizing the architecture $arch2$, a certain fraction of routing resources are assigned high-Vt and then placement and routing is done.

The high-Vt assignment to routing switches is done as follows. Consider a routing channel width of 100 with two kinds of segments, such that 50 segments are driven by pass transistor switches in the switch box and 50 segments are driven by buffered pass transistor switches as defined by the VPR architecture file. The output pins of the CLB drive all these segments with a buffered switch in the connection box. When high-Vt is assigned to, say 40% of routing segments driven by pass transistor switches, it means that 20 segments of those 50 segments will be driven by high-Vt pass transistor switches. Also, all the output pin switches of the CLBs which drive these 20 segments will be assigned high-Vt.

Gate boosting for these switches are done to minimize the voltage degradation when a logic 1 is propagated [9]. Other techniques, such as the use of level restorer, can be used to eliminate voltage degradation in the routing switches [23]. SPICE simulation shows that the logic level 1 passing through a high-Vt pass transistor with gate boosting to 1.5 V ($Vdd = 1.2V$) degrades to only 1.198 V from 1.2 V. The SPICE simulation results indicate that with gate boosting only 1.5% extra static leakage current flows in the driven buffer through a high-Vt pass transistor as compared to the case when there is no voltage degradation. Therefore the static power dissipation due to voltage degradation in high-Vt pass transistor switches is sufficiently small to be ignored.

Heterogeneous Architectures: Based on the re-clustering results, for a given FPGA array, the number of $type1$ and $type2$ CLBs are determined. These values are estimated based on the ratio of $type1$ and $type2$ logic blocks in the re-clustered netlist, after providing sufficient margin to account for the performance degradation because of physical restriction of FPGA regularity. These CLBs are assumed to be distributed evenly throughout the FPGA array. For the architecture $arch3$ *constrained placement* followed by routing is done. For $arch4$, a certain fraction x of the total routing switches are assigned high-Vt and then *constrained placement* followed by routing is done. The value of x is varied to determine the leakage savings and performance tradeoffs. The algorithm for constrained placement is shown in Algorithm 3. Since the VPR uses simulated annealing for timing-driven placement, a very effective placement technique, the basic placement algorithm has not been modified. However, the delays of the high-Vt and low-Vt parts of the CLBs and the routing resources are incorporated into the timing graph accordingly, so that the VPR can optimize the overall placement for performance.

Since the VPR router is timing driven, it optimizes the overall routing. The delays of

Algorithm 3 Constrained Placement

P-ratio = (Num $type1$ CLBs)/(Num $type2$ CLBs)
Determine physical locations for the two types of CLBs based on P-ratio
Assign high-Vt to $frac$ fraction of routing switches
Initial Random placement for all the CLBs
Start placement based on VPR placement algorithm
Allow the $type2$ CLBs to be placed on physical locations meant for $type1$ CLBs and vice versa
End Placement
for each $type2$ CLB **do**
  if $Logic - Block - not - on - same - type - CLB$ **then**
    Convert the logic block to $type1$ or $type2$ according to its placement
  end if
end for

the high-Vt switches would be more than the delays of the low-Vt switches which would force the router to use more of low-Vt switches. Therefore, the increased delays of the high-Vt routing switches are provided to the VPR, so that the router can optimize the overall routing.

5.4.6 Stage 6

This stage computes the results obtained from the previous stages of the CAD flow. After the computation of delays and power for the dual-Vt FPGA implementation is over, this stage computes the leakage power savings obtained and the delay penalty for the benchmark. The leakage power model developed in the previous chapter has been used for computing the leakage power before and after dual-Vt implementation.

5.5 Evaluation, Results and Discussions

This section describes the realization and evaluation of different FPGA architectures and their comparison. The method of developing and evaluating these architectures is by using a number of benchmarks on the proposed dual-Vt FPGA CAD flow to find out which subblocks can be high-Vt for maximizing leakage savings and reducing the delay penalties. It would be worthwhile to point out the difference between the physical design of the FPGA and the mapping of application on the FPGA. The physical resources of an FPGA are fixed and the applications are mapped later on. The dual-Vt FPGA CAD flow

that is outlined above is meant for the actual physical design of the FPGA and later on after the physical design of the FPGA is complete, for the design of the associated CAD tools. This dual-Vt FPGA CAD flow is to be used (along with other CAD tools) during the design of the FPGA, where number of benchmarks are evaluated before a design for an FPGA is finalized. This CAD flow is not meant for mapping of an application on to an FPGA chip, as all the logic blocks and routing resources would then be fixed, which would be a routine job using the standard FPGA CAD tools. Therefore, the dual-Vt FPGA CAD flow is meant for developing and evolving a dual-Vt FPGA, to determine its architectural parameters.

5.5.1 Evaluation Methodology

For obtaining the results from the benchmarks the LUT size of 4, and a cluster size of 12 were used. It was shown in [8] that a LUT size of 4 leads to minimization of total power. It was also shown in [8] that a cluster size of 8 or 12 leads to less power consumption than other cluster sizes. The inputs per cluster was chosen as $K/2(N+1)$, where N is the number of subblocks per cluster and K is the LUT size [24]. The default routing architecture present in the FPGA architecture file of VPR, having 50% routing segments with pass transistor switches and 50% routing segments with buffered pass transistor switches was used. For computing the leakage savings and performance tradeoffs, a fixed FPGA array of 20x20 for all the benchmarks was used, except for bigkey, clma, des, dsip and s38584.1, for which a 35x35 array was used. A routing channel width of 100 was assumed for the FPGA architecture. The fixed FPGA architecture represents a real world scenario.

Table 5.1 shows the results of the dual-Vt assignment. Based on these results, for the homogeneous architecture, architectures with 4, 6, and 8 high-Vt subblocks per CLB were considered for evaluation. The results for the number of $type1$ CLBs and low-Vt subblocks in $type2$ CLBs shown in Table 5.1 corresponds to a minimum sized FPGA and these values are ideal values which does not consider the regularity of structure of the FPGAs. Hence, to provide sufficient margin, to prevent severe degradation of final placement and routing imposed due to the limitation that the FPGA needs to be regular and uniform in structure, the values for $type1$ CLBs and number of low-Vt subblocks in $type2$ CLBs are set as 50% and 10, respectively.

5.5.2 Realizing and evaluating different FPGA architectures for leakage savings and design tradeoffs

Table 5.1 shows the results of *ideal* dual-Vt assignment. This indicates that a high percentage of subblocks can be assigned high-Vt. This is because most of the subblocks have a large slack available with them and a large part of the delay is contributed by the routing. We explain below the results for the various architectures.

$arch1$

This is a homogeneous architecture in which all the logic blocks are identical such that each logic block has a fixed ratio of high-Vt and low-Vt subblocks. A cluster size of 12 was used for these results. Based upon these results 3 different homogeneous architectures were evaluated with CLBs having 8, 6 and 4 high-Vt subblocks in each CLB. After assigning high-Vt to, say 8 subblocks in each CLB, different benchmarks were placed and routed to evaluate the leakage savings and performance tradeoffs. Table 5.1 shows the overall average leakage saving results for these three different cases. The mean performance penalties were 3.7%, 4.8% and 5.04% for the architectures with 4 , 6, and 8 high-Vt subblocks in each CLB. The delay penalties are close to each other because of two reasons, (1) there are sufficient number of low-Vt subblocks for the critical path(s) and, (2) the routing contributes to a larger portion of total delay, which means that even if there are few high-Vt subblocks on the critical path this would not cause a large delay penalty.

$arch2$

This is the homogeneous architecture with routing resources considered for dual-Vt assignment. The evaluation methodology is same as that of the $arch1$ case, except that we vary the fraction of the two kinds of routing resources assigned high-Vt and estimate the leakage savings and the delay tradeoffs. The evaluation was done for the three cases, viz. FPGA architecture with 4, 6 and 8 high-Vt subblocks per CLB for a cluster size of 12.

Fig. 5.5, shows the leakage savings and the performance tradeoffs when the fraction the high-Vt switches are varied over the base architecture having 6 high-Vt subblocks per CLB. Fig. 5.5(a) shows the case when only the buffered switches are considered for high-Vt assignment along with the CLB output pin switches which can drive these segments (i.e., those segments which can be driven by these high-Vt buffered switches), all the pass transistor switches remaining low-Vt. Fig. 5.5(b) shows the case when pass transistor switches are considered for high-Vt assignment along with the CLB output pin

Table 5.1: Leakage savings with logic blocks assigned dual-Vt for a cluster size of 12 for homogeneous and heterogeneous architectures

Bench-mark	No. of Sub-blocks / BLE	% of high-Vt subblocks for ideal dual-Vt Assign-ment	homogeneous Arch.($arch1$)			heterogeneous Arch.($arch3$)		
			4 high-Vt sub-blocks per CLB	6 high-Vt sub-blocks per CLB	8 high-Vt sub-blocks per CLB	% of $type1$ CLBs	No. of low-Vt sub-blocks in $type2$ CLB	Leakage Sav-ings
alu4	1522	89.6%	10.79%	16.1%	21.5%	46%	9	18.7%
apex2	1878	97%	11.7%	17.5%	23.4%	82.3%	9	20.75%
bigkey	1707	98%	8.1%	12.2%	16.4%	91.6%	11	13.9%
clma	8383	98%	13.7	20.7%	27.6%	87.7%	9	28.8%
des	1591	99%	8.4%	12.8%	17%	90%	8	14.8%
diffeq	1497	91%	10.5%	15.9%	21.2%	66.4%	8	18.4%
dsip	1370	97.2%	7.8%	11.6%	15.3%	94%	8	13%
elliptic	3604	99%	13.9%	21%	28%	94%	10	23.8%
ex5p	1064	97%	8.4%	12.8%	17.2%	79%	11	14.1%
frisc	3556	97.6%	13.3%	19.9%	26.6%	87.5%	9	23.2%
misex3	1397	97%	9.7%	14.7%	19.6%	83.7%	9	16.7%
pdc	4575	99.5%	12%	17.9%	23.9%	97%	10	20.5%
s298	1931	92.7%	12.3%	18.6%	24.8%	64.5%	12	20.9%
s38584.1	6447	96%	12.8%	19.1%	25.3%	78.3%	10	22%
seq	1750	96%	12.7%	18.3%	23.9%	71.9%	10	20.4%
tseng	1047	93.2%	9.6%	14.3%	19.1%	78%	10	17.2%
Mean	-	**93.3%**	**11%**	**16.5%**	**22%**	**93.3%**	**10**	**19.3%**

switches which can drive these segments (i.e., those segments which can be driven by these high-Vt pass transistor switches), all the buffered switches remaining low-Vt. The leakage savings increases considerably when the routing resources are assigned high-Vt with some impact on the performance. This is because a major part of leakage comes from the routing resources and is discussed further in section 5.5.4. Leakage savings for both the cases, when the pass transistor switches are assigned high-Vt and buffered switches are assigned high-Vt, are almost same. Although the buffered switches have more devices, but the pass transistor switch is 8 times the minimum size, whereas the pass transistor in the buffered switch is 2 times the minimum size and the buffer is 2 times the minimum sized buffer, which leads to almost similar leakage savings. The delay penalties for these architectures vary between 10.7% and 3.32%, which shows that it does not vary much with high-Vt assignment to the routing switches. This can be attributed to the fact that a large percentage of routing switches are unutilized [4], and therefore they do not contribute to delay penalties even if they are assigned high-Vt. Further, the routing optimization effort of the VPR would also increase, tending to use more of low-Vt switches, which compensates, to some extent, for the increased delay of the high-Vt switches.

$arch3$

This is a heterogeneous architecture in which there are two kinds of CLBs in the FPGA. Table 5.1 shows the leakage savings for this architecture. This architecture takes advantage of the fact that a high percentage of subblocks are assigned high-Vt in the ideal case and hence there can be many CLBs with all high-Vt subblocks. Since the FPGA architecture needs to be regular in structure the two kinds of CLBs are distributed uniformly throughout the array. However, this does not affect the CAD tool for the final mapping of the application. Based on the clustering results, $type1$ CLBs was assumed to be 50% of total CLBs, and $type2$ CLBs had 10 low-Vt subblocks. Table 5.1 shows that overall mean leakage savings for this architecture is 19.3%. The mean delay penalty for this architecture is 6.9%.

$arch4$

This architecture is realized using the architecture $arch3$. The base architecture is same as that of $arch3$, and over that architecture routing resources are also assigned high-Vt to realize the architecture $arch4$. The evaluation methodology is same as that for $arch3$, except that the fraction of high-Vt routing resources is varied and the leakage savings and design tradeoffs are calculated. Fig. 5.6(a) shows the case when only the

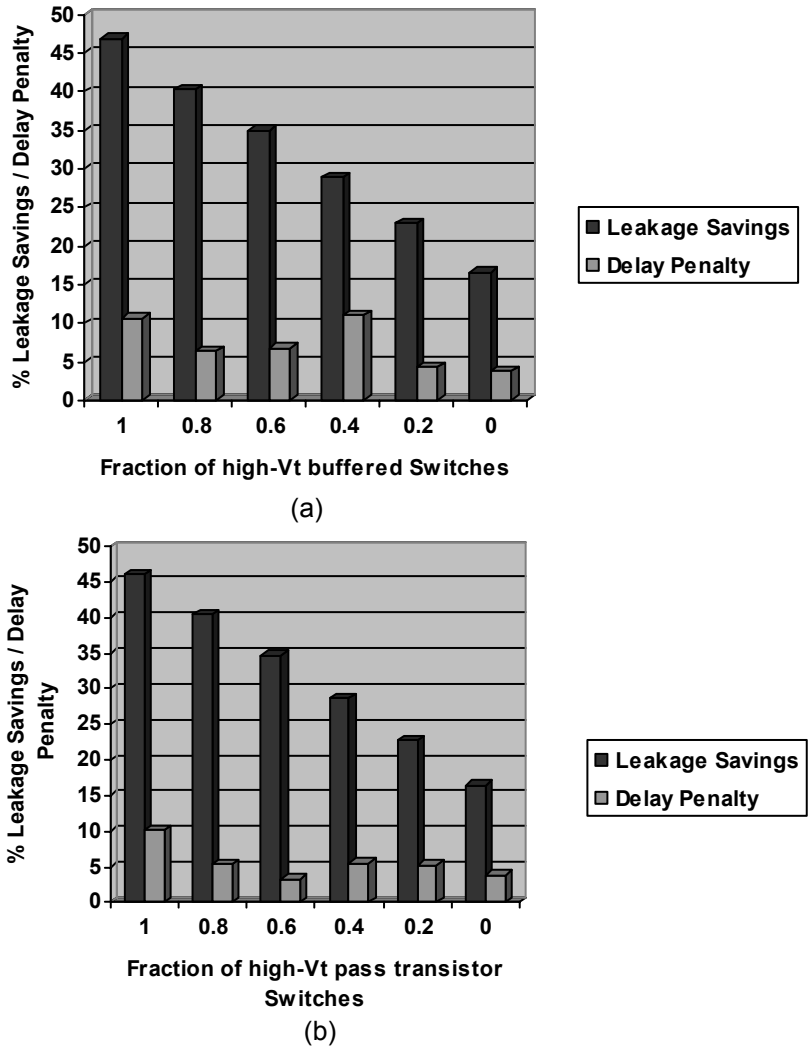

Figure 5.5: Leakage savings for $arch2$ with 6 high-Vt subblocks per CLB (a) Buffered switches assigned high-Vt (b) Pass transistor switches assigned high-Vt

buffered switches are considered for high-Vt assignment along with the CLB output pin switches which can drive these segments (i.e., those segments which can be driven by these high-Vt buffered switches), all the pass transistor switches remaining low-Vt. Fig. 5.6(b) shows the case when pass transistor switches are considered for high-Vt assignment along with the CLB output pin switches which can drive these segments (i.e., those segments which can be driven by these high-Vt pass transistor switches), all the buffered switches remaining low-Vt. The delay penalties vary between 12.4% and 4.8% for different fractions of high-Vt switches. The increase in delay penalty is not too large when routing switches are assigned high-Vt because a large fraction of routing resources are unutilized. It can be seen that the leakage savings for this architecture is about 3% greater than the leakage savings for the $arch2$ (6 high-Vt subblocks per CLB) with almost the same delay penalty.

5.5.3 Design tradeoffs

Before deciding upon any dual-Vt FPGA architecture, it is important to consider the design tradeoffs. In this work two types of architectures were evaluated - homogeneous and heterogeneous. The design tradeoffs for this approach are the delay penalties, and the impact on overall power. These were evaluated for all the FPGA architectures. It was observed that the dynamic power dissipation remains almost constant for all the architectures.

A dual-Vt custom VLSI design does not lead to any delay penalty. However, because of the very nature of programmability of the FPGA, there would be some delay penalty associated with a circuit implemented on a dual-Vt FPGA as compared to single low-Vt FPGA. The delay penalties would vary with the benchmark. Essentially, the delay penalty occurs because the critical path of a circuit in the dual-Vt FPGA changes from that in the single low-Vt FPGA. There are three sources of delay penalties in a dual-Vt FPGA :

1. The increased delays of the subblocks, some of which might lie on the critical path.
2. The altered placement and routing in the dual-Vt FPGA because of the different delays through the subblocks which impacts the placement and subsequently routing.
3. The increased delays of high-Vt routing switches some of which might lie on the critical path.

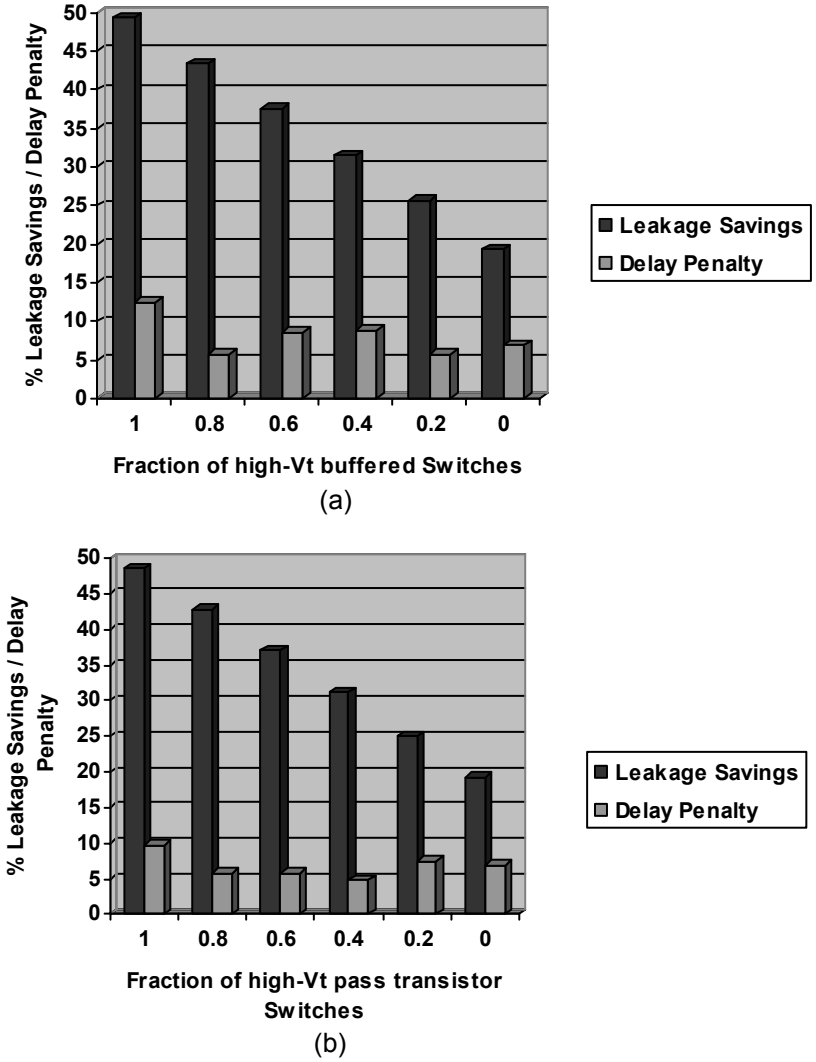

Figure 5.6: Leakage savings for $arch4$ (a) Buffered switches assigned high-Vt (b) Pass transistor switches assigned high-Vt

Table 5.2 shows the leakage savings and the delay penalties for $arch2$, $arch4$, and the *all high-Vt subblocks* architectures. It can be seen that the leakage savings for $arch2$ (6 high-Vt subblocks) with 80% high-Vt pass transistor switches is approximately 3% less than the leakage savings for the $arch4$ with 80% high-Vt pass transistor switches. The delay penalties for both these architectures are comparable. These architectures have a very high percentage of high-Vt switches and would be most susceptible to performance degradation. However, the mean delay penalty for $arch2$ with 80% high-Vt pass transistor switches is 6.4% with the maximum delay penalty being 15.3% for a mean leakage savings of 40.3%. For $arch4$ with 80% high-Vt switches the mean delay penalty is 5.8% with the maximum delay penalty being 13.6% for a mean leakage savings of 43.5%. The results of all high-Vt architecture, where all the subblocks in all the CLBs are high-Vt, have been shown for comparison with other architectures. The mean leakage savings is 32.6%, the mean delay penalty being 19%, with the maximum delay penalty as high as 48.5%. It can be seen that keeping all high-Vt subblocks leads to severe performance degradation in many of the benchmarks. This is because in the architectures which contain low-Vt subblocks, the placer and the router can search for low-Vt subblocks to minimize the delay, which is not possible in the case of architecture with all high-Vt subblocks. The performance degradation in the all high-Vt subblocks architecture is more pronounced for benchmarks which have large number of subblocks on the critical path. The benchmark diffeq which has a severe delay penalty of 48.5%, has 30 subblocks on the critical path for the implementation on the all high-Vt subblocks architecture. The benchmark bigkey has only 6 subblocks on the critical path and the routing of the VPR compensates for the increased delay and which leads to almost no delay penalty. Hence, having an architecture with all high-Vt subblocks would not be a good design, as expected.

There is almost no area penalty as the dual-Vt technique, inherently does not lead to any area penalty. In the experimental results for all the benchmarks, the routing channel width and the number of CLBs were same for both the baseline single-Vt FPGA architecture and the dual-Vt FPGA architectures. This shows that there is no area penalty associated with the proposed design of the dual-Vt FPGA architectures.

5.5.4 Distribution of Leakage Savings

In all these evaluations it was assumed that high-Vt SRAM cells are used for both the single low-Vt implementation and dual-Vt implementations. The leakage savings is solely from the logic and routing parts of the FPGA. Fig. 5.7 shows the distribution of leakage among the logic blocks and the routing resources. Fig. 5.7(a) shows the leakage distribution in the single low-Vt implementation for the benchmark alu4. It can be seen

Table 5.2: Design tradeoffs for homogeneous architecture $arch2$ with 6 high-Vt subblocks and 80% high-Vt pass transistor switches, heterogeneous architecture $arch4$ with 80% high-Vt pass transistor switches, and all high-Vt subblocks architecture.

Benchmark	arch2		arch4		all high-Vt subblocks	
	Leakage Savings	Delay Penalty	Leakage Savings	Delay Penalty	Leakage Savings	Delay Penalty
alu4	40.6%	10%	43.43%	7.5%	32.5%	16%
apex2	40.4%	2%	44.8%	8.4%	43.2%	13.6%
bigkey	41.34%	3.1%	42.8%	1%	24.2%	1%
clma	41%	4.9%	44.2%	6.2%	41%	6.2%
des	41.3%	11.8%	43.5%	1%	25.5%	22.6%
diffeq	40.7%	5.6%	43.9%	1%	31.7%	48.5%
dsip	40.9%	1%	43.1%	1%	22.6%	13.8%
elliptic	40.3%%	10.5%	43.5%	9%	41.2%	31%
ex5p	40%	11.9%	42%	9.2%	25.7%	18.1%
frisc	39.3%	7.1%	42.8%	5.3%	39.8%	21.4%
misex3	40.4%	4.1%	41.7%	10.8%	39.4%	13.6%
pdc	37.1%	15.3%	38.2%	6%	37.9%	25%
s298	41%	8.6%	43.6%	13.6%	37.1%	25%
s38584.1	41.3%	5.4%	44.4%	6.4%	27.9%	28.8%
seq	41.6%	1%	44%	0.02%	36%	1.1%
tseng	40.9%	2%	44%	8.2%	28.3%	42.6%

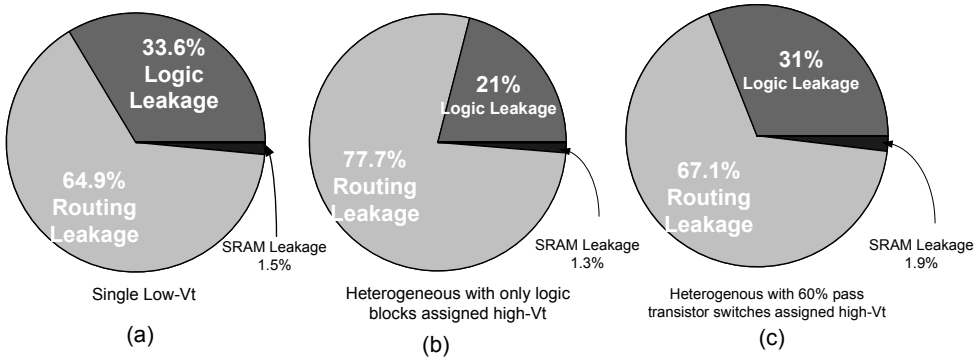

Figure 5.7: (a) Leakage contributions of routing resources, logic resources and SRAM cells for single low-Vt implementation, (b) and (c) after dual-Vt implementation for alu4.

that almost two thirds of leakage is from the routing resources. This is because of a large amount of unused routing resources [4]. SRAM leakage is negligible as they have high-Vt to reduce the leakage. Assigning dual-Vt to the logic blocks reduces the leakage component of the logic blocks as shown in Fig. 5.7(b). Fig. 5.7(c) shows the impact on leakage distribution because of dual-Vt assignment to both the logic blocks and the routing resources. The contribution of routing leakage to total leakage increases by a small amount, even though it results in overall leakage savings. This can be explained by the leakage power results for a benchmark alu4 as shown in Fig. 5.8. For architecture $arch3$, the logic leakage reduces by more than 50%, with the routing leakage remaining the same. When 60% of pass transistors switches are also assigned high-Vt, the routing leakage decreases by 28%. This results in an overall leakage savings of 37%, with 19% savings from the logic blocks and 18% savings from the routing resources.

The leakage power model in [10] models both the gate and the subthreshold leakage. For CMOS 130nm, the gate leakage is 6 orders of magnitude smaller than the subthreshold leakage. Therefore, the gate leakage does not impact the methodology proposed in this work.

5.6 Designing a Dual-Vt FPGA

Designing a dual-Vt FPGA would require selecting a dual-Vt FPGA architecture and then determining the parameters for the dual-Vt FPGA. The dual-Vt FPGA CAD framework proposed in this work is intended for this purpose. Fig. 5.9 shows the steps involved in the design of a dual-Vt FPGA. The previous sections explored the various dual-Vt FPGA architectures for leakage power savings and performance tradeoffs. It can

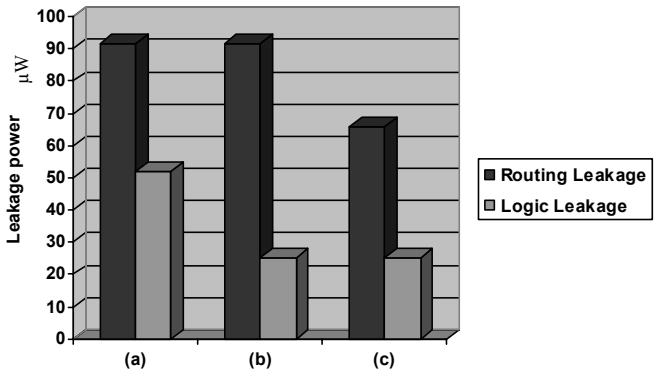

Figure 5.8: Leakage power for alu4. (a) Single low-Vt implementation (b) dual-Vt $arch3$ (c) dual-Vt $arch4$ with 60% high-Vt pass transistor switches

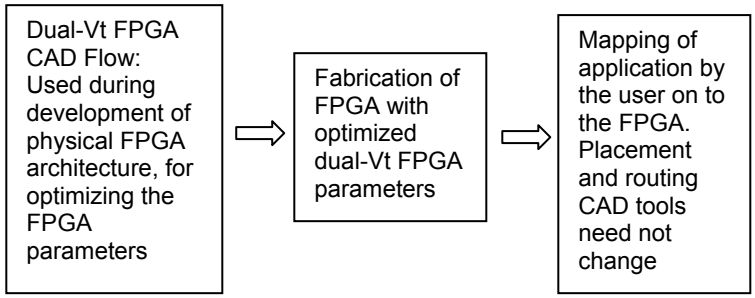

Figure 5.9: Realizing a dual-Vt FPGA design

be seen from the results of the previous section that architectures with high-Vt switches lead to significantly increased leakage savings with small increase in delay penalties. Further, some of the architectures are more suitable for some benchmarks, whereas other benchmarks are more suitable for another architecture. This is because of the number of interconnections between the logic blocks are different for different benchmarks. This implies that for some benchmarks having a large number of high-Vt switches would not have a large impact on delay because of shorter total wire length in the critical path, whereas for benchmarks having a high percentage of high-Vt subblocks per CLB would not have a big impact on performance because there are very few subblocks in the critical path. homogeneous architectures would, in general, be easier in implementation as compared to heterogeneous architecture. However, a careful design of heterogeneous architectures and the associated CAD tools can lead to increased leakage savings for same delay penalties as that for homogeneous architectures.

The above discussion leads to what can be called as Field Programmable System-on-Chip for implementing complete systems. The low-leakage FP-SoC can be made up of a combination of the above architectures. Fig. 5.10 shows one such design of a Field Programmable SoC. This architecture is a combination of heterogeneous, homogeneous, all high-Vt and all low-Vt architectures, where the complete chip is divided into parts, each having a different design. The extremely critical parts of the circuit can be implemented on the all low-Vt part of the FPGA, whereas the part of the circuit in which performance is not a concern, can be implemented on the all high-Vt part of the FPGA. Other parts of the circuit can be implemented on the homogeneous or heterogeneous architectures, depending on performance tradeoffs.

5.7 Summary

This chapter presented dual-Vt FPGA architectures and a dual-Vt FPGA CAD framework for designing the dual-Vt FPGA architectures. Two kinds of architectures are explored, homogeneous and heterogeneous architectures. Logic blocks and routing resources are considered for dual-Vt assignment. Results indicate that leakage savings of upto 50% can be obtained.

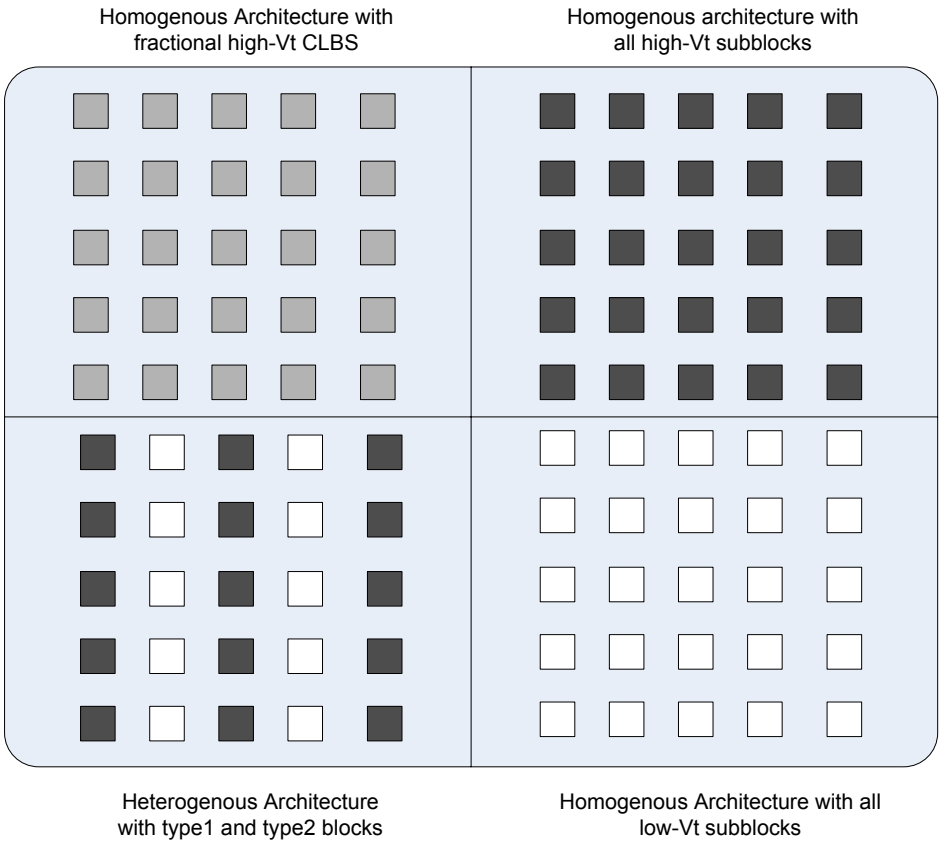

Figure 5.10: A low-leakage Field Programmable SoC

Chapter 6

Conclusions and Future Work

This work focused on leakage power, a critical design challenge for deep sub-micron FPGAs. With the emergence of highly complex FPGAs, capable of implementing complete systems, it has become imperative to develop innovative solutions for leakage power mitigation in FPGAs.

This work developed an analytical state dependent leakage power model for accurately computing the leakage power. Subthreshold and gate leakage were modeled since they are the major sources of leakage power in the current generation technologies. The computation of leakage based on the probability of states gives more accurate results compared to the results presented in [11]. The leakage computation engine (LCE) models the dominant short channel effects as well. The leakage power model was developed based on physical and empirical equations for the devices, making it technology independent and can be used for rapidly analyzing different FPGA architectures across different technology nodes. A leakage analysis for FPGAs using an industrial CMOS 130nm and CMOS 90nm was presented. The results indicate that the leakage power in CMOS 90nm FPGA is almost 4 times larger than the leakage in the CMOS 130nm FPGA, which is as expected.

The dual-Vt FPGA architectures explored indicate that on an average leakage power savings of up to 50% can be obtained by the dual-Vt FPGA architectures. We proposed a dual-Vt FPGA CAD flow for implementation and evaluation of dual-Vt FPGA architectures. The results show that a high percentage of subblocks are assigned high-Vt for the ideal dual-Vt assignment. This indicates that a large amount of slack is available with the logic blocks which can be exploited to reduce the leakage power. We explored two primary architectures, homogeneous and heterogeneous. Results indicate that there is a large percentage of unused routing and logic resources such that dual-Vt FPGA architectures can lead to good leakage savings without large delay penalties.

The leakage power model developed for FPGAs takes into account the state dependency of leakage power. However, the primary inputs were considered independent. The future work for the enhancement of the leakage power model would be to consider the correlation among the inputs. In addition, in sub-100nm designs, process and environment variations have become an important design consideration. The model can be enhanced to compute leakage power under variations.

The main bottleneck in using any leakage reduction technique for FPGAs is the very nature of programmability of the FPGA, leading to an architecture not very compatible with leakage reduction techniques, designed specifically for ASICs. For our future work we intend to develop novel leakage aware FPGA architectures which would be especially adapted to implement leakage reduction techniques without incurring large delay penalties. Process and environment variations also affects the design of low-leakage FPGAs, as the leakage power is impacted because of variations. The design challenges for the dual-Vt FPGAs under process and environment variations can be analyzed to account for the variations in the design phase. These would essentially relate to the architectural changes, CAD tool enhancements, and/or development of new CAD tools.

Bibliography

[1] L. Wei, Z. Chen, K. Roy, Mark C. Johnson, Y. Ye, and Vivek K. De, "Design optimization of dual-threshold circuits for low-voltage low-power applications," *IEEE Trans. VLSI, Vol. 7, pp. 16-24, March 1999.*

[2] J. Kao, S. Narendra, and A. Chandrakasan, "Subthreshold leakage modeling and reduction techniques," *IEEE Int. Conf. on Computer-Aided Design, pp. 141-148, 2002.*

[3] Jason H. Anderson, F.N. Najm, and T. Tuan., "Active leakage power optimization for FPGAs," *FPGA, pp. 33-41, 2004.*

[4] T. Tuan, B. Lai, "Leakage Power Analysis of a 90nm FPGA," *IEEE Custom Integrated Circuits Conf., pp. 57-60, 2003.*

[5] S. Sirichotiyakul, T. Edwards, C. Oh, R. Panda, D. Blaauw, "Duet: An accurate leakage estimation and optimization tool for dual-Vt circuits," *IEEE Trans. VLSI, Vol.107, pp. 79-90, April 2002.*

[6] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "Reducing leakage energy in FPGAs using region-constrained placement," *FPGA, pp.51-58, 2004.*

[7] A. Rahman and Vijay Polavarapuv, "Evaluation of low-leakage design techniques for Field Programmable Gate Arrays," *FPGA, pp. 23-30, 2004.*

[8] F. Li, D. Chen, L. He, J. Cong, "Architecture evaluation for power efficient FPGAs," *FPGA, pp. 175-184, 2003.*

[9] V. Betz, J. Rose and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, MA 1999, ISBN: 0792384601

[10] A. Kumar and M. Anis, "An Analytical State Dependent Leakage Power Model for FPGAs," *DATE, 2006, (in press).*

[11] K. Poon, A. Yan and S.J.E. Wilton, "A flexible power model for FPGAs," *International Conf. on Field Programmable Logic and Applications, pp. 312-321, 2002.*

[12] Shekhar Borkar, "Design challenges of technology scaling," *Micro, IEEE, Vol. 19, pp. 23-29, 1999.*

[13] K. Roy, S. Mukhopadhyay, H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proc. IEEE, Vol. 91, pp. 305-327, 2003.*

[14] M. Anis, M. Mahmoud, M. Elmasry, S. Areibi, "Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique,"*DAC, pp. 480-485, 2002.*

[15] J. Kao, A. Chandrakasan, D. Antoniadis, "Transistor sizing issues and tools for Multi-threshold CMOS technology," *DAC, pp. 409-414, 1997.*

[16] S. Mutah, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, J. Yamada, "1-V power supply high speed digital circuit technology with multi-threshold voltage CMOS," *IEEE Journal of Solid State Circuits, vol. 30, no. 8, pp. 847-853, 1995.*

[17] N. Kato, Y. Akita, M. Hiraki, T. Tamashita, T. Shimizu, F. Maki, K. Yano, "Random modulation: Multi-threshold-voltage design methodology in sub-2-V power suppy CMOS," *IEICE Trans. Electron., vol. E83-C, no. 11, pp. 1747-1754, 2000.*

[18] L. Wei, K. Roy, C. Koh, "Power minimization by simultaneous dual-Vth assignment and gate-sizing," *IEEE Custom Integrated Circuits Conf., pp. 413-416, 2000.*

[19] Y-F. Tsai, D. Duarte, N. Vijaykrishnan, M. J. Irwin, "Implications of technology scaling on leakage reduction techniques," *DAC, pp. 187-190, 2003.*

[20] E. M. Sentovich et al., "SIS: A system for sequential circuit analysis," *University of California, Berkeley, 1992.*

[21] J. Cong and Y. Ding, "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Trans. CAD, pp. 1-12, 1994.*

[22] Fei Li, Yan Lin, Lei He, and Jason Cong , "Low-power FPGA using predefined dual-Vdd/dual-Vt fabrics," *FPGA, pp. 42-50, 2004.*

[23] Guy Lemieux and David Lewis,*Design of Interconnection Networks for Programmable Logic*, Kluwer Academic Publishers, 2004, ISBN: 1-4020-7700-9

[24] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep submicron FPGA performance and density," *FPGA, pp. 3-12, 2000*

[25] BSIM4 Models, University of California, Berkeley. Available online: *http://www-device.eecs.berkeley.edu/ bsim3/bsim4.html*

[26] D. Lee, D. Blaauw, D. Sylvester, "Gate oxide leakage current analysis and reduction for VLSI circuits," *IEEE Trans. on VLSI, Vol. 12, pp. 155-166, Feb. 2004*

[27] A. Sangiovanni-Vincentelli, A. El Gamal, and J. Rose, "Synthesis methods for Field-Programmable Gate Arrays," *Proceedings of IEEE, pp. 1057-1083, July 1993*

[28] R. Brayton, G. Hachtel, and A. Sangiovanni-Vincentelli, "Multilevel logic synthesis," *Proceedings of IEEE, pp. 264-300, Feb 1990*

[29] J. Cong, J. Peck, and Y. Ding, "RASP: A General Logic Synthesis System for SRAM-based FPGAs," *FPGA, pp. 137-143, 1996*

[30] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing," *Science, pp. 671-680, May 13, 1983*

[31] M. Huang, F. Romeo, and A. Sangiovanni-Vincentelli, "An efficient general cooling schedule for simulated annealing," *ICCAD, pp. 381-384, 1986*

[32] W. Swartz and C. Sechen, "New algorithms for placement and routing of macro cells,"*ICCAD, pp. 336-339, 1990*

[33] A. Marquardt, V. Betz, and J. Rose, "Timing-Driven Placement for FPGAs," *FPGA, pp. 203-213, 2000*

[34] C. Ebeling, L. McMurchie, S. A. Hauck and S. Burns, "Placement and routing tools for Triptych FPGA," *IEEE Trans. on VLSI, pp. 473-482, Dec. 1995*

[35] K. Roy and S. C. Prasad,*Low Power CMOS VLSI Circuit Design*, Wiley-Interscience, John Wiley & Sons, 2000, ISBN: 0-471-11488-X

[36] V. George, H. Zhang, and J. Rabaey, "The design of a low energy FPGA," *ISLPED, pp. 188-193, 1999*

[37] E. Kusse and J. Rabaey, "Low-Energy embedded FPGA structures," *ISLPED, pp. 155-160, 1998*

[38] A. Singh and M. Marek-Sadowska, "Efficient circuit clustering for area and power reduction in FPGAs," *FPGA, pp. 59-66, 2002*

[39] A. Gayasen, K. Lee, N. Vijayakrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "A Dual-Vdd low power FPGA architecture," *FPL, August 2004*

[40] V. Betz, *VPR and T-Vpack User's Manual (Version 4.30)*, Available online: *http://www.eecg.toronto.edu/ vaughn/vpr/vpr.html*

[41] M. Anis and M. Elmasry, *Multi-Threshold CMOS Digital Circuits: Managing Leakage Power*, Kluwer Academic Publishers, MA 2003, ISBN: 1-4020-7529-4

[42] J. Lamoureux and S. J. E. Wilton, "On the interaction between power-aware FPGA CAD algorithms,"*ICCAD, pp. 701-708, 2003*

[43] J. H. Anderson and F. N. Najm, "Power-Aware Technology Mapping for LUT-Based FPGAs,"*FPT, pp. 211-218, 2002*

[44] F. Li, Y. Lin, and L. He, "FPGA Power Reduction Using Configurable Dual-Vdd,"*DAC, pp. 735-740, 2004*

[45] J. H. Anderson and F. N. Najm, "Low-Power Programmable Routing Circuitry for FPGAs," *ICCAD, pp. 602-609, 2004*

[46] International Technology Roadmap for Semiconductors, 2003. *Available online: http://public.itrs.net*

Appendix A

List of publications from this work

1. A. Kumar and M. Anis, "An Analytical State Dependent Leakage Power Model for FPGAs," *ACM/IEEE Design Automation and Test in Europe, Munich, Germany, 2006 (accepted for publication).*

2. A. Kumar and M. Anis, "Dual-Vt design of FPGAs for subthreshold leakage tolerance" *IEEE International Symposium on Quality Electronic Design, San Jose, USA, 2006 (accepted for publication).*

3. A. Kumar and M. Anis, "Dual-Vt FPGA design for leakage power reduction," *ACM International Symposium on FPGAs, Monterey, USA, 2005 (Abstract-Poster).*

4. A. Kumar and M. Anis, "Dual threshold CAD framework for subthreshold leakage power aware FPGAs," *IEEE Trans. on CAD, (accepted after revision).*

5. A. Kumar and M. Anis, "Dual-Vt FPGA design for leakage power reduction," *Proc. of Micronet Annual Workshop, pp. 51-52, Ottawa, Canada, May 2005.*