POWER ESTIMATION FOR FIELD PROGRAMMABLE GATE ARRAYS

by

Kara Ka Wing Poon

B.A.Sc, University of British Columbia, 1999

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

The Faculty of Graduate Studies

Department of Electrical and Computer Engineering

We accept this thesis as conforming to the required standard:

The University of British Columbia

August 2002

ABSTRACT

POWER ESTIMATION FOR FIELD PROGRAMMABLE GATE ARRAYS

Power dissipation is becoming a major concern for semiconductor vendors and customers. Compared to ASICs and other custom chips, Field Programmable Gate Arrays (FPGAs) have long routing tracks with significant parasitic capacitance, and dissipate a significant amount of power at high frequencies. Previous work has presented point solutions, which are applicable only to particular architectures [24][36][65]. Other work has proposed numerous CAD algorithms that focus primarily on reducing switching activity to achieve low power [29][44][57][61][68]. To thoroughly investigate the power consumption within FPGAs, there is a need for a universal power model capable of estimating power for a wide variety of programmable logic architectures.

This thesis describes a power model that estimates the dynamic, short circuit, and leakage power for a wide variety of FPGA architectures. This power model has been integrated into the *Versatile Place and Route (VPR)* CAD tool, widely used software for FPGA architectural studies. This thesis also investigates the impact of various architectural parameters on the power-efficiency of FPGAs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

It has been a great pleasure for me to study for my Master degree at the University of British Columbia. I am so thankful that I have a wonderful supervisor, Dr. Steven Wilton. His encouragement and invaluable advice really motivated me to work hard on this thesis project. I am impressed by Dr. Wilton's enthusiasm about his teaching and his research.

This work is supported by Micronet, the British Columbia Advanced Systems Institute, and the Natural Sciences and Engineering Research Council of Canada. Their support is greatly appreciated. Also, I would like to thank Dr. Vaughn Betz for providing VPR CAD tool, and Dr. Resve Saleh, Dr. F.N. Najm and Li Shang for their helpful discussions.

Within these past two years, I worked with a great research team – Andy, Danna, Ernie, Kim, Julien, Martin, Noha, Rebecca, Steve, and Tony. I really treasure their insightful opinions on my research work. Moreover, I am so grateful to be part of the SOC Research Group at UBC. All the team members are energetic and supportive. I would like to thank Gary, Louis, James, Marwa, Ronald, Victor, and Zahra for sharing their ideas and their senses of humor with me. They have made my stay at UBC so unforgettable.

Finally, I dedicate this thesis to my parents, Patrick and Patty Poon, for their continuous support. Also, I would like to thank my greatest brother, Kevin, for his incredible amount of encouragement. Last but not least, I am gratified to have Alfred, who is my source of motivation and support throughout my academic years at UBC.

Chapter 1

1 INTRODUCTION

1.1 Motivation

Power dissipation is becoming a major concern for semiconductor vendors and customers. Figure 1.1 and Table 1.1 illustrate the projection of allowable maximum power consumed by three types of semiconductor devices from the *International Technology Roadmap for Semiconductors 2001*[30]:

Table 1.1 Power management system in semiconductor devices

Semiconductor Device Type	**Power Management**
high-performance (such as desktop)	Heat sink is permitted on the package
cost-performance (such as laptop)	Only economical power management systems are provided
handheld devices	Power is limited by the battery and no cooling system is included

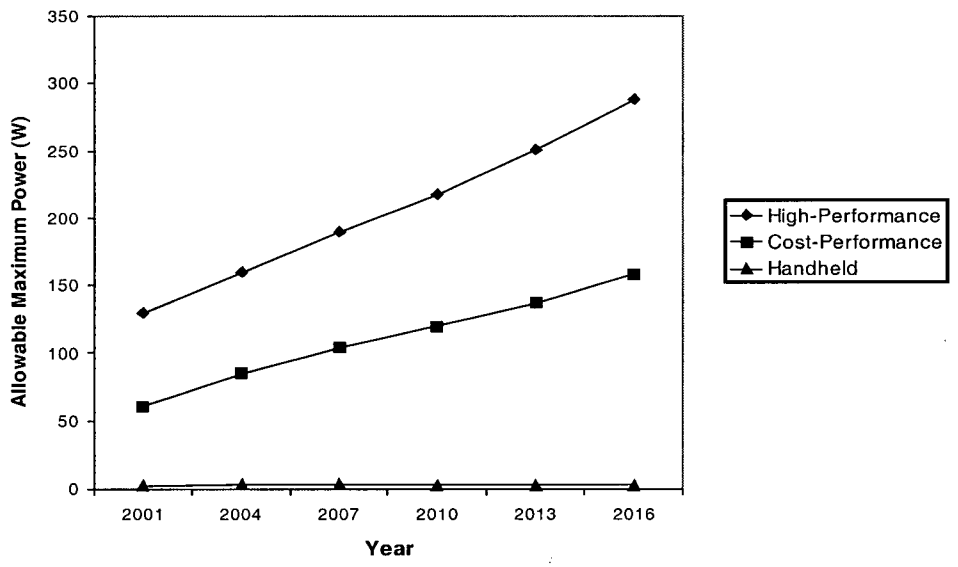


Figure 1.1 Projection of allowable maximum power for semi-conductor devices (from[30])

As shown in the graph, the allowable power dissipation of high-performance and cost-performance devices will double by 2016. However, if current design trends continue, a typical microprocessor (MPU) will consume 50 times more power than that can be supported by cost-effective packaging techniques by 2016 [3]. Power, instead of design complexity, has become the most important design consideration in today's process generation [45].

Power is especially a concern for Field-Programmable Gate Arrays (FPGAs). The post-fabrication flexibility provided by these devices is implemented using a large number of pre-fabricated routing tracks and programmable switches. These tracks can be long, and can consume a significant amount of power every time they switch. In addition, the programmable switches add capacitance to each track; this further increasing the power dissipation of FPGAs. Finally, the generic logic structures that are at the heart of every FPGA consume more power than the dedicated circuitry that would be found on an ASIC. For all these reasons, FPGA vendors have indicated that power is one of the primary concerns of their customers.

There has been a modest amount of work developing low-power FPGA architectures and FPGA CAD algorithms that optimize for low power [24][29][36][43][56][65]. Each of these previous studies, however, has presented "point solutions" for specific FPGA architectures or specific FPGA CAD programs. In addition, these works tend to use fairly crude models to estimate the power savings, and often don't take into account many important design details that may negate any advantages claimed by the proposed techniques.

Our long-term goal is to understand and investigate the effects of various architectural and CAD tool optimizations on the power consumed by FPGAs. As a first step in this effort, this thesis presents a *detailed power model* for FPGAs. The power model is flexible, in that it can be

used to estimate power in a wide variety of FPGA architectures. It is *fast*, in that estimates can be obtained without the time-consuming computation of programs such as SPICE. Also, the model gives good *fidelity*; although there may be significant absolute errors in the power estimation, the relative comparisons between two alternative architectures or algorithms will be close to the relative errors that would be obtained by the actual devices and CAD tools. We have made this model publicly available, and expect that it will be an invaluable tool for future FPGA architecture and CAD tool studies.

1.2 Research Goals

There are two objectives of this research:

1. To design a power model that can be incorporated into the popular Versatile Place and Route (VPR) CAD suite. As described above, the model must be flexible enough to target FPGAs with different look-up-tables (LUT) sizes, different interconnect strategies (segment length, switch block type, connection flexibility), different cluster sizes (for hierarchical FPGA), and different process technologies. The model must take into account dynamic power, short-circuit power, and leakage power. It also must be fast enough to be used in the inner loop of FPGA architectural and CAD tool experiments.

2. To use the power model to investigate the impact of various architectural alternatives on the power consumption of FPGAs. This will serve as an example of how the power model can be used, and the results will be directly applicable to the design of next-generation FPGAs.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 provides an overview of FPGA architectures and the CAD flow for logic synthesis. It also describes previous work on power estimation techniques. Chapter 3 describes the original and modified framework of VPR, the parameterized architectural model used in VPR along with its enhancements. Chapter 4 proposes the flexible power model developed on top of the VPR CAD tool. It also provides an explanation and evaluations of each component inside the architecture. Chapter 5 presents the analysis of the architectural impacts on the energy consumption within the FPGA. It also provides a sensitivity analysis of the primary input density assumption and the routing algorithm. Finally, the thesis concludes with a brief summary, possible directions for future research, and the contributions this project has made to the FPGA research community. Parts of this thesis have been published in [34].

Chapter 2

2 BACKGROUND AND PREVIOUS WORK

This chapter presents an overview of FPGA architectures, followed by a description of the CAD flow for mapping user circuits onto FPGAs. It also provides a background on power estimation methods, along with a discussion of the available power estimation tools in the commercial market. Finally, the focus and contributions of this research project are presented.

2.1 Overview of FPGA Architectures

Field-Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs) are the two main choices in today's high-capacity programmable logic device (HCPLD) market [5]. CPLDs implement logic as sums-of-products while most FPGAs contain look-up-tables to implement logic [14]. Figure 2.1 shows a taxonomy of technologies used in FPGAs and CPLDs. Antifuse-based devices cannot be erased or reconfigured after configuration. EPROM, EEPROM, and FLASH devices are reprogrammable and non-volatile. SRAM-based devices offer in-circuit reconfigurability, but are volatile [5]. This research project focuses on the SRAM-based FPGA devices.

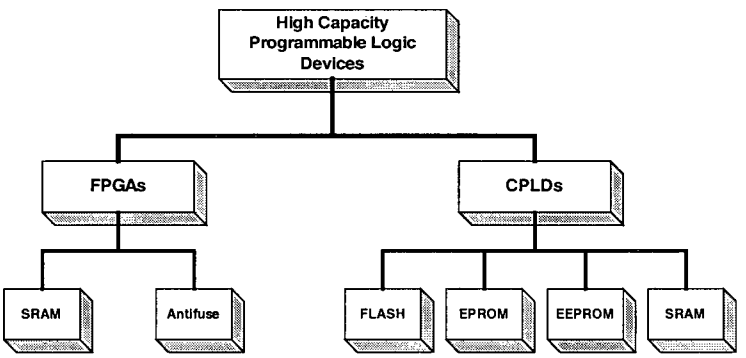
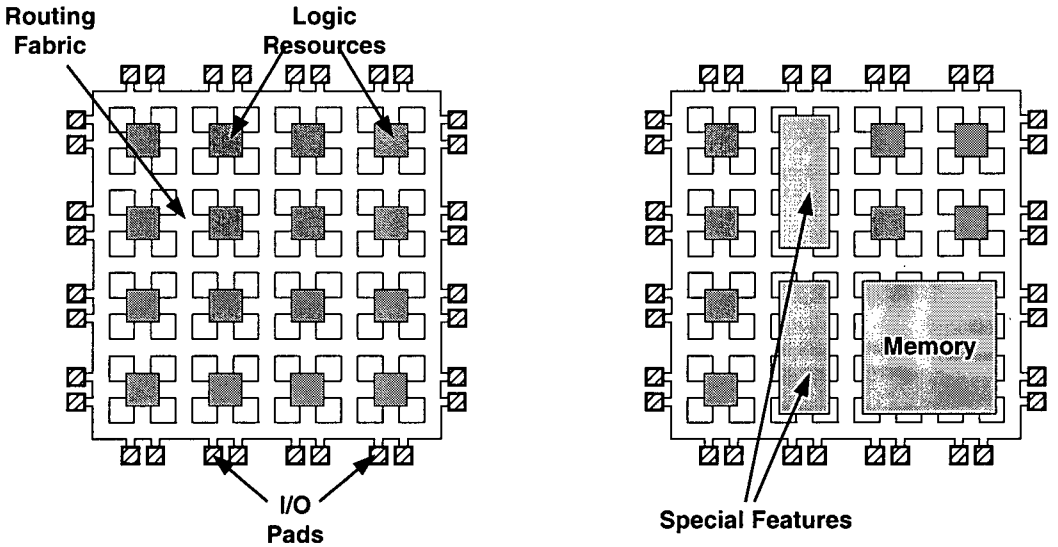


Figure 2.1 High-capacity programmable logic devices (from [5])

FPGAs were first introduced in 1985 by Xilinx. Early FPGAs contained three main components: input/output pads, logic resources, and the routing fabric as illustrated in the conceptual model in Figure 2.2(a) [14][78]. Today, vendors also embed memory and special feature blocks (which usually target communication applications) on their FPGAs as illustrated in the conceptual model of Figure 2.2(b). Table 2.1 shows a comparison among three leading SRAM-based FPGA families from Altera, Xilinx, and Lattice [6][38][77]. The maximum clock speed of today's top-of-the-line FPGAs is approximately 250Mhz, which is significantly slower than non-programmable chips. The key advantage of FPGAs, however, is their programmability and flexibility. Table 2.2 provides a brief overview of the main FPGA components. This research project concentrates on the core portion of the FPGA, which includes only the logic resources and routing fabric. Most modern FPGAs use an architecture referred as the *Island-Style Architecture*, in which logic resources are treated as *islands* surrounded by a *sea* of routing fabric. The following two subsections describe the logic and routing resources in a typical FPGA.



a) Early FPGAs (from[14]) b) Modern FPGAs

Figure 2.2 Conceptual FPGA models

Table 2.1 Comparison of three SRAM-based FPGA families

Vendor	**Altera**	**Xilinx**	**Lattice**
Model	Stratix	Virtex II Pro	ORCA Series 4
CMOS Technology (um)	0.13	0.13	0.16
Maximum Clock Speed (MHz)	250	300	250
Supply Voltage (V)	1.5	1.5	1.5
Number of look-up-tables	10, 570 to 114,140	3,168 to 125,136	4,992 to 16,192
Number of input per LUT	4	4	4
On-chip Memory (Kbits)	920 to 10,118	216 to 125,136	74 to 147
Number of I/O pins	422 to 1310	204 to 1200	400 to 720
Number of global networks	Up to 16	Up to 12	Up to 8
Special Features	DSP blocks	PowerPC processors	Embedded Microprocessor Interface

Table 2.2 FPGA components

Component	**Function**
Input or Output Pad	An interface for the communication with off-chip electronic components
Logic Resources	A piece of programmable resource used to implement logic functions
Routing Fabric	Connections between logic blocks, on-chip memory, and embedded specialty blocks
Memory	RAM-based on-chip memory, which is either distributed or included as individual blocks
Special Features	Special components, such as embedded processor, and digital signal processing (DSP) blocks

2.1.1 Logic Resources

Most commercial FPGAs implement logic using look-up-tables (LUTs). Each k-input LUT contains 2^k configuration bits, in which a desired truth table is programmed to perform any k-input logic function [14]. A majority of FPGA devices use 4-input LUTs as the basic logic resource. Several LUTs, flip flops, and multiplexers are grouped together into a *logic element.* Logic elements are then clustered to form a *logic block.* Such an architectural arrangement allows local interconnects between the logic elements to be optimized. Figure 2.3 illustrates a hierarchical view of a typical logic block. In the Altera Stratix devices, each logic elements

contain one LUT and ten logic elements are linked together to become a *Logic Array Block (LAB)* [6]. In the Xilinx Virtex II-Pro devices, two LUTs are contained in a logic element and four logic elements are joined to form a *Configurable Logic Block (CLB)* [77]. In ORCA series 4 (now produced by Lattice), eight LUTs are packed together to be a *Programmable Logic Cell (PLC)* [38].

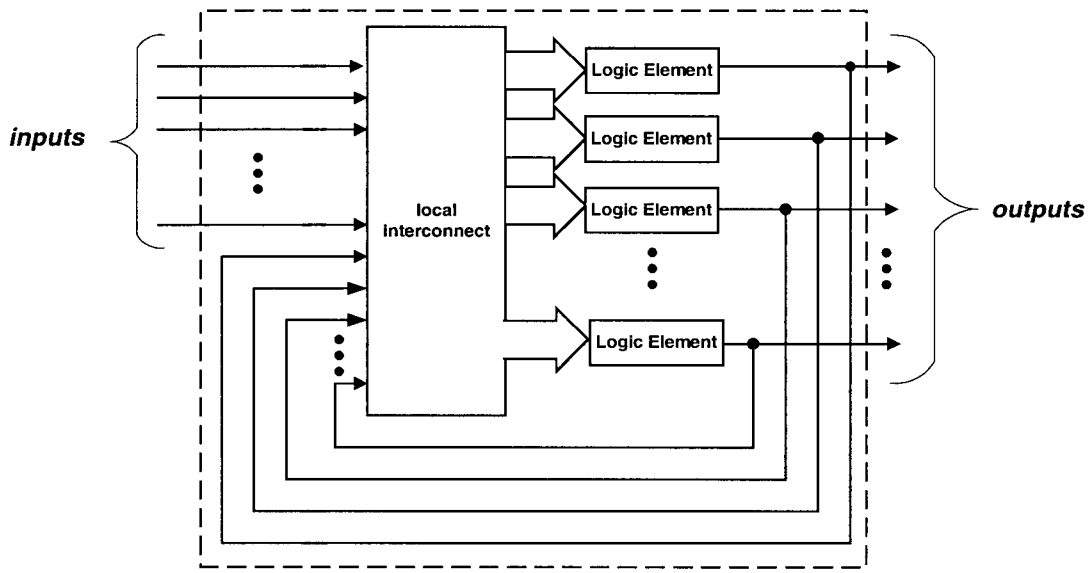


Figure 2.3 A hierarchical view of a logic block (from [10])

2.1.2 Routing fabric

The routing fabric is an essential component for FPGAs since it connects all the internal components together. The routing fabric inside a FPGA can be divided into two parts:

- *segmented local routing* for connections among logic blocks
- *dedicated routing* for global networks, such as clocks

2.1.2.1 *Segmented Local Routing*

Interconnections among logic blocks for a typical island-style architecture is illustrated in Figure 2.4. The routing resources consist of the following three main components:

- *Segmented wires* which are prefabricated and used to implement programmable connections among switch blocks, connection blocks, and logic blocks.
- *Connection blocks* to dynamically connect logic block inputs and outputs with the segmented wires
- *Switch blocks* which can be programmed to connect the horizontal and vertical segmented wires together

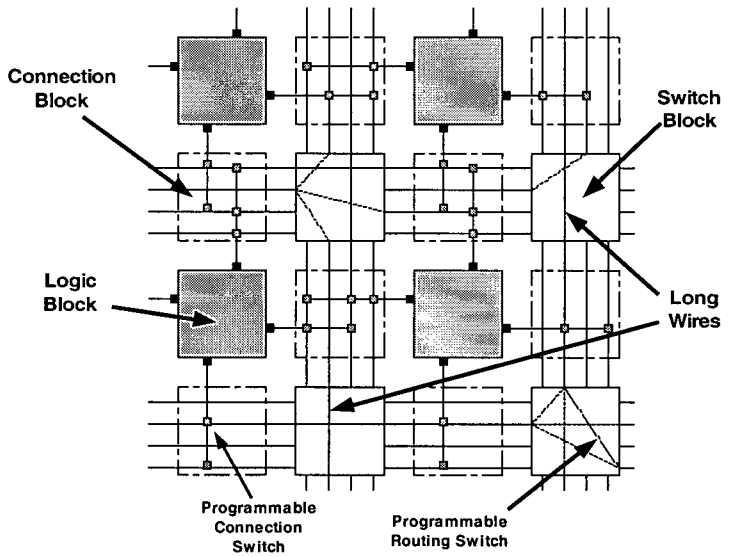


Figure 2.4 An island-style FPGA (from [10])

The number of logic blocks spanned by each routing wire is defined as the *segment length*. For example, if a wire spans four logic blocks, it has a *segment length* of four. Short wires provide routing flexibility, but buffered long wires are necessary for long connections to optimize for speed. A mixture of segmented wires with different lengths is available in commercial FPGAs [4][6][38][72][77].

2.1.2.2 *Dedicated Global Routing*

Dedicated routing tracks are provided for signals, like clock, reset, and preset, that fan out to the entire chip and require low-skew transmission. Figure 2.5 shows a typical clock distribution network. The global clock signal is distributed throughout the device in "spines and ribs" fashion. FPGA vendors determine the number of spines and ribs. Xilinx's Virtex-E has a clock distribution network with 24 spines and 12 ribs at the top and bottom of the chip while Actel's ProASIC series provide 6 to 14 spines for their clock trees [2][72]. Compared to the normal H-tree clock distribution network, the "spines and ribs" network suffers from a relatively large clock skew [23]. However, FPGA vendors claim that such a network is more flexible [2][72]. To compensate for the clock skew, commercial FPGAs have a number of phase-locked loops (PLLs) or delay-locked loops (DLLs) for clock de-skew purposes [4][6][38][72][77]. Some vendors, like Actel, recommend that users map their high-fanout critical nets to the spines in order to reduce the timing penalty [2]. Clock multiplexers and clock management circuitry are also available for disabling part of the clock network to reduce power consumption. The design of FPGA clock distribution network is beyond the scope of this research project. For simplicity, an H-tree clock distribution network is employed in the FPGA model for this project.

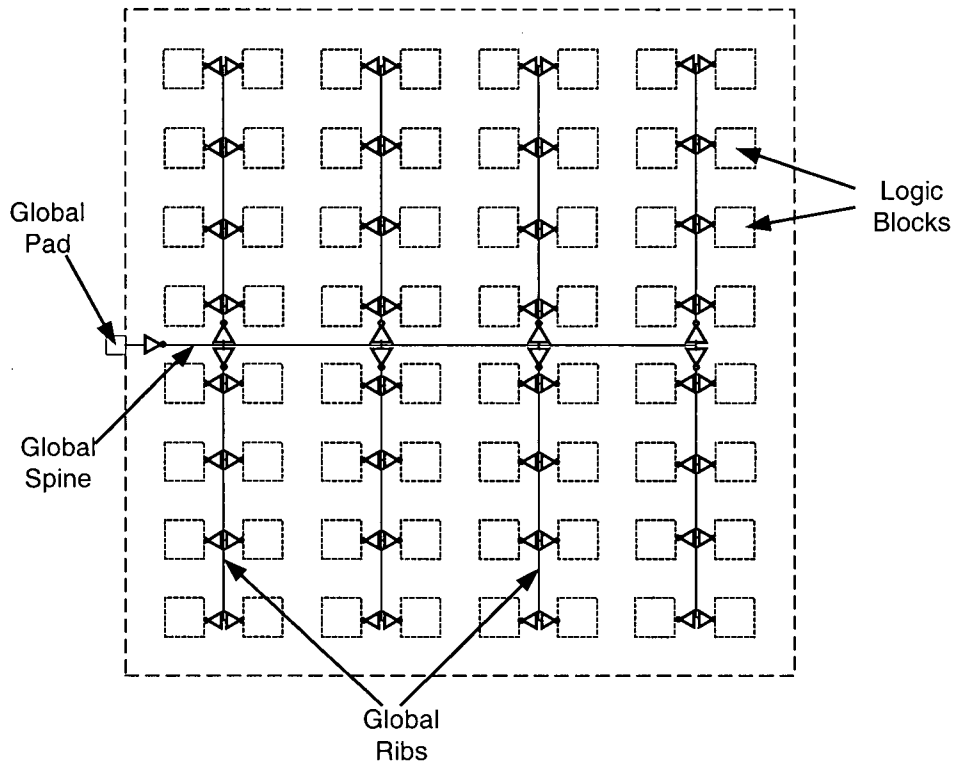


Figure 2.5 "Spines and Ribs" clock distribution network (from [2])

2.2 FPGA Computer Aided Design Flow

The computer-aided design (CAD) flow for FPGAs converts a high-level description of a circuit to a stream of programming bits for an FPGA (see Figure 2.6). FPGA users describe their designs using a schematic, or high-level description language, such as VHDL and Verilog. The circuit description is then processed into a *netlist* of basic gates during *High-level Synthesis.* The *Logic Synthesis* is the core of the CAD flow. It consists of four major steps: logic optimization, technology mapping, placement, and routing [14].

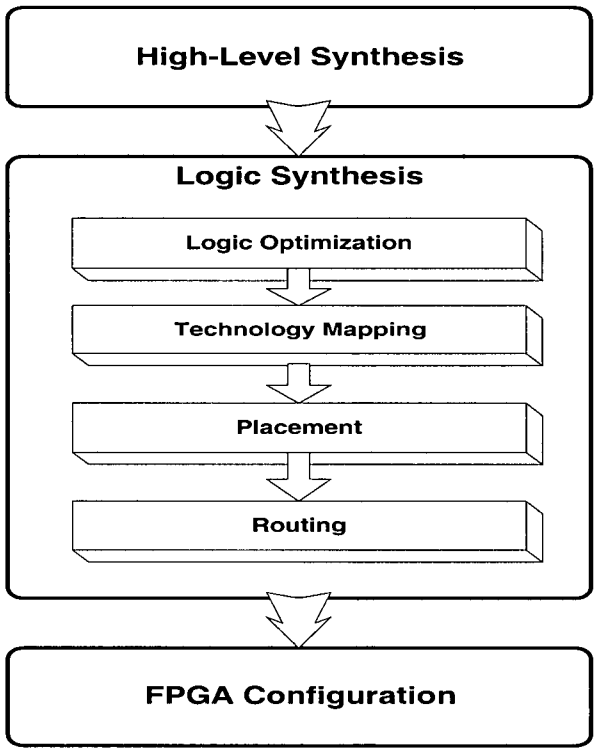


Figure 2.6 A typical FPGA CAD flow

The first step of logic synthesis is *logic optimization*. It is often referred as *technology- independent optimization* because in this phase, the goal is to simplify the logic functions without considering the target technology. Redundant logic is removed to reduce the size of the network.

The second step is *technology mapping*. Most commercial FPGAs contain look-up-tables (LUTs) as their basic logic element. Technology mapping is the process that maps a user circuit into LUTs. A K-input LUT is a digital memory that can implement any Boolean function with K inputs; therefore, a K-input LUT can perform any one of the 2^{2^k} different Boolean functions [14]. When targeting FPGA architectures with logic blocks that contains more than one LUT, the netlist of LUTs is packed into clusters of LUTs correspond to the number of LUTs per logic block.

The third step is called *placement*. This step determines the physical locations of the each logic block. A typical *placer* strives to achieve the following goals:

- Minimize the amount of wiring required for the connections between the logic blocks
- Balance the usage of routing wires across the FPGA to avoid congestion
- Optimize the critical path delay by placing the logic blocks of the timing-critical nets close together

A number of well-known algorithms have been introduced to solve the placement problem. *Simulated-annealing* is employed in *Versatile Place and Route* (VPR), the CAD tool that is chosen for this project. The simulated-annealing algorithm originated from the industrial annealing process used to cool molten metal [10]. Initially, logic blocks are placed randomly on a FPGA. The placement is gradually improved by randomly swapping logic blocks. After each swap, a cost function, which depends on the wire length, congestion, and path delay of the nets, is used to evaluate the move. The probability of accepting a bad swap (a swap that causes a increase in the cost function value) is high at the beginning of the process to avoid the placement being trapped in a local minimum of the cost function. The probability then gradually decreases as the process continues until no more good moves can be found.

The final step is *routing*. Routing resources are assigned for the connections between logic blocks. A typical *router* attempts to:

- Route all required nets using the available routing resources and avoid any contention
- Optimize the critical path delay by routing the critical net using short and fast paths

There are two types of routers: global and detail routers. A *global router* considers routing the circuit without worrying about the details of the types of wires and switches used for the net connections. A *detailed router* selects wire segments and routing switches for each net [14]. VPR

contains both types of routers. It employs the *Pathfinder* algorithm to solve the routing problem [10]. The key principal of the Pathfinder algorithm is to allow a physical track to be shared by multiple nets at the beginning of the routing process. After all nets are routed, the nets are then ripped up and re-routed. As the process continues, the cost for sharing a physical track increases to discourage the router from routing multiple nets using the same track. The process ends when no more than one net uses one track; this results in a legal routing. After the routing, the programmable information is then translated to a bit-stream, which corresponds to the configuration bits for the SRAM cells on the FPGA.

2.3 Power Estimation Methods

VLSI low-power design problems can be classified into two types: *analysis* and *optimization* [80]. For analysis, techniques are carried out to estimate power dissipation at different stages of the design process. The accuracy of the power analysis depends on the availability of the physical details of the design. Power analysis performed at higher abstraction levels are less accurate, but requires less computing resources (see Figure 2.7). Power analysis is recommended at each stage of the design process in order to discover power violations as early as possible since the cost for fixing these violations is relatively low in the early design stages [80]. The analysis techniques also serve as the foundation for power optimization methods. Power analysis techniques can be divided into two categories: *simulation* and *probabilistic* [52]. Table 2.3 lists several common simulation and probabilistic power analysis techniques.

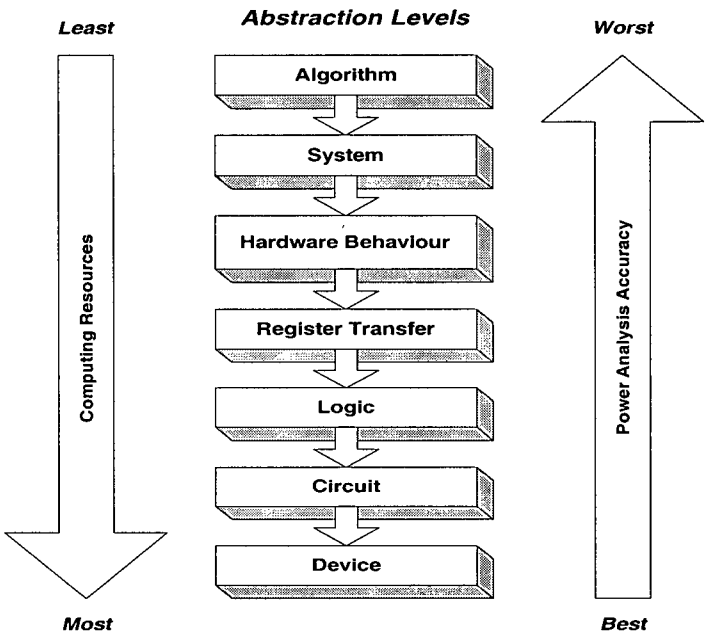


Figure 2.7 Impacts of abstraction levels on computing resources and accuracy for power analysis (from [80])

Table 2.3 A list of several common simulation and probabilistic power analysis techniques

Simulation	Probabilistic
• SPICE • Gate-Level Simulation • Monte-Carlo Simulation • Macro-Modeling	• Signal Probability • Transition Probability • Transition Density

2.3.1 Simulation Power Analysis Techniques

The simulation techniques are often termed *strongly input pattern-dependent* because the simulation techniques require a set of input vectors. Simulation performed at lower level of the circuit design results in better accuracy, but it requires more computing resources. One of the most popular simulation tools is *Simulation Program with IC Emphasis (SPICE).* SPICE simulates circuit power dissipation with a high degree of accuracy. Transient analysis is applied to model the dynamic behavior of the circuit over time and determine dynamic, static, and leakage power dissipation accurately. However, SPICE analysis requires intensive computation resources, and therefore, it is not suitable for large circuits [80].

To speed up the simulation process, gate-level logic simulation is often used. Circuits are modeled as a set of logic gates and nets. Gate-level logic simulation is event-driven; events are defined by the logic switching of nets within a circuit at particular simulation time. This analysis technique can be summarized as follows [80]:

1. Run logic simulation with a set of input vectors

2. Monitor the switching activity of each net and calculate the dynamic power dissipation using $(P=CV^2f)$.

3. Monitor and determine an equation for the dynamic energy dissipation of the internal switching power, which includes the short-circuit power and the dynamic power of the internal nodes embedded in the logic gates.

4. Monitor and define a formula for the leakage power based on the static states of each gate.

5. Determine the total power dissipation by summing the dynamic, internal, and leakage power obtained in steps (2) to (4).

The input vectors have to be chosen carefully as they directly affect the accuracy of the power estimation. If proper input vectors are used to reflect desired operating conditions, the accuracy of the gate-level simulation can be within 10 to 15% of SPICE simulation results [12].

Generating a desirable set of input vectors for the circuit simulation is difficult, as the physical details of the circuit may not be completely specified during the design phase [51]. To solve this problem, a *Monte Carlo simulation* method has been suggested. Even though the designers

may not be able to generate a set of input vectors which reflects the intended application of the circuit, the circuits can be simulated with thousands of randomly generated input vectors to achieve an relatively accurate power estimate. The algorithm of Monte Carlo power simulation is shown as a flowchart in Figure 2.8.

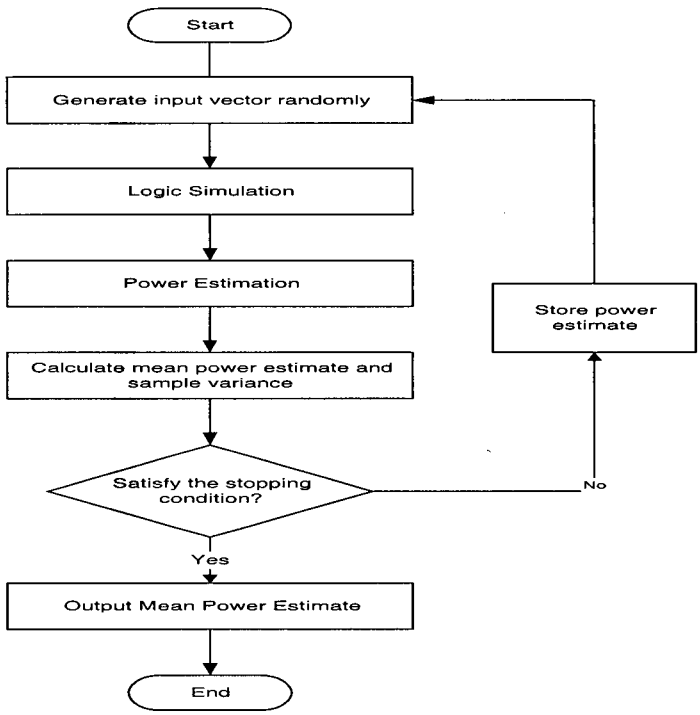


Figure 2.8 Monte Carlo power simulation

Sets of input vectors are generated randomly. Logic simulation and power estimation is performed for each set of input vectors. The *Mean power estimate*, P_{mean}, and the *sample variance*, S^2, are calculated using the following equations [15][49][80]:

$$P_{mean} = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{2.1}$$

$$S^2 = \frac{1}{N}\sum_{i=1}^{N} (p_i - P_{mean})^2 \tag{2.2}$$

where N is the total number of power estimation samples, and p_i represents a single power estimate obtained from a set of input vector. The *stopping condition* is defined as:

$$N \geq \left(\frac{t_{[N-1,\alpha/2]} \cdot S}{\varepsilon \cdot P_{mean}} \right)^2 \tag{2.3}$$

where $t_{[N-1,\alpha/2]}$ is a value from the t-distribution as the distribution of the sample values is modeled using a normal distribution function; N is the number of samples and the confidence level, $(1-\alpha)$; and ε represents the error tolerance. At the end of each iteration, if the *stopping condition* is satisfied, power simulation is complete. This method provides an accurate estimate of the total power dissipation of a large circuit, but the power consumed by individual gates or small group of gates cannot be estimated accurately [51]. The power consumption of individual gates is essential to diagnose the sources of high power consumption violations in the circuit. Therefore, a modification of the Monte Carlo simulation has been proposed to provide individual-gate power estimation by applying *Central Limit Theorem* [51][52][80]. However, this approach is slow. A study shows that running Monte Carlo simulation with an error tolerance of 3% on a circuit with 54 logic blocks could take 60 minutes using a 1 GHz processor [62]. Also, the importance of properly defined input pattern characteristics from the users is still critical to obtain an accurate power estimate using this method [62].

The simulation approaches discussed above are referred as *top-down* methods, which is useful for new designs when the applications and the internal details within the circuit are not precisely defined. Another simulation approach, called *macro-modeling*, is classified as a *bottom-up* method. Such a technique is effective for circuits which have been designed and reused [51][52]. This technique contains a training phase and an evaluation phase [51][52][60]. Usually, a particular *macro-model* with coefficients is setup at the beginning of the training phase. When

the training starts, power estimation is carried out for a specific set of benchmark circuits using both the macro-model and another more accurate simulation technique, such as Monte-Carlo simulation, which acts as a reference. The CAD tool then adjusts the coefficients in the macro-model to match the power estimates from the simulation technique using the *Recursive Least Square (RLS)* algorithm [25][27][51]. During the evaluation phase, the macro-model is applied to calculate the power dissipation for new circuits. Note that the macro-models are highly dependent on the training vectors. If the models are used to estimate power for circuits which are very different from those in the training set, the power estimate would be inaccurate [51].

2.3.2 Probabilistic Power Analysis Techniques

Compared with the simulation approaches, probabilistic power analysis techniques are more computationally efficient. Each logic signal is characterized based on numerical statistical characteristics, such as the probability for the signal to maintain a high logic value, and the switching rate of the signal. This eliminates the need for a large set of specific input vectors. Therefore, the probabilistic techniques are classified as *weakly input pattern-dependent.* Instead of simulating a large number of input vectors and then calculating the average result, only one set of computations is required. Most of the probabilistic approach uses simplified delay models for the circuit components. Supply and ground voltages are fixed, and only the signals toggling are considered. Generally, the techniques assume the circuit inputs are independent; this assumption is referred as *spatial independence.*

Among all the probabilistic techniques, *signal probability* is the simplest technique. The signal probability at a node, x, is defined as the average fraction of clock cycles in which the steady

state value of the node is logic high [52]. The probability at the output of each gate depends on the logic function of the gate and the probability at each input.

Another common probabilistic method is *transition probability*. It is defined as the probability of a signal switching in one clock cycle. This technique generally assumes that the values of the same signal in two consecutive clock cycles are independent to each other, also referred as *temporal independence* [52]. The formula for transition probability considers the possibilities for transitions from 0 to 1 and from 1 to 0, where $P(y)$ is the signal probability at the output of the gate:

$$P_t(y) = (1 - P(y)) \cdot P(y) + P(y) \cdot (1 - P(y)) = 2 \cdot P(y) \cdot (1 - P(y)) \qquad (2.4)$$

The power dissipated by a circuit with n nodes using transition probability is:

$$\text{Power} = \frac{1}{2} \cdot f_{clk} \cdot V^2 \cdot \sum_{i=1}^{n} C_i \cdot P_t(x_i) \qquad (2.5)$$

where f_{clk} is the clock frequency, V represents the swing voltage, and C_i is the node capacitance. Because the transition probability only allows at most one transition per clock cycle, this technique only estimates the lower bound of the true average power [51].

Another more effective measure of switching activity in logic circuits is called *transition density* [80]. The transition density is used to determine the switching activity of each signal within the circuit. The model has two parameters:

1. Transition density: the average number of transitions per unit time, where $n_x(T)$ represents the number of transitions within time T.

$$D(x) = \lim_{T \to \infty} \frac{n_x(T)}{T} \qquad (2.6)$$

2. Static probability: the probability of the signal being high for a certain time period:

$$P(x) = \lim_{T\to\infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} x(t)dt \tag{2.7}$$

In a synchronous circuit, the relationship between the transition density and the transition probability of a signal is:

$$D(x) \geq \frac{P_t(x)}{T_c} \tag{2.8}$$

where T_c is the clock cycle and $P_t(x)$ is the transition probability described above. The transition probability of the signal within the clock cycle will always be less than the transition density of the signal [52].

Figure 2.9 and Table 2.4 illustrates the transition density and static probability of four example signals. Assume that we monitor all the signals for four clock cycles. The transition density and static probability of each signal are defined with respect to the clock based on the statistical values within the four clock cycles in this example. A typical clock signal has a half-clock duty cycle and toggles twice every clock period. As a result, it has a static probability of 0.5 and a transition density of 2. In Figure 2.8, *signal A* represents a normal signal driven by the rising edge of the clock and has a high logic value half of the time. Signals with glitches switch faster than a half of the clock frequency (i.e. signals with transition density higher than 1). S*ignal B* is an example of a signal that has glitches. *Signal* C, on the other hand, represents the normal assumption for primary inputs, with a transition density of 0.5 and a static probability of 0.5.

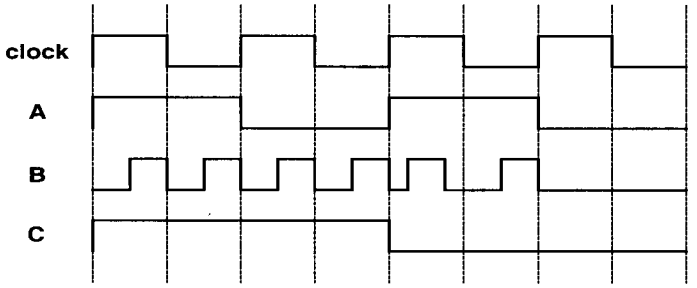


Figure 2.9 Examples of signals represented transition densities and static probabilities

Table 2.4 Examples of signals represented transition densities and static probabilities

Signal	Transition Density	Static Probability
Clock	2	0.5
A	1	0.5
B	3.0	0.375
C	0.5	0.5

The propagation of transition density through each logic gate can be calculated as follows [80].

1. Express the logic function of the gate *f(x)*
2. Determine the *Boolean difference* at the output with respect to each, x_i:

$$\frac{\partial f(x)}{\partial x_i} = f(x)\big|_{xi=1} \oplus f(x)\big|_{xi=0} \tag{2.9}$$

3. Calculate the probability of the *Boolean difference*, $P(\partial f(x)/\partial x_i)$, which becomes the static probability of x_i causing a change in the output.
4. Apply the following equation for the transition density at the output of the gate:

$$D(y) = \sum_{\text{all inputs}} P\left(\frac{\partial f(x)}{\partial x_i}\right) \cdot D(x_i) \tag{2.10}$$

As all inputs are assumed to be uncorrelated to each other, each input contributes a static probability, $P(\partial f(x)/\partial x_i)$, and a transition density, $D(x_i)$, to the total density, $D(y)$, at the output [52]. Usually, the primary inputs are represented as normalized random signals, which have a static probability of 0.5 and a transition density of 0.5. The power estimation for a circuit with n number of nodes using transition density can be calculated as:

$$\text{Dynamic Power} = \sum_{\text{all nodes}} 0.5 \cdot C_y \cdot V^2 \cdot D(y) \cdot f_{clk} \qquad (2.11)$$

where f_{clk} is the clock frequency; V represents the swing voltage, and C_i is the node capacitance. The transition density method is more accurate than the signal probability and transition probability methods because of the additional parameter, transition density, to characterize the randomness of the signal. However, the spatial independence assumption introduces uncertainties to the power analysis. The absolute error of the power analysis using transition density is within 23% [53]. Indeed, a careful consideration on the correlation of the input signals can improve the accuracy of the power estimates. Unfortunately, the problem for determining the correlation among the inputs is *NP-Hard* [51]. There have been several extensions of the transition density model, which handle correlation [49][52][51][80]. However, these extensions complicate the algorithm and degrade the computation efficiency, which is the main advantage of probabilistic analysis.

2.4 Available FPGA Power Estimation Tools

Power consumption issue captures FPGA vendors' attention because of the growing complexity of the programmable devices and the emerging handheld market, which are sensitive to the power usage [69]. Designers have to be aware of the power consumption of circuits during the design stage. Most FPGA companies provide Excel or online spreadsheets for their customers to estimate power dissipation for particular devices [1][7][8][40][41][73][76].

Some vendors, such as Xilinx and Altera, have incorporated the power estimation feature in their CAD tools [9][75].

2.4.1 Power Estimation Spreadsheets

In general, these spreadsheets analyze power in three portions: static power, dynamic power and the maximum allowed power. The static power of each component in the FPGA is provided in the device family data sheet. Designers can determine the total static power based on the number of specific components used. For the dynamic power calculation, designers need to provide the average switching frequency for all the logic blocks, or for each module in their design. Coefficients for adjusting the dynamic calculation are provided in the device data sheet. The total estimated power is the summation of the static and dynamic power. For evaluation, the maximum allowed power is determined. If the total estimated power is less than the maximum allowed power, then the power estimation of the circuit design is assumed to be valid. The flow of the power calculation is expressed below [7][76]:

1. Identify the components in the design and see which ones dissipate static power. Use the following equations to calculate static power. Note that the $P_{static\_per\_component}$ is from the device data sheet

$$P_{static} = \sum_{\text{all components}} P_{static\_per\_component} \tag{2.12}$$

2. Determine the average switching activity of the logic blocks, or of each module in the device using the spreadsheet. Apply the formula below for dynamic power. The coefficient value K is provided in the data sheet, $F_i(max)$ and S_i is the maximum frequency and average switching activity for device respectively; C

represents the capacitance related to that particular module, V_{supply} is the supply voltage, and V_{swing} is the swing voltage.

$$P_{dynamic} = \sum_{\text{all modules}} K \cdot f_i(max) \cdot S_i \cdot C_i \cdot V_{swing} \cdot V_{supply} \tag{2.13}$$

3. The total estimated power is:

$$P_{total} = P_{static} + P_{dynamic} \tag{2.14}$$

4. The maximum allowed power (P_{MAX}) for a device is calculated:

$$P_{MAX} = \frac{T_j - T_A}{\theta_{jA}} \tag{2.15}$$

T_j and θ_{jA} represent the maximum junction temperature and thermal resistance of the device. The ambient temperature is determined by the user.

5. Check if P_{total} is less than P_{MAX}

This method can only give a rough approximation of power. It also requires designers to thoroughly understand the switching activity inside their circuit, which is nearly impossible at the design stage.

2.4.2 Power Estimation CAD Tools

To automate the power estimation process, Xilinx and Altera include additional features in their CAD tools, *Xpower* and *Quartus* [9][69]. Both companies appear to take the same approach to estimate switching activities. Figure 2.9 shows the flowchart of the power estimation flow. First, users specify the circuit in a high-level description language. After the circuit is synthesized to a *netlist* during *High-level Synthesis*, users then provide input stimuli as waveforms, absolute switching frequencies, or toggle rates relative to the clock frequency [9][69]. The *activity generation software* estimates the switching activities inside the circuit by simulation [47]. The simulation time must be long enough to ensure valid activity rate

information. Xilinx recommends that users edit the switching activity of the nodes in the activity output file after simulation to achieve a better accuracy on power estimation [75]. Power is estimated based on the routing information from *Logic Synthesis* and the switching activity information.

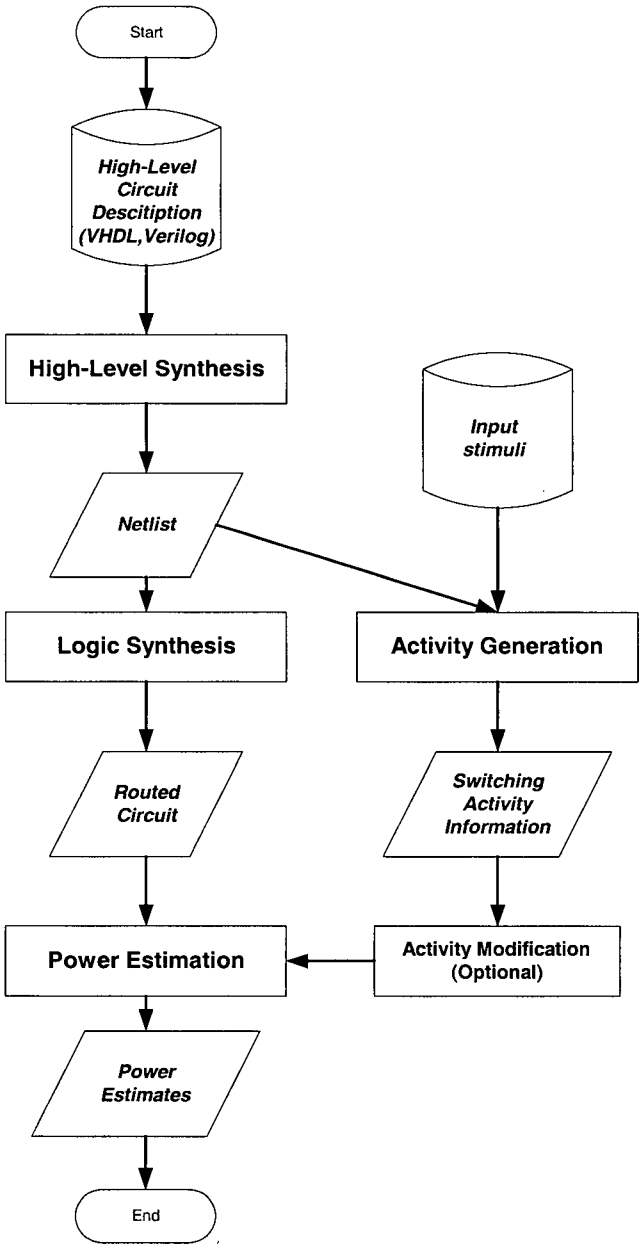


Figure 2.10 Power estimation flow

2.5 Focus and Contributions of this Thesis

The goal for this research project to build a power analysis tool for future FPGA architectural and CAD tool research. Our approach is similar to the power estimation flow discussed in Section 2.4.2, except that we apply a probabilistic method for faster activity generation and a parameterized architectural model to define a wide variety of FPGA architectures. The power model has been integrated into the widely used *Versatile Place and Route (VPR)* CAD Tool. In Chapter 3, we review the parameterized architectural model of VPR and propose our enhancements to the original model. In Chapter 4, we discuss the details of our power model. In Chapter 5, we investigate the impact of architectural changes on power dissipation and investigate the sensitivity of our results.

The contributions of this thesis are summarized as follows:

1. Enhancements to the existing parameterized architectural model of FPGA
2. A novel power model for activity generation and power estimation
3. Experimental evaluations of the effect of various architecture changes on the power dissipation of an FPGA

Chapter 3

3 OVERVIEW OF VERSATILE PLACE AND ROUTE (VPR) CAD TOOL

This chapter begins with a description of the original and the modified Versatile Place and Route (VPR) CAD tool framework. It also provides a description of the parameterized FPGA architectures in VPR, and explains different routing, logic block, and clock network architectures. The architectural assumptions described here are critical to the power model described in Chapter 4.

3.1 VPR Framework

3.1.1 Original Framework

The *Versatile Place and Route (VPR)* CAD tool is a widely used placement and routing tool available for FPGA architectural studies [10]. It is used in combination with VPACK or TVACK, a logic block packing tool which packs each logic block to capacity and minimizes the number of inter-cluster connections on the critical path [11]. VPR has two components: a place and route tool, and a detailed area and delay model (See Figure 3.1). The place and route tool maps a circuit to the FPGA. The area and delay models estimate the area and critical path delay based on results from the place and route tool. The two components interact with each other to determine the best placement and routing for a user circuit. A description of the underlying FPGA architecture is provided to the tool in the form of an *architecture file*, which contains information such as segment length, connection topologies, logic block size and

composition, and process parameters. The architecture file is an important feature in VPR--it allows any architecture to be specified, and hence makes the CAD tool highly flexible.

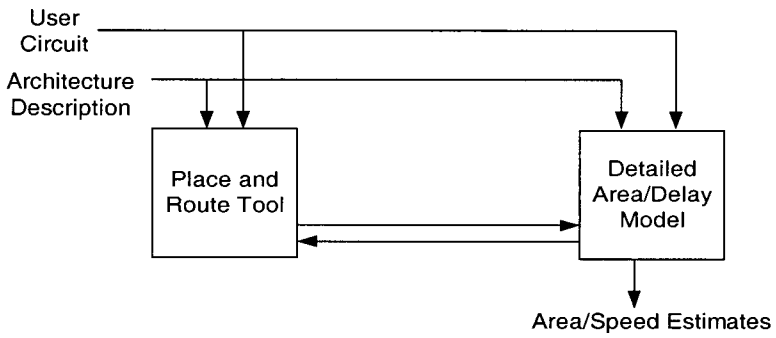


Figure 3.1 VPR framework

3.1.2 Modified Framework

Figure 3.2 shows the VPR framework with the new power model. In this framework, the power model is part of the area and delay model. An activity estimator has been developed to estimate the switching frequencies of all nodes in the circuit. Details of both the activity estimator and the power model can be found in Chapter 4. In the current implementation, the activity estimator and the power model are not used to guide the placement and routing. It estimates the power consumption only after placement and routing has occurred. In future implementations, however, it is possible to use the power estimates to guide the placement and routing process in order to optimize for power.

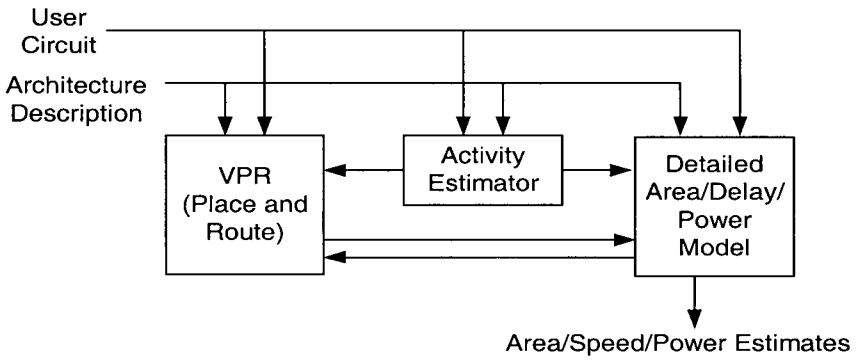


Figure 3.2 Framework with power model

3.2 FPGA Architectural Assumptions

VPR targets SRAM-based FPGAs. SRAM cells are used to control the pass-transistor switches and tri-state buffers along the routing tracks. They are also used to store the logic functions implemented by each look-up-table, and act as select inputs for multiplexers within the logic blocks. VPR assumes that each SRAM cell has six minimum-size transistors (see Figure 3.3). The *data* and *data_bar* values stored in the cell can be connected directly to other wires without any isolating inverter. Since an NMOS pass transistor conducts a "*logic 1*" value poorly due to the body effect, the output voltage is reduced by the threshold voltage. To solve this problem, the gate voltage of these pass transistors is often boosted to overcome the body effect [10]. VPR is intended for *island-style* FPGAs. The architecture description of an FPGA is parameterized for VPR. Additional parameters are added to the original parameterized architecture for the power calculations. The logic block, routing, and clock network architectures are described in the following sections.

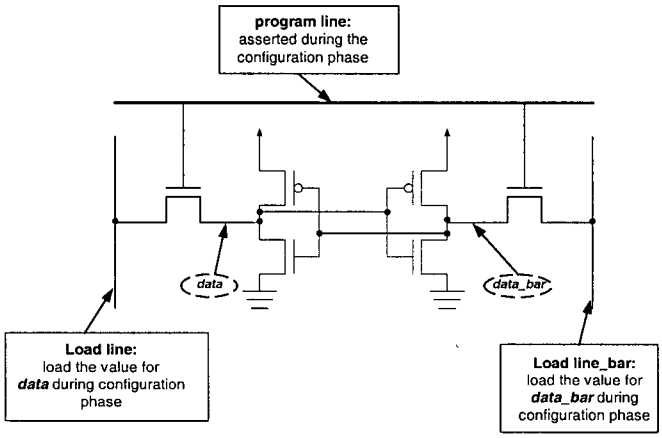


Figure 3.3 Schematic of a six-transistor SRAM cell

3.2.1 Logic Block Architecture

VPR assumes a cluster-based logic block. Figure 3.4 shows the schematic of a cluster-based logic block with two LUTs and four inputs. Note that the number of LUTs in each logic block and the number of inputs into a logic block cluster can be specified by the user in the architecture file. Table 3.1 lists the parameters used to describe the logic block architecture [11].

Table 3.1 Logic block architectural parameters

Parameters	**Description**
Cluster_size	Number of LUTs per logic block
LUT_size	Number of inputs per LUT
CLB_Cwire	Capacitance of the internal wire per unit length
Class	Nature of the input and output pins
Location	Locations around the logic block where the input and output pins can be programmably connected to the routing fabric

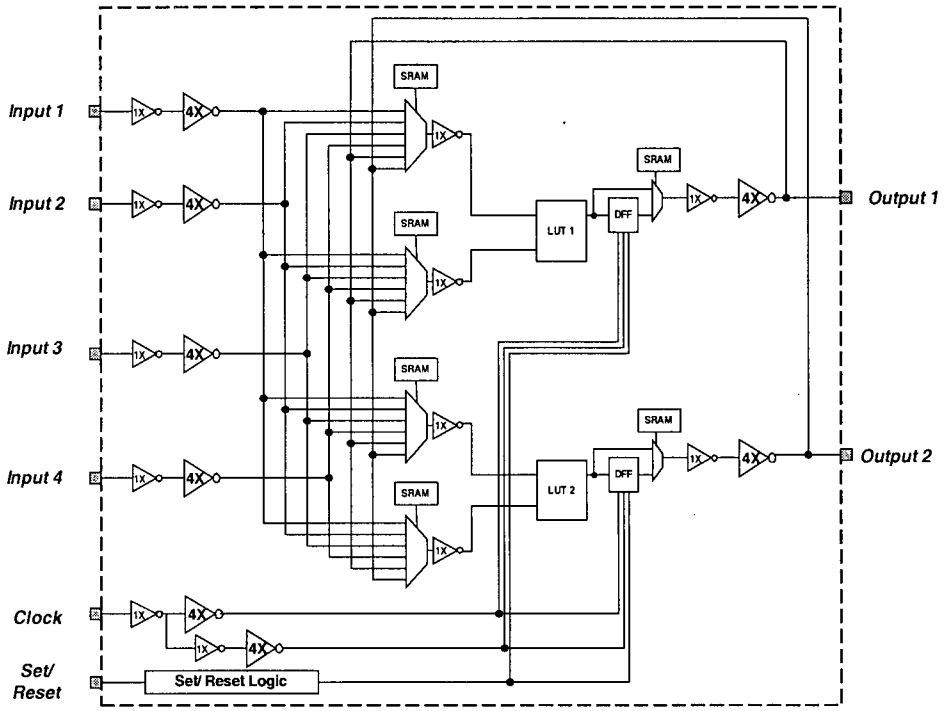


Figure 3.4 Schematic of a clustered logic block with 2 LUTs and 4 inputs

Figure 3.4 shows an example logic block that contains two LUTs; each LUT has two inputs. Therefore, the values for both the *cluster_size* and *LUT_size* are 2. Moreover, since each logic block is surrounded by routing fabric in the *Island-Style* architecture (see Figure 2.4), it is essential for the router to understand where a particular input/output can be programmed to connect to the routing tracks.

Figure 3.5 shows another example, in which *LUT 1* has been programmed to implement an AND function while *LUT 2* has been programmed to implement a NOR. The *in 1* and *in 2* are inputs of *LUT 1* while *in 3* and *in 4* are of LUT2. Clearly, the inputs of each LUT can be interchanged; these pins are said to be *logically equivalent.* For VPR, the logically equivalent inputs/outputs are grouped into the same class in the *architectural file.* Since *out 5* and *out 6* are the outputs on two different functions, they are of different classes. *In 7* is a global input

(clock, set, or reset) for the D-flip-flops inside the logic block; it belongs to another class. The connection among the global input and D-flip-flops is not shown in the figure. The fourth column of Table 3.3 explains the location of each input and output, where the connection to the routing fabric can be made. For instance, *in 1* can be reached from both the top and the bottom of the logic block [10].

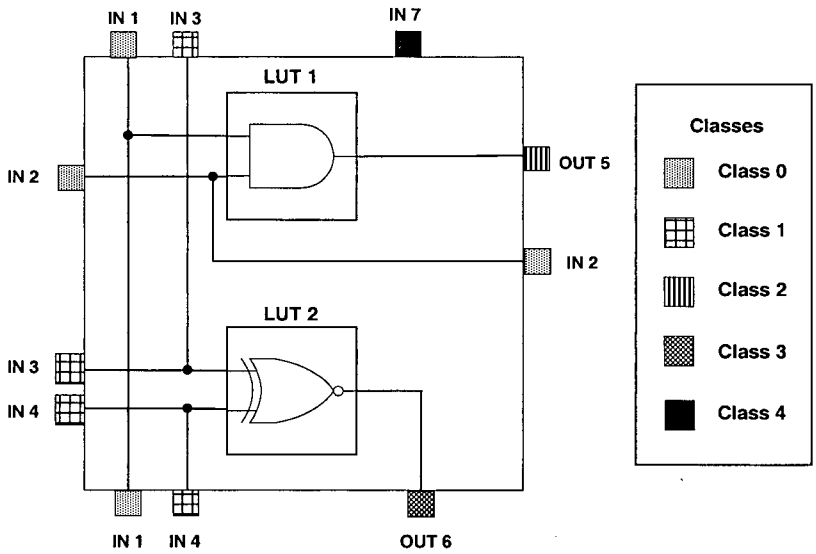


Figure 3.5 An example on logic block description

Table 3.2 Logic block parameters for the example in Figure 3.5

I/O Number	I/O Type	Class	Location
1	Input	0	Top, Bottom
2	Input	0	Left, Right
3	Input	1	Top, Bottom
4	Input	1	Bottom, Left
5	Output	2	Right
6	Output	3	Bottom
7	Input	4	Top

CLB_Cwire is an additional parameter by which the user can specify the wire capacitance for the internal routing within the logic block based on the assumed metal and process technology. Note that the capacitance of the transistors embedded inside the logic block is not defined in

the *architecture file*. Minimum size transistors are assumed to be used in the LUTs and multiplexers within the logic blocks.

3.2.2 Routing Architecture

The parameters for describing the routing architecture can be divided into three groups: channel, switch block, and wire parameters. From these parameters, VPR constructs a *routing resource graph* containing a representation of the entire routing fabric. Each group of parameters and the routing resource graph will be explained below.

3.2.2.1 Channel

The channel descriptions specify the channel width, the connections between the I/O pads and the routing tracks, and the connections between the logic blocks and the routing tracks. Parameters for the channel architecture are listed below [11].

Table 3.3 Channel parameters

Parameters	**Description**
io_rat	The number of I/O pads that fit into one row or column of logic blocks
chan_width_x	The relative channel width of the horizontal channel
chan_width_y	The relative channel width of the vertical channel
chan_width_io	The relative channel width of the channel between the logic blocks and the I/O pads
$F_{c,input}$	The number of tracks connected to each logic block input pin
$F_{c,output}$	The number of tracks connected to each logic block output pin
$F_{c,pad}$	The number of tracks connected to each I/O pad

Figure 3.6 shows an example FPGA architecture. This example has a 3 by 3 logic block array. Since two I/O pads can fit into each row or column of logic blocks, *io_rat* is 2. Both of the channel widths in vertical and horizontal direction are the same. Therefore, the relative value

of both channel widths, *chan_width_x* and *chan_width_y*, are 1. The channel width between the I/O pad and the logic block is half the width of the vertical and horizontal channels. Thus, the relative value of the I/O channel, *chan_width_io,* is 0.5.

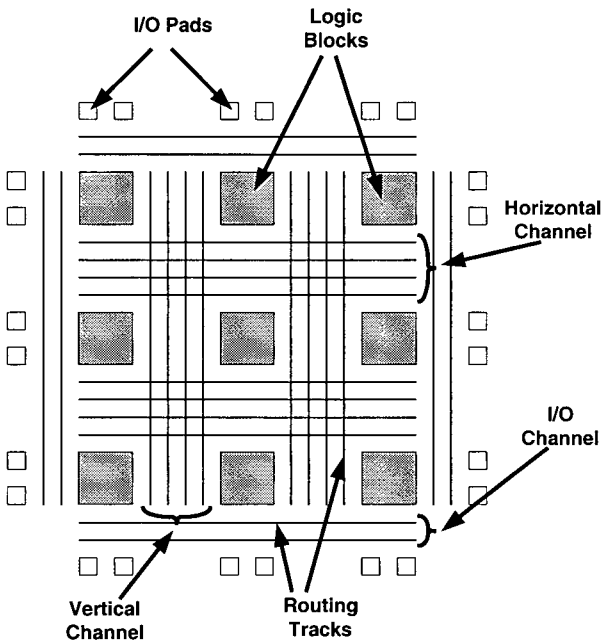


Figure 3.6 High-level FPGA model assumed by VPR

Figure 3.7 shows the different kinds of connections within the channels of the example. In this case, the connection flexibility for the I/O pad, $F_{c,pad}$, is 1 because all the available tracks in each I/O channel are connected to each I/O pin. Since the input pin of each logic block is connected to only two of the four tracks in the channel, the $F_{c,input}$ value is 0.5. The $F_{c,output}$ value is 0.25 as the output of the logic block is connected to only one of the four tracks in the channel.

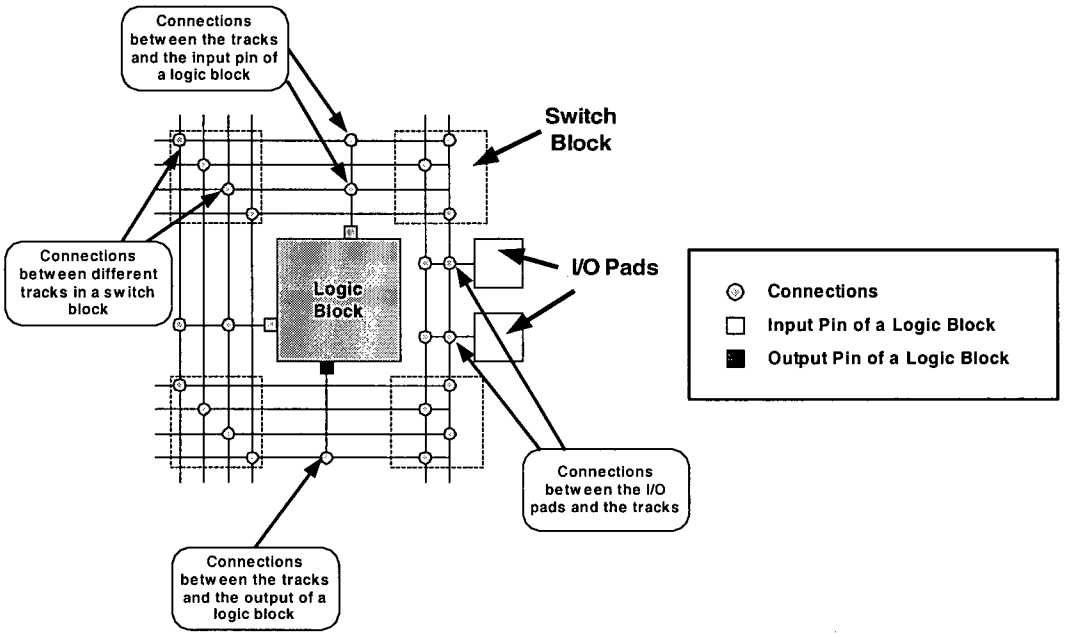


Figure 3.7 Connections between the logic blocks, routing tracks, and I/O pads

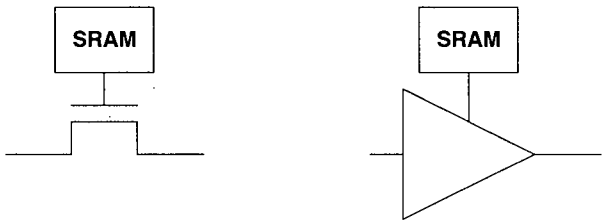
3.2.2.2 Switch Block

A switch block is used to program the connections between the horizontal and vertical routing tracks. VPR describes the architecture of the switch block in two levels: the characteristics of the switches in the switch block, and the topology of the switch block itself. The following are the parameters associated with the architectural description of a switch block [10].

Table 3.4 Switch block parameters

Parameter	**Description**
buffered	Switch type (*No* for Pass transistor, and *Yes* for tri-state buffer)
R	The resistance of the switch
Cin	The input capacitance of the switch
Cout	The output capacitance of the switch
Tdel	Intrinsic delay through the switch
Fs	Switch block connection flexibility
switch_block_type	Switch block topology

Each switch inside a switch block can be either a pass transistor or a tri-state buffer, and is controlled by a SRAM cell (See Figure 3.8). For timing analysis, the resistance and intrinsic delay of the switch specified in the architecture file are used. For power estimation, the input and output capacitance of the switch are employed. Figure 3.9 shows the schematic of a 5X tri-state buffer. The buffer chain within a tri-state buffer is assumed to be built with a stage ratio of approximately 4.



a) Pass Transistor b) Tri-state buffer

Figure 3.8 SRAM-controlled pass transistor and tri-state buffer

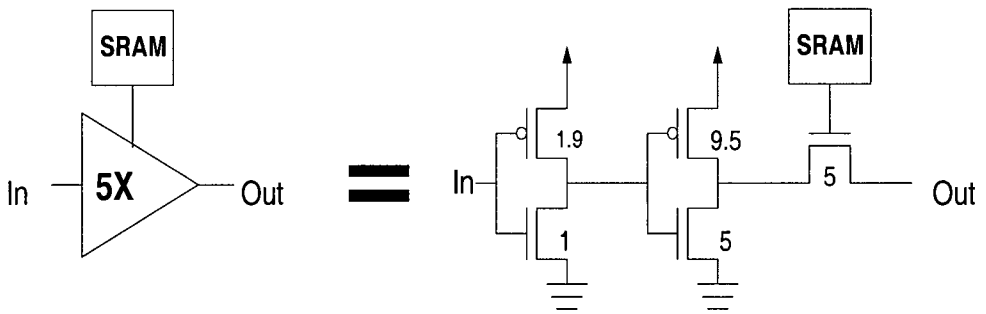


Figure 3.9 Schematic of a 5X tri-state buffer

Consider the switch block shown in Figure 3.10. Each pin on the switch block is designated by a side and a pin number. For instance, *top0* refers to the pin at the top left corner. The connection flexibility of the *top0* pin is defined by the number of connections available from this pin to pins on the three other sides. Since the *top0* pin is connected to six other pins, its F_s value is 6. Previous studies have shown that the optimal value of F_s is 3 [10].

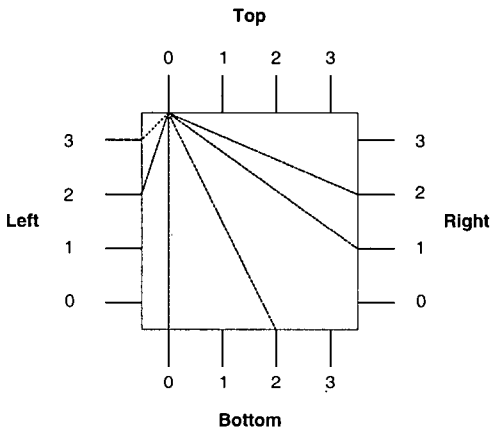


Figure 3.10 Switch block connection flexibility

VPR supports four different switch block topologies: Disjoint [42], Universal [17], Wilton [70], and Imran [46]. The Disjoint switch block, also known as subset switch block, connects each pin to pins with the same pin number on the three other sides of the switch block. The Disjoint switch block topology shows a symmetric pattern along the diagonal of the switch block (see Figure 3.11(a)). The Universal switch block focuses on maximizing the number of simultaneous connections among the routing tracks. It forms regular diamond patterns along the diagonals (see Figure 3.11(b)). The Wilton switch block is similar to the Disjoint, except that the diagonal connections from the pins are rotated by one track (see Figure 3.10(c)). A previous study shows that Wilton switch block provides higher routing flexibility than the Disjoint block [70].

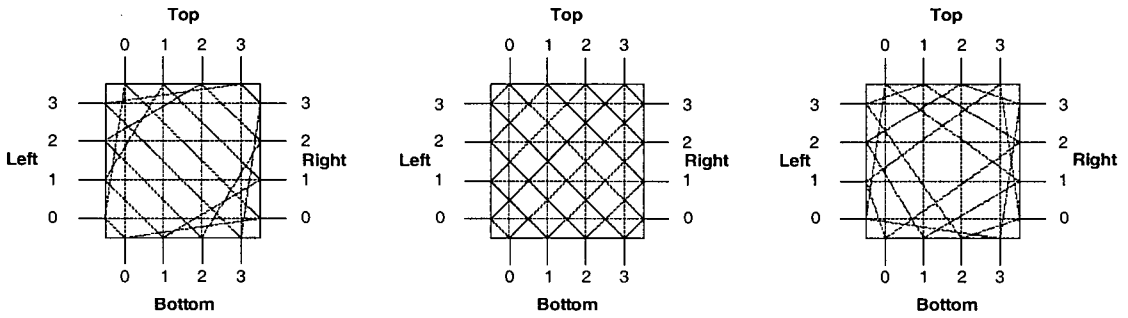
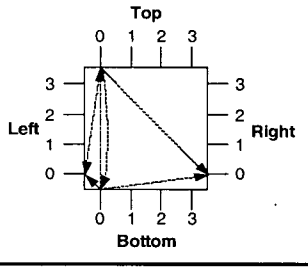
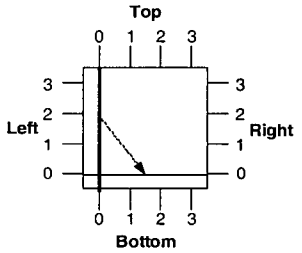
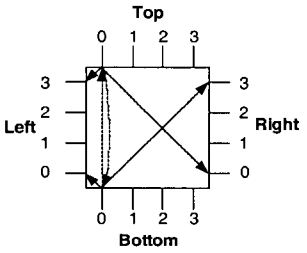
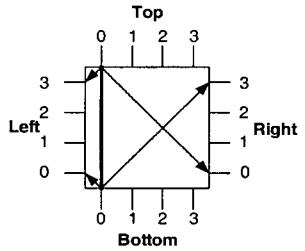
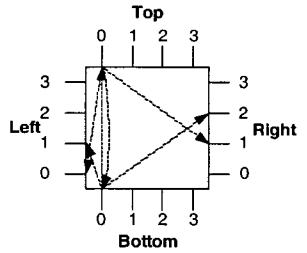
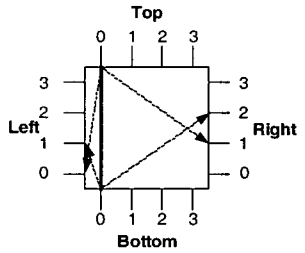
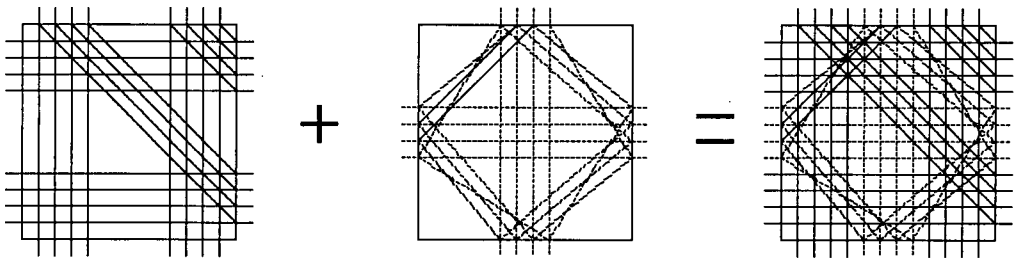


Figure 3.11 Three switch block topologies supported by VPR (from [46])

Bypass wires eliminate some of the switches within switch blocks. Table 3.5 demonstrates how these eliminations occur in the above three different types of switch block topologies. Each switch is represented by an arrow. The direction of the arrow corresponds to the direction of the signal transmitted via the switch. Note that in the Disjoint topology, five of the six switches for each pin can be eliminated, but only two can be removed in the other two topologies. As a result, the Disjoint topology is more area-efficient for longer wires compared with the others. To optimize the routability and area-efficiency, the Imran switch block applies the Disjoint topology for the tracks that bypass the switch block and the Wilton topology for the tracks that terminate in the switch block (see Figure 3.12) [46].

Table 3.5 Connection differences within a switch block when a bypass wire exists

Switch Block Topology	When a wire terminates in the switch block	When a wire bypasses the switch block
Disjoint		
Universal		
Wilton		



Disjoint topology **for tracks that bypass the switch block**

Wilton topology **for tracks that terminate at the switch block**

combined connections

Figure 3.12 Imran switch block [46]

3.2.2.3 Wire Segments

Each type of wire segment in the architecture is described using the parameters listed in Table 3.6 [11]. For example, a wire segment type with a segment frequency of 0.5 means that it makes up fifty percent of the wire segments in the FPGA. The length of the segment, *segment_length*, is determined by the number of logic blocks spanned by the wire segment. Users can specify the resistance and capacitance of the wire based on the process technology.

Table 3.6 Wire Segment Parameters

Parameters	**Description**
segment_frequency	How often is this segment being used in the FPGA
segment_length	Number of logic blocks that are spanned by this wire
Rmetal	Resistance per unit length
Cmetal	Capacitance per unit length
opin_switch	The switch type used to link the wire and the logic blocks together
wire_switch	The switch type used to connect one wire segment to another wire segments
Frac_cb	The internal population of connection switches for the segment
Frac_sb	The internal population of wire switches for the segment

The internal populations of the wire and connection switches are calculated as follows:

$$\text{Frac\_sb} = \frac{\text{Number\_of\_Wire\_Switches}}{(\text{Segment\_Length} + 1)} \tag{3.1}$$

$$\text{Frac\_cb} = \frac{\text{Number\_of\_Connection\_Switches}}{\text{Segment\_Length}} \tag{3.2}$$

Consider the examples in Figure 3.13. Assuming that three types of switches have already been defined, major parameters such as, *wire_switch*, *opin_switch*, *segment_length*, *Frac_cb*, and *Frac_sb,* can be determined for each segment (see Table 3.7). For instance, segment A spans one logic block. Therefore, its segment length is 1. The wire switch for segment A is type 1, and the internal population of the wire switches is also 1, according to Equation (3.1). The connection

switch for A is type 3 and the internal population of the connection switches is 1, according to Equation (3.2).

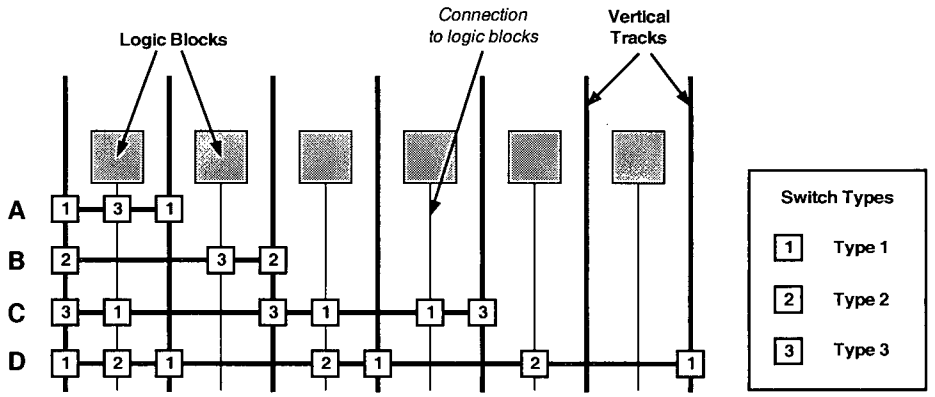


Figure 3.13 Wire segment examples

Table 3.7 Parameters for the wire segment examples

Segment	*segment_length*	*wire_switch*	*opin_switch*	*Frac_sb*	*Frac_cb*
A	1	1	3	1	1
B	2	2	3	0.66	0.5
C	4	3	1	0.6	0.75
D	6	1	2	0.57	0.5

3.2.2.4 Routing Resource Graph

As discussed in the previous sections, the routing architecture is parameterized into a format that can easily be imported to VPR. Based on the information in the *architecture file*, VPR generates a routing-resource graph for the connectivity information in the FPGA. In the routing-resource graph, the node and edges can be interpreted as follows [10]:

- a node represents a wire or logic block pin
- a directed edge represents a unidirectional switch, such as a tri-state buffer
- a pair of directed edges represents a bi-directional switch, such as a pass transistor

Figure 3.14 is a routing-resource graph representing connections between two logic blocks, each of which has 2 input pins and 1 output pin. The graph starts at a source, which points to the output of *logic block B*, *B_out*. It then splits into two branches as below:

- One branch goes through *wire 1*, a pass transistor in a switch block, *wire 3*, and then to the input pin of *logic block A*, *A_in2*.
- The other branch passes through *wire 2*, a tri-state buffer, *wire 4* and to the input pin, *A_in1*.

Both branches end at *logic block A*; therefore, they have a common sink in the graph.

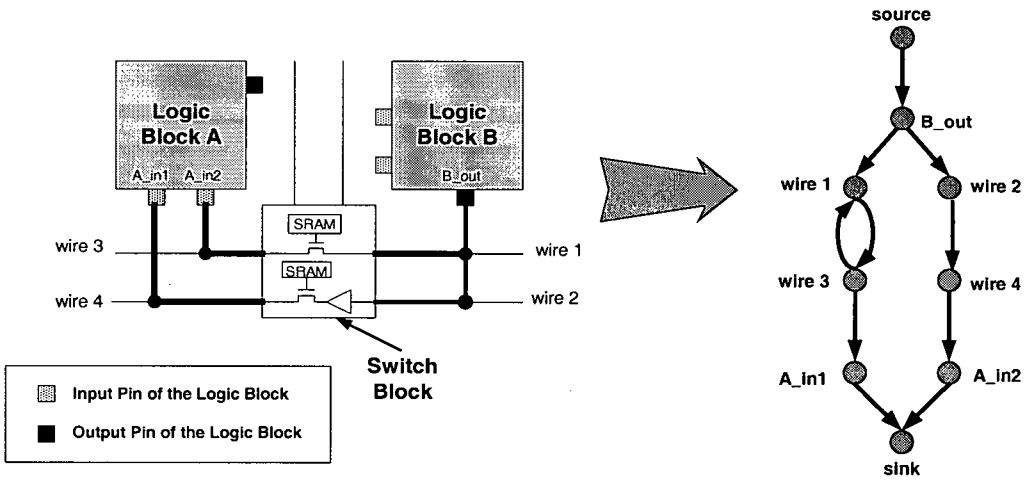


Figure 3.14 Routing resource graph for connections between two logic blocks

Within VPR, each node and edge is associated with a local record. For a node, the record contains information including its occupancy, capacity, edges linked to the node, and capacitance of that particular node. For instance, if a logic block has two logically equivalent input pins and only one of them is being used, the two input pins can be represented by one node with a value of 1 for its occupancy and 2 for its capacity. The capacitance of a node what the node represents. If the node is a wire, its capacitance includes the wire capacitance and the switches' capacitance. If the node is an input or output pin, the capacitance includes the

connection switches and the capacitance of the pin. The capacitance at each node is important for the power estimation.

For each edge on the graph, the record stores information about each switch in the routing architecture, such as switch type used for that particular connection [11]. This information is essential for calculating the number of unused switches, which is required for the estimation of leakage power within the FPGA. By looping through the nodes and edges in the routing-resource graph, the utilization of each type of switch and wire segment can be determined. The total number of unused switches for each switch type can be calculated by pseudo-code shown in Figure 3.15.

```
Let U = total number of wire segments;
Let available_switch(i) be the number of available switches for switch type, i;
Let used_switch(i)be the number of used switches  for switch type, i;
Let unused_switch(i) be the number of unused switches  for switch type, i;

available_switch(i) = 0;
used_switch(i) = Get_utilization_information();

for (each type of wire segment, j) {
        seg_len  = length_of_segment(j)
        seg_freq = segment_frequncy(j);
        W = wire_switch_type(j) ;
        C = connection_switch_type(j) ;

        available_switch (C) += number_of_connection_switches(seg_len, seg_freq, Frac_cb);
        available_switch (W) += number_of_wire_switches(seg_len, seg_freq, Frac_sb);
}

unused_switch(i) = available_switch(i) - used_switch(i);
```

Figure 3.15 Pseudo-code for calculating the number of unused switches

3.2.3 Clock Distribution Network

VPR originally assumes that all global signals (including the clock) are connected to a special, dedicated resource; it does not attempt to route any global signals. As a result, it is necessary to assume a clock distribution network in the FPGA model for realistic power estimation. An H-

tree clock distribution network is assumed in our model, as shown in Figure 3.16. Additional parameters are required for the clock network architecture description:

Table 3.8 Clock network parameters

Parameter	Description
global_clock_num	Number of global clock networks
buffer_R	Resistance of clock buffers
buffer_Cin	Input capacitance of clock buffers
buffer_Cout	Output capacitance of clock buffers
Rwire	Wire capacitance per unit length
Cwire	Wire resistance per unit length
Cin_per_clb_clock_pin	Input capacitance for the clock input pin of each logic block

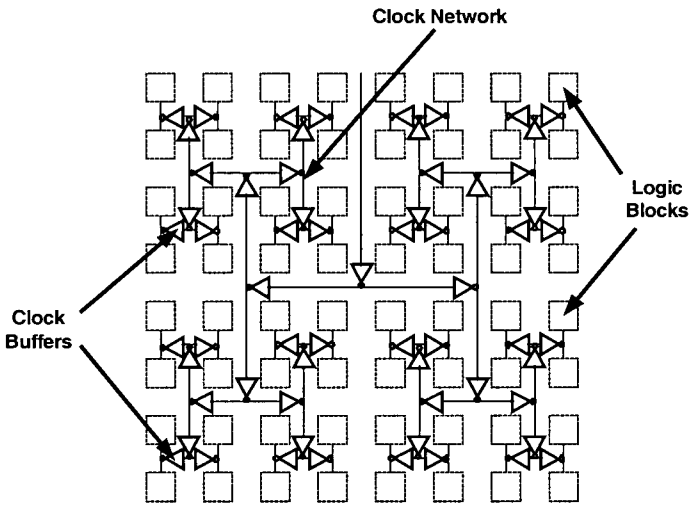


Figure 3.16 An H-Tree clock distribution network

The wire segments between the clock buffers can be very long in large FPGAs. As delay is proportional to the square of the wire length, extra buffers may need to be inserted to separate the long wire segments in order to optimize the clock delay [32]. Figure 3.17 illustrates the distributed RC ladder network model with the clock buffers added along a long wire, where:

- N represents the size of the clock buffer
- R_t represents the resistance of the clock buffer
- C_g and C_d represent gate and drain capacitance of the clock buffer

- M represents the total number of the distributed wire segments separated from the long wire
- R_w and C_w are the distributed resistance and capacitance per segment length
- X represents the total segment length of the long wire

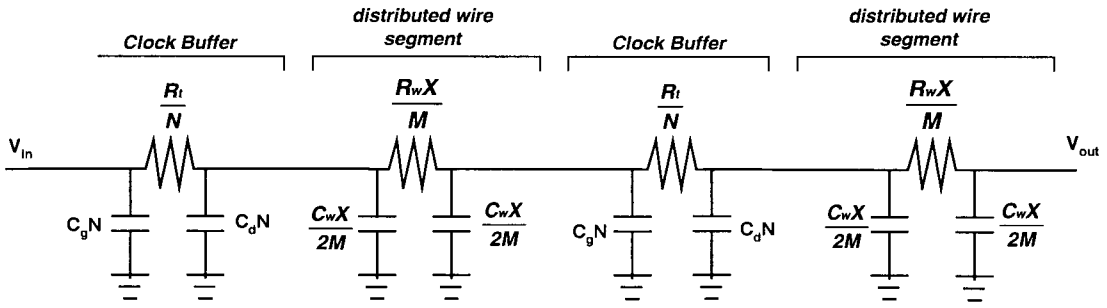


Figure 3.17 Distributed RC ladder network model for a long clock wire segment

Assume that each clock buffer is followed by a distributed wire segment. According to the above distributed RC ladder network model, the delay of each distributed segment is equal to the delay for each distributed wire segment plus the delay of clock buffer associated with the segment, which is:

$$T_{distributed\_segment} = \frac{R_t}{N}(C_dN + \frac{C_wX}{2M}) + (\frac{R_t}{N} + \frac{R_wX}{M})(\frac{C_wX}{2M} + C_gN) \quad (3.3)$$

The total delay of the whole segment becomes:

$$T_{segment} = M * T_{distributed\_segment} \quad (3.4)$$

By differentiating Equation (3.4) with respect to N and M, we obtain the following equations to determine the optimal values for the buffer size and the length per distributed segment:

$$N = \sqrt{\frac{R_tC_w}{R_wC_g}} \quad (3.5)$$

$$M = \sqrt{\frac{R_w C_w X^2}{2 * R_t * (C_d + C_g)}} \qquad (3.6)$$

Users can calculate the optimal buffer size by applying Equation (3.6) and put the appropriate values for *buffer_R*, *buffer_Cin*, and *buffer_Cout* in the *architecture file*. The values for *Rwire* and *Cwire* depend on the metal used for the clock wire. The modified VPR then employs the Equation (3.6) to determine the number of buffers required for a particular clock wire segment. The total capacitance of the clock network is:

$$C_{Clock\_Network} = C_{clock\_buffers} + C_{clock\_wires} \qquad (3.7)$$

This capacitance will be used to estimate the power dissipated by the clock network in the next chapter.

3.3 Summary

The original VPR framework has been modified to incorporate two modules: an activity estimator and a power model. Details of these modules will be discussed in next chapter. In this chapter, three main groups of parameters (including logic block, routing, and clock network) have been introduced. The logic block parameters determine the internal features of a logic block. The routing parameters define the connections among the logic blocks and routing tracks. The routing-resource graph created by VPR contains capacitance information, which is going to be essential for the power calculation. Since the clock dissipation is critical for the power analysis of the whole circuit, we assumed an H-tree clock network for the FPGA architectures. The total capacitance of the clock network will be used for the power estimation in next chapter.

Chapter 4

4 POWER MODEL

Power estimates are more accurate when performed during late stages of logic synthesis [12]. Therefore, the power model developed for this project estimates power after placement and routing when most implementation details have been determined. The power model incorporated into the VPR CAD tool has two modules: the activity generation module, and the power estimation module. The first module employs a transition density model to determine the switching activities inside the circuit. The second module estimates the power consumption at the transistor level based on the results from HSPICE simulations. The model was calibrated assuming a 0.18um CMOS technology. However, the model is general enough to apply to any technology.

4.1 Activity Generation

The model employs the transition density signal model described in Section 2.3.2 to determine switching activities [53]. A filter has also been used to take into account the effect of inertial delays of logic gates [50]. Both the transition density signal model and the filter are discussed in the following sections.

4.1.1 Transition Density Signal Model

The transition density technique introduced in Section 2.3.2 can be applied directly to estimate the switching activity of nets that connect logic blocks within an FPGA. The transition density

and static probability are calculated recursively from the primary inputs to the primary outputs. For each LUT, the function implemented can be expressed as $f(x)$. For each input, x_i, two Boolean functions can be derived by setting input x_i to 1 and 0 respectively. The *Boolean difference* at the output with respect to an input, x_i, is [80]:

$$\frac{\partial f(x)}{\partial x_i} = f(x)\Big|_{x_i=1} \oplus \; f(x)\Big|_{x_i=0} \tag{4.1}$$

The probability of the *Boolean difference*, $P(\partial f(x)/\partial x_i)$, represents the static probability of a change in x_i causing a change in the output. Each input contributes a static probability, $P(\partial f(x)/\partial x_i)$, and a transition density, $D(x_i)$. The total density, $D(y)$, at the output is [80]:

$$D(y) = \sum_{\text{all inputs}} P\left(\frac{\partial f(x)}{\partial x_i}\right) \cdot D(x_i) \tag{4.2}$$

Even though the original transition density model applies only to combinational circuits, it can be extended to sequential circuits. For D flip-flops, the output probability can be set the same as the input probability. The transition density of the output, $D(y)$, of the flip-flop is the same as its transition probability, $P_t(y)$, which can then be written as:

$$D(y) = P_t(y) = 2 \cdot P(x) \cdot (1 - P(x)) \tag{4.3}$$

where $P(x)$ represents the input probability. For each sequential feedback loop, a mutual probability is determined for the output of the D-flip-flop through iterations. Even though this is an approximate method for calculating the transition probabilities of the feedback loops, a previous study shows that the average error obtained by using this method for three iterations is within 5% compared to the exact transition probabilities values [64]. Therefore, we are confident that a small amount of iterations is sufficient to achieve reasonable accuracy.

4.1.2 Sample Calculation for the Transition Density Model

To illustrate the algorithm, consider the example shown in Figure 4.1 and Table 4.1. In this example, each logic block contains a single 2-input LUT. As the figure illustrates, logic block A implements an AND function; logic block B implements an OR function; and logic block D implements an XOR followed by a flip-flop, with an internal feedback between the output of the flip-flop and one of the XOR inputs.

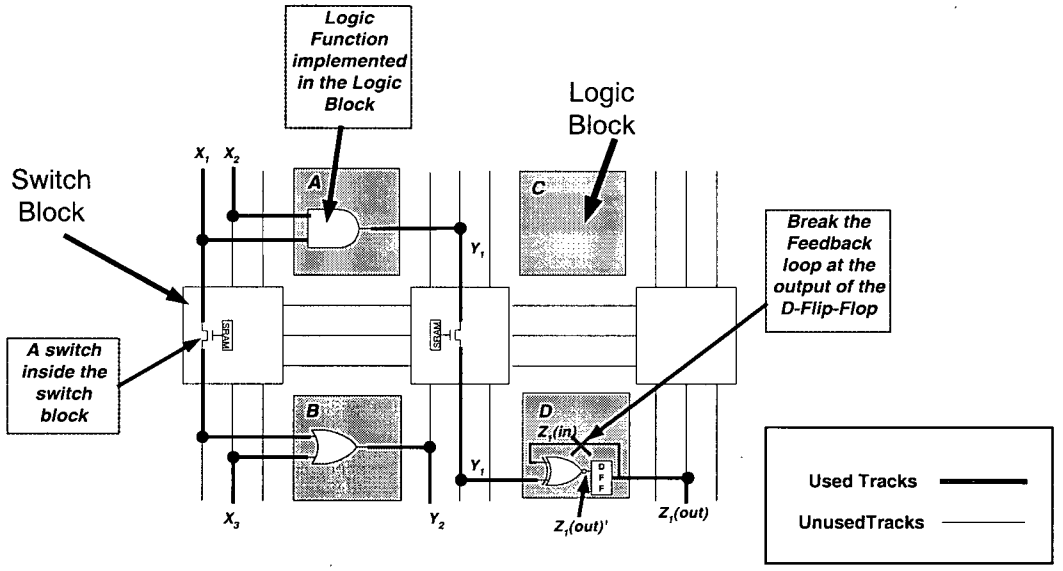


Figure 4.1 Logic functions implemented in the FPGA

Table 4.1 List of logic functions implemented

Logic Block	Logic Function
A	$f_A(X_i) = X_1 \cdot X_2$
B	$f_B(X_i) = X_1 + X_3$
D	$f_D(X_i) = Y_1 \cdot \overline{Z_1(in)} + \overline{Y_1} \cdot Z_1(in)$

Table 4.2 List of assumed transition densities and static probabilities

Nodes	Transition Density	Static Probability
X_1	0.3	0.5
X_2	0.2	0.4
X_3	0.4	0.1 .

The transition density of each node in this example is calculated as follows. Initially, we break the feedback loop in logic block D, creating a combinational circuit (the lower input to the XOR gate is treated as a primary input, $Z_1(in)$. The transition density algorithm from [53] is then used on this combinational circuit. In this example, assume the transition densities and static probabilities of the inputs are shown in Table 4.2. Since the static probability of node $Z_1(in)$ is unknown, we initially assume a value of 0.5. Using the algorithm from [52], this leads to the static probabilities for Y_1, Y_2, and $Z_1(out)$ listed in Table 4.3.

Table 4.3 Static probability calculations

Nodes	Static Probability
Y_1	$P(Y_1) = P(X_1) \cdot P(X_2) = 0.5 \cdot 0.4 = 0.2$
Y_2	$P(Y_2) = P(X_1) + P(X_3) = 0.5 + 0.1 = 0.6$
Z_1(out)	$P(Z_1(out)) = P(Y_1) \cdot (1 - P(Z_1(in))) + (1 - P(Y_1)) \cdot P(Z_1(in)) = 0.2 \cdot 0.5 + 0.8 \cdot 0.5 = 0.5$

We then compare the static probability $P(Z_1(out))$ with the assumed static probability $P(Z_1(in))$. In this example, the two quantities are the same, so the probability calculation terminates. If the difference between the two quantities is greater than some threshold (our algorithm uses a threshold of 0.05), we would update our estimate of $P(Z_1(in))$ by setting $P(Z_1(in))$ to $P(Z_1(out))$, and repeating the above algorithm. Once the probabilities have been found, Equations (4.1) and (4.2) are then used to calculate the transition densities for nodes Y_1 and Y_2, and Equation (4.19) is used to calculate the transition density of node $Z_1(out)$. The results of these calculations are summarized in Table 4.4.

Table 4.4 Boolean difference and transition density calculations

Node	Boolean Difference	Transition Density
Y_1	$\frac{\partial f_A}{\partial X_1} = X_2$ $\frac{\partial f_A}{\partial X_2} = X_1$	$D(Y_1) = P\left(\frac{\partial f_A}{\partial X_1}\right)\cdot D(X_1) + P\left(\frac{\partial f_A}{\partial X_2}\right)\cdot D(X_2)$ $= P(X_2)\cdot D(X_1) + P(X_1)\cdot D(X_2)$ $= 0.4\cdot 0.3 + 0.5\cdot 0.2 = 0.22$
Y_2	$\frac{\partial f_B}{\partial X_1} = \overline{X_3}$ $\frac{\partial f_B}{\partial X_3} = \overline{X_1}$	$D(Y_2) = P\left(\frac{\partial f_B}{\partial X_1}\right)\cdot D(X_1) + P\left(\frac{\partial f_B}{\partial X_3}\right)\cdot D(X_3)$ $= (1 - P(X_3))\cdot D(X_1) + (1 - P(X_1))\cdot D(X_3)$ $= 0.9\cdot 0.3 + 0.5\cdot 0.4 = 0.47$
Z_1(out)		$D(Z_1(\text{out})) = 2\cdot P(Z_1(\text{in}))\cdot[1 - P(Z_1(\text{in}))]$ $= 2\cdot 0.5\cdot 0.5 = 0.5$

4.1.3 Filter

In an actual FPGA, pulses shorter than the propagation delay of a gate are filtered out because the gate cannot respond fast enough [50]. To simulate the filtration effect of circuit inertial delays, a "low pass filter" is modeled at the output of each gate (each LUT for FPGAs). A transition is transmitted across the filter only when the input remains stable over a certain period of time. A probability distribution function estimates the probability that the output of each LUT is stable over a certain period of time. The following equations for the static probability, *P(y)*, and transition density, *D(y)*, at the output of the filter are derived from using the formulas in [50]: (Refer to Appendix A for the mathematical derivation):

$$D(y) = P_F \cdot D(x) \tag{4.4}$$

$$P(y) \approx P_F \cdot \left[\frac{1}{e^{\frac{-\beta \cdot D(x)}{2\cdot(1-P(x))}}} - P(x) \right] \tag{4.5}$$

where:

$$P_F = \frac{e^{\frac{-\beta \cdot D(x)}{2}\left[\frac{1}{P(x)\cdot(1-P(x))}\right]}}{e^{\frac{-\beta \cdot D(x)}{2P(x)}} + e^{\frac{-\beta \cdot D(x)}{2\cdot(1-P(x))}} - e^{\frac{-\beta \cdot D(x)}{2}\left[\frac{1}{P(x)\cdot(1-P(x))}\right]}}$$

where:

- P(x) represents the input probability of the filter
- D(x) is the input density of the filter
- β is the percentage of the rise and fall time relative to the clock cycle

4.1.4 Implementation of the Activity Estimator

Activity estimation is carried out in three steps: LUT organization, static probability calculation and transition density calculation. The pseudo-code for the algorithm is shown in Figure 4.2. First, LUTs are ordered from the primary inputs to the primary outputs. For sequential circuits, the outputs of the flip-flops are initially assumed to be primary inputs. Then, the calculation of the static probability is carried out for each LUT. In sequential circuits, several iterations may require to obtain a probability for each LUT output, as described in Section 4.1.2. Finally, the transition density calculation is performed. As part of the transition density calculation, the CAD tool determines whether glitches exist at the output of each LUT. Signals with a transition density value higher than one are considered to have glitches. The filter mechanism described in Section 4.1.3 is used to filter out the inertial delay for these signals to avoid unrealistic activity values. In our implementation, the β value used for the filter is set to 0.1; this corresponds to a rise and fall time is approximately 10% of the clock signal.

```
Organize LUTs from primary inputs to primary outputs;

/* Static Probability Calculation */
do {
    For (each LUT in the organized order)
    {
        Calculate static probability;
        Update the static probability in the database;
    }
} until (static probability difference < error tolerance)

/* Transition Density Calculation */
for (each LUT in the organized order)
{
    Calculate transition density;
    If glitches exist at the output
    {
        apply the filter (static probability, transition density);
    }
    Update transition density and static probability in the database;
}
```

Figure 4.2 Pseudo-code of the transition density algorithm

4.2 Power Estimation

After the switching activities have been determined, the next step is to analyze the power dissipation at transistor level for each component inside the FPGA. The average power consumption in digital circuits consists of three main components: *dynamic*, *short-circuit* and *leakage* power [32]. The estimation methodology for these three components will be described and a comparison with HSPICE will be given in the following sections.

A significant portion of a FPGA is composed of SRAM cells. The values stored in these cells do not change after the configuration process. Therefore, these SRAM cells play no role in the dynamic and short-circuit power estimation, but they are considered in the leakage power estimation.

4.2.1 Capacitance Model

Even though the *architecture file* defines most of the capacitance values, it does not describe embedded capacitance, such as transistors inside the LUTs, multiplexers, and buffers of the logic block. To estimate the power consumption of this embedded capacitance, the capacitance model designed for the CACTI project [71] has been used. The CACTI capacitance model is concerned with the gate capacitance and the drain capacitance for small and large transistors. The values for C_{gate} , $C_{polywire}$, $C_{diffarea}$, $C_{diffgate}$, and $C_{diffside}$ in the following formulas are process dependent parameters. Figure 4.3(a) shows the layout of a transistor that is smaller than 30 times the minimum transistor width (30X). The total gate capacitance, which includes the gate oxide capacitance and the polysilicon capacitance, is estimated as follows:

$$\text{Gate Capacitance} = W \cdot L_{eff} \cdot C_{gate} + L_{poly} \cdot L_{eff} \cdot C_{polywire} \quad (4.6)$$

where:

- L_{eff} is the effective length of the transistor
- W is the width of the transistor
- L_{poly} is the length of the polysilicon line going into the gate
- C_{gate} is the gate oxide capacitance per unit area
- $C_{polywire}$ is the polysilicon capacitance per unit area

Equation (4.6) can be used for different types of transistors, including NMOS and PMOS, with different C_{gate} and $C_{polywire}$ values.

The source/drain capacitance consists of the area and the perimeter components. The drain capacitance is estimated as follows:

$$\text{Drain Capacitance} = 3L_{eff} \cdot W \cdot C_{diffarea} + (6L_{eff} + W) \cdot C_{diffside} + W \cdot C_{diffgate} \quad (4.7)$$

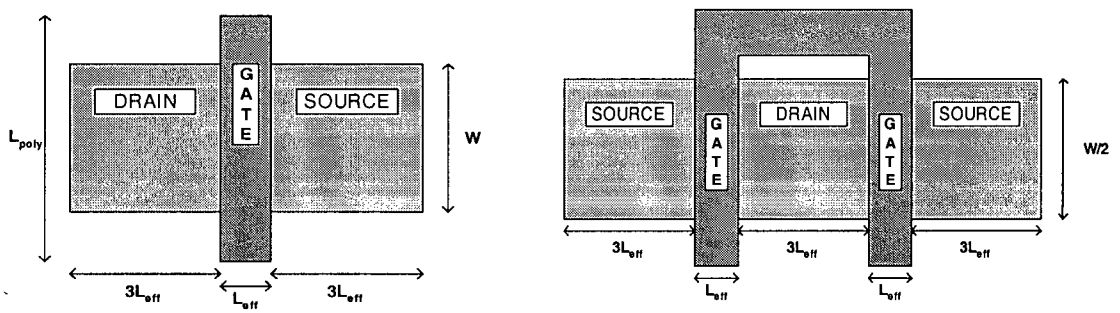
- $C_{diffarea}$ represents the area junction capacitance
- $C_{diffside}$ is the sidewall junction capacitance
- $C_{diffgate}$ is the sum of the zero-bias gate-edge sidewall bulk junction capacitance and the gate-drain overlap capacitance

Figure 4.3(b) shows the layout of a transistor with the width larger than 30X. Folding is assumed for large transistors. As a result, the source and drain capacitance for the transistor can be estimated as:

$$\text{Source Capacitance} = 2 \cdot \left[3L_{eff} \cdot \frac{W}{2} \cdot C_{diffarea} + \left(6Leff + \frac{W}{2} \right) \cdot C_{diffside} + \frac{W}{2} \cdot C_{diffgate} \right] \quad (4.8)$$

$$\text{Drain Capacitance} = 3L_{eff} \cdot \frac{W}{2} \cdot C_{diffarea} + 6L_{eff} \cdot C_{diffside} + W \cdot C_{diffgate} \quad (4.9)$$

Note that both Equations (4.8) and (4.9) can be applied to NMOS and PMOS transistors, but with unique values for $C_{diffarea}$, $C_{diffside}$, and $C_{diffgate}$ for each type of transistor.



a) smaller than 30X b) larger than 30X

Figure 4.3 Layout of transistors

4.2.2 Dynamic Power

Dynamic power is the dominant component of the total power. It is dissipated every time a signal changes due to the charging and discharging of the load and parasitic capacitance. Therefore, dynamic power is closely related to the transition density of all nodes inside the circuit. The total dynamic power dissipation can be expressed as [80]:

$$\text{Dynamic Power} = \sum_{\text{all nodes}} 0.5 \cdot C_y \cdot V_{supply} \cdot V_{swing} \cdot D(y) \cdot f_{clk} \qquad (4.10)$$

where

- V_{swing} is the swing voltage of each node
- V_{supply} is the supply voltage
- C_y is the node capacitance being charged and discharged during each transition
- D(y) is the individual transition density at each node
- f_{clk} is the clock frequency of the circuit, which is bounded by the critical delay path of the circuit

4.2.2.1 Routing Fabric

A large part of the dynamic power is due to switching tracks within the routing fabric of the FPGA. Figure 4.4 shows a typical routing track with the attached buffers and switches (the optimum buffer sizes have been determined in a previous study [10]). The capacitance along this track is due to :

- The metal capacitance of the track
- The parasitic capacitance of all switches attached to the track

The raw values of the capacitance for each element are from the architecture file; our implementation uses the routing resource graph to determine exactly how many switches are

attached to each track, and determines the actual capacitance of each track. For each net in the design, the capacitance of all tracks that are used to route the net are summed, and the activity of the net is then used, along with this capacitance, to calculate the power dissipated by that net. This is summarized in Figure 4.5.

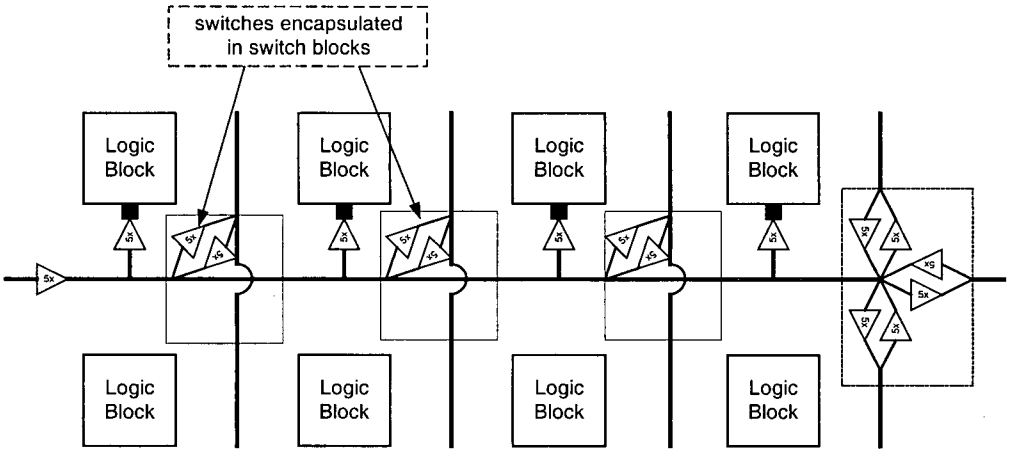


Figure 4.4 Wiring assumed for dynamic power experiment

```
Total power = 0;
for each net i
{
        alpha = activity of net i;
        net power = 0;
        for each track used to route this net
        {
                net power += switching power from Equation (4.10);
        }
        total power = total power + net power;
}
```

Figure 4.5 Overview of algorithm to calculate dynamic power of routing fabric

To verify the model, an HSPICE model was created. Figure 4.6 shows the power predicted by the model along with the power predicted by HSPICE, for a range of segment lengths. The wires were switched at 20MHz, to ensure that the wires had fully charged or discharged during

each cycle. As the graph shows, the model results match the HSPICE results very closely, with an average error of 4.8%.

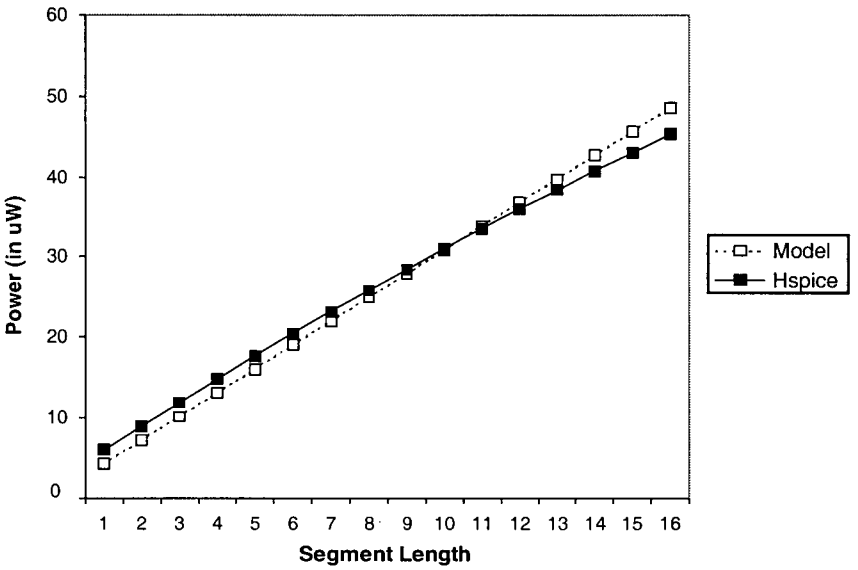


Figure 4.6 Dynamic power for a one segment at 20Mhz

4.2.2.2 Logic Block

In a typical logic block, there are set/reset logic, LUTs, large multiplexers, and D-flip-flops. The set/reset logic dissipates only negligible dynamic power because the set/reset logic does not get switched often during normal operation. The dynamic power dissipation is estimated based on the architecture shown in Figure 4.7; the model has been designed to account for any number of internal routing wires, LUTs, multiplexers, and D-flip-flops. Each input or output of the logic block is assumed to contribute one segment length of Metal 1 or Metal 2 wire to internal routing. The dynamic power consumed by the internal routing can be calculated using Equation (4.10). The dynamic power dissipated inside the LUTs, multiplexers, and D-flip-flops can be calculated as follows.

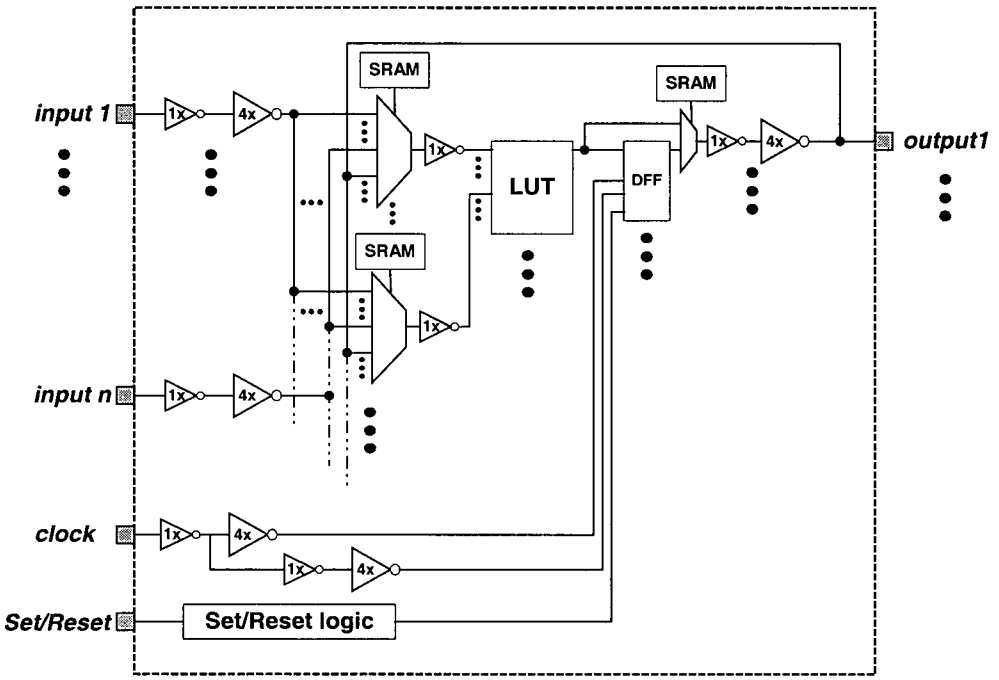


Figure 4.7 Schematic of a logic block (from [10])

To determine the transition densities of nodes within the LUTs and multiplexers, the LUTs and multiplexers are modeled as trees of 2-input multiplexers. Using the transition density model discussed in Section 2.3.2, the static probability and transition density at the output of each two-input multiplexer is:

$$\begin{aligned} P(\text{output}) = {} & \overline{P(\text{in}_0)} \cdot P(\text{in}_1) \cdot P(\text{select}) \\ & + P(\text{in}_0) \cdot \overline{P(\text{in}_1)} \cdot \overline{P(\text{select})} \\ & + P(\text{in}_0) \cdot P(\text{in}_1) \cdot \overline{P(\text{select})} \\ & + P(\text{in}_0) \cdot P(\text{in}_1) \cdot P(\text{select}) \end{aligned} \tag{4.11}$$

$$D(\text{output}) = \overline{P(\text{select})} \cdot D(\text{in}_0) + P(\text{select}) \cdot D(\text{in}_1) + \left(P(\text{in}_0) \oplus P(\text{in}_1)\right) \cdot D(\text{select}) \tag{4.12}$$

The modeling of a 2-input LUT using multiplexers is illustrated in Figure 4.8. The output of each multiplexer is an internal node inside the LUT. Each node is associated with three

source/drain capacitance and one gate capacitance. Note that the internal nodes inside the LUT can be affected by the body effect of the pass transistors, and may swing at a degraded supply voltage.

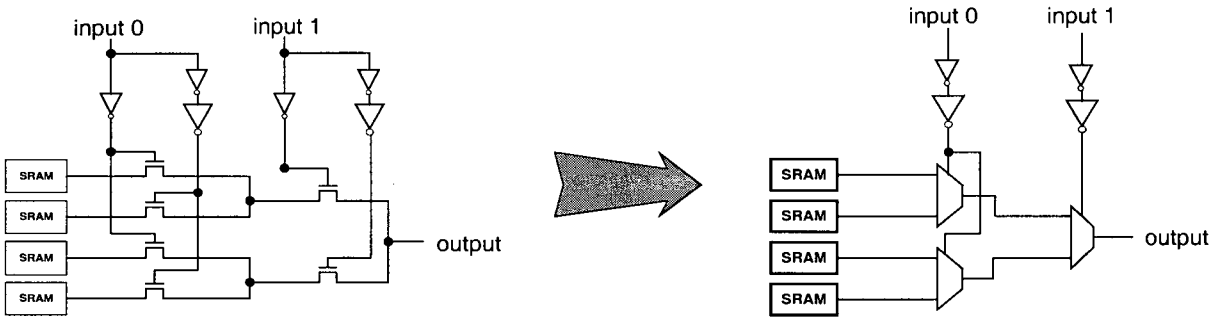


Figure 4.8 Modeling of a 2-Input look-up-table using 2-input multiplexers

HSPICE simulations have been executed to evaluate the power estimation for different input densities and for different LUT sizes (size 2 to 7). As it is impossible for us to evaluate all 2^{2^k} functions for each k-input LUT, the worst case scenario is considered. For each different k-LUT, a function is assigned so that a change in the least significant signal affects all of the internal nodes. Figure 4.10 demonstrates the worse case function for a 3-input LUT. Note that a change in the least significant input, In_0, impacts all internal nodes. To estimate the power dissipated at all internal nodes in the LUT, the transition density of each node is calculated using Equations (4.11) and (4.12), and the power is then calculated using Equation (4.10). In all cases, we assumed minimum-sized transistors.

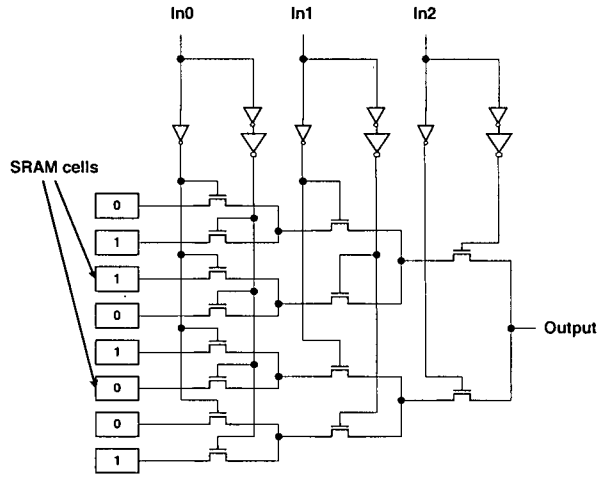


Figure 4.9 Example of worse case function for a 3-input LUT

To verify the model, an HSPICE simulation was used. The power predicted by the HSPICE simulation depends heavily on the *relative* switching times of the inputs. Since our model does not take this into account, we performed several thousands of HSPICE simulations, each with different relative input switching times. The maximum, minimum, and average values for the average power obtained from these simulations for a 4-LUT is shown in Figure 4.10. The power estimate from our model is also shown on the graph; as the graph shows, our estimate clearly lies within the maximum and minimum HSPICE predictions in all cases. The graph also illustrates that the model power is closer to the maximum HSPICE prediction than the minimum prediction; this is expected because the transition signal density model assumes all inputs to the multiplexers are switching at different times – the worst-case scenario.

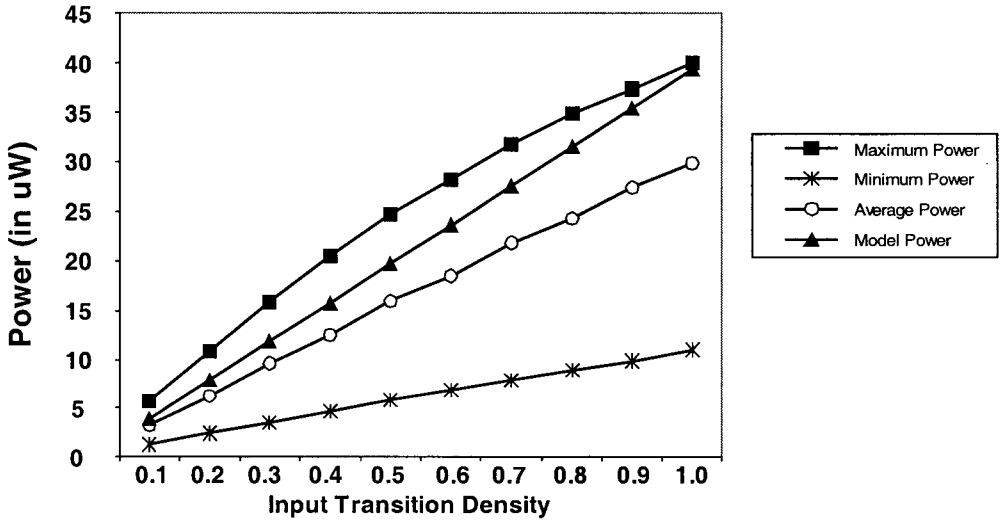


Figure 4.10 Power versus average input transition densities for a 4-Input LUT

We then fixed the density of each input to 0.5 and varied the LUT size between 2 and 7, and repeated the experiments. Figure 4.11 shows the results. Again, the model results are slightly less than the maximum HSPICE predictions. Note that, in both figures, the *fidelity* between the model predictions and HSPICE results is very good. The average difference between the maximum HSPICE prediction and the modeled values is 14.5%.

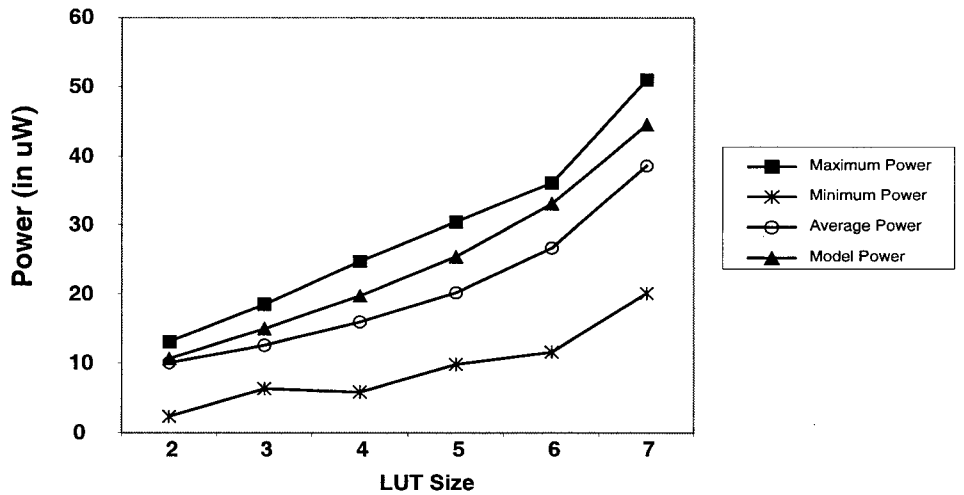


Figure 4.11 Power versus different LUT sizes with an input transition density of 0.5

Since the schematic of a multiplexer is very similar to that of a LUT (see Figure 4.12), the calculation for the transition density for a multiplexer is similar. In this case, however, the gate voltage of the pass transistors in the SRAM cells has been boosted, so that the internal nodes within the multiplexers are not affected by the body effect. Moreover, there is an overestimation if the power estimation method used for the LUT is applied directly to the multiplexer because some internal nodes within the multiplexers are always correlated. Consider the 4-input multiplexer in Figure 4.12. When input 0 is selected, transistors A, C, and E are turned on. Since the SRAM cells are stable during the operation phase, all the selected transistors are *on* during operation phase. As a result, the internal node at the output of the 2-input multiplexer representing the transistors A and B should toggle at the same time as the output of the LUT. Since the transition density signal model assumes all nodes switching at different times, it overestimates the power dissipation of the multiplexers.

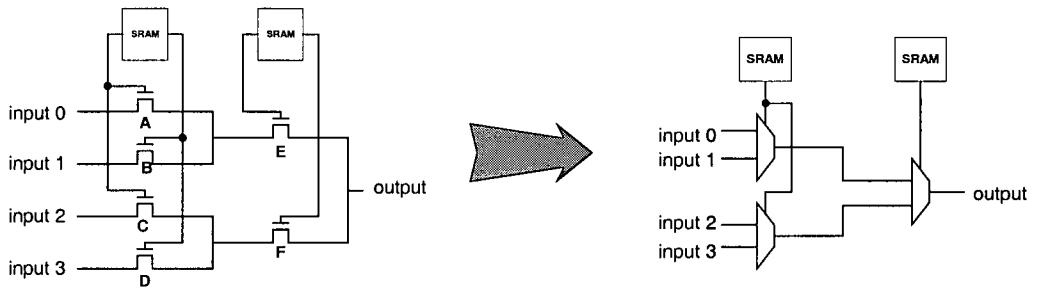


Figure 4.12 Modeling of a 4-input multiplexer using 2-input multiplexers

Compared to the HSPICE simulations with different input densities and different multiplexer sizes, the power estimates using the same method as that applied for LUTs are 21.8% larger (see Figures Figure 4.13 and Figure 4.14). To represent a more realistic operation of the multiplexers, a correlation factor of 0.8 has been inserted into the original power estimation. As a result, the power estimation for the multiplexer is found as follows:

1. Estimate the power using the same method as that for LUTs

2. Multiply the power estimate by the correlation factor

When the correlation factor is used, the absolute error between the power estimates and the HSPICE simulation results drops to 5.3%. The relationship between the dynamic power dissipation and the input densities is shown in Figure 4.13, and the relationship between the power estimates and multiplexer sizes is shown in Figure 4.14.

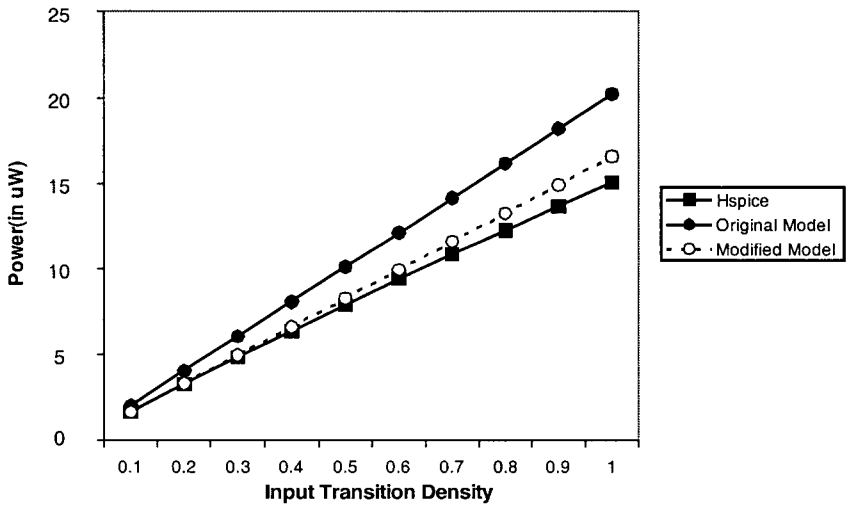


Figure 4.13 Power versus different average input transition densities for a 16-Input Multiplexer

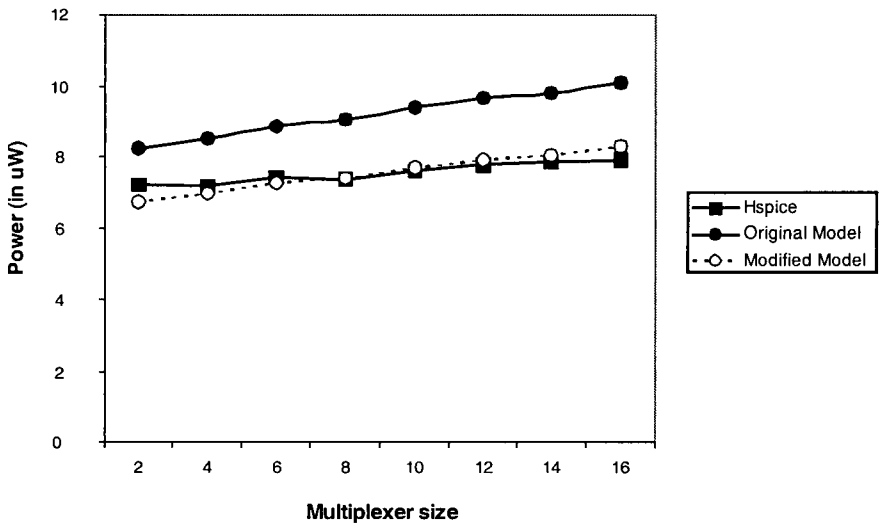


Figure 4.14 Power versus multiplexer size at an average input transition density of 0.5

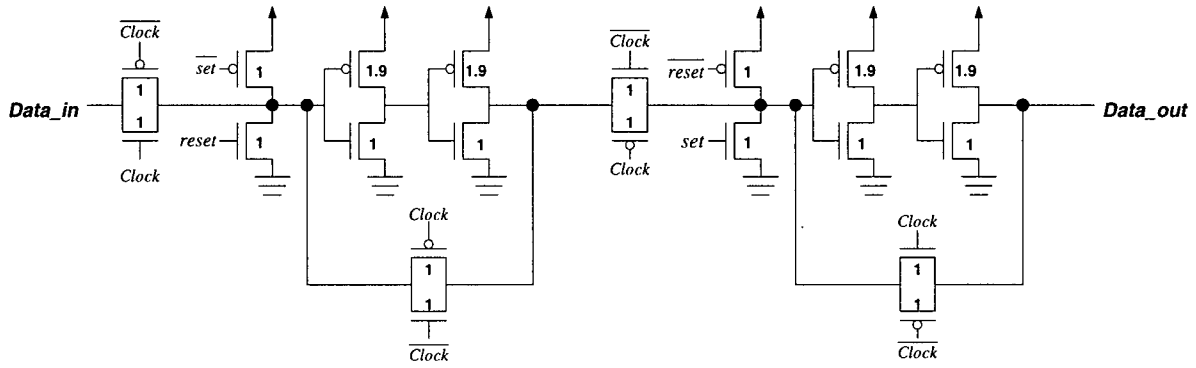


Figure 4.15 D-flip-flop with asynchronous set and reset

Each logic block contains one or more flip-flops. The schematic of the D-flip-flop assumed in VPR is shown in Figure 4.15. The internal switching activity inside a D-flip-flop can be much higher than the transition density of the input signal. HSPICE simulations at different clock frequencies were performed to investigate the relationship between the input densities and the power dissipated by the D-flip-flop. Based on the simulation results, the following formula has been derived using Matlab:

$$\text{Dynamic Power(DFF)} = 0.5 \cdot C_{DFF} \cdot (\text{Effective Density}) \cdot V_{supply} \cdot V_{swing} \cdot f_{clk} \qquad (4.13)$$

$$\text{Effective Density} = (-0.074) \cdot D(\text{input}) + (5.2486) \cdot D(\text{input})^2 \qquad (4.14)$$

where:

- D(input) is the density of the input signal for the D-flip-flop
- C_{DFF} is the total capacitance inside the D-flip-flop
- V_{supply} is the supply voltage
- V_{swing} is the swing voltage
- f_{clk} is the clock frequency

Figure 4.16 shows the relationship between the input density and the power consumption within the D-flip-flop. The average difference between the power estimates and the HSPICE simulation results is 10.5%.

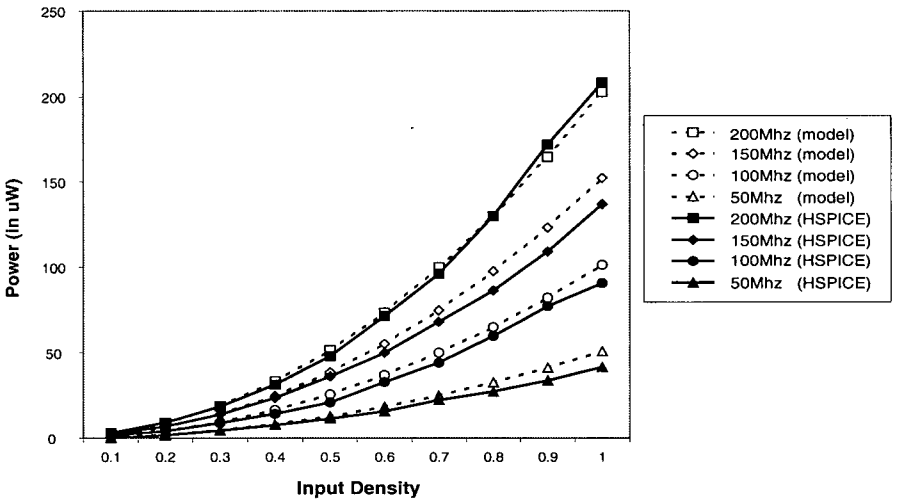


Figure 4.16 Dynamic power of D-flip-flop versus input density

To investigate the overall error for our dynamic power estimate of each logic block, we used a 4-input LUT, which is widely used in commercial FPGAs [4][6][38][72][77][78]. Clearly, the dynamic power dissipated by a component is proportional to the capacitance contributed by that component. Table 4.5 shows the following factors:

- the capacitance of each component at a percentage of the total capacitance
- the average error of the power estimate for that component
- the impact of this error on the overall error

From Table 4.5, the total error of the dynamic power estimate is 8.4%. We expect the total error for other sizes of logic blocks to be close to that of 4-LUT since the amount of internal routing, the size of LUTs and multiplexers, and the number of D-flip-flop increases together at approximately the same rate.

Table 4.5 Dynamic error calculation for a 4-LUT

Component	Capacitance Percentage	Error per component	Relative Error to the total power estimate
LUTs	22.1%	14.5%	3.2%
Multiplexers	36.6%	5.3%	1.9%
D-flip-flop	21.9%	10.5%	2.3%
Internal Routing	19.4%	4.8%	0.9%
Total Error			8.4%

4.2.3 Short-Circuit Power

Short-circuit power is the power dissipated by the inverters in the circuit through a direct current path formed between the power supply and the ground during the rise and fall time of each transition. Short circuit power is a function of the rise and fall time and the load capacitance [22][66][67].

To analyze the short-circuit power dissipation, HSPICE simulations have been carried out for three sizes of inverters:

- The minimum size (1X)
- Five times the minimum size (5X)
- Ten times the minimum size (10X)

A load scale is assigned for the load capacitance of each inverter. We define the load scale of an inverter to be the ratio of the load capacitance to the output capacitance of the inverter. For example, if the load capacitance is 10 times of the output capacitance of the inverter, the load scale for the inverter is 10. By fixing the load scale and increasing the rise and fall times, we obtained the summation of the dynamic power and short-circuit power from HSPICE. The short-circuit power can be found using Equation (4.15). The dynamic power is determined by setting the rise and fall time to the time step (5 ps) in HSPICE.

$$\text{Short-Circuit Power} = \text{Total\_Power(from HPSICE)} - \text{Dynamic\_Power} \qquad (4.15)$$

According to our simulation results, all of the three sizes of inverters demonstrate the same trend. Figure 4.17 illustrates the relationship among the power dissipation, rise and fall time, and the load scale for the 1X inverter. The short-circuit power is expressed as a percentage of the dynamic power in the graph. As the rise and fall time increases, the short-circuit power dissipated increases very dramatically when the load scale is small. However, when the load scale is large, the increase is less dramatic. This observation can be explained as follow.

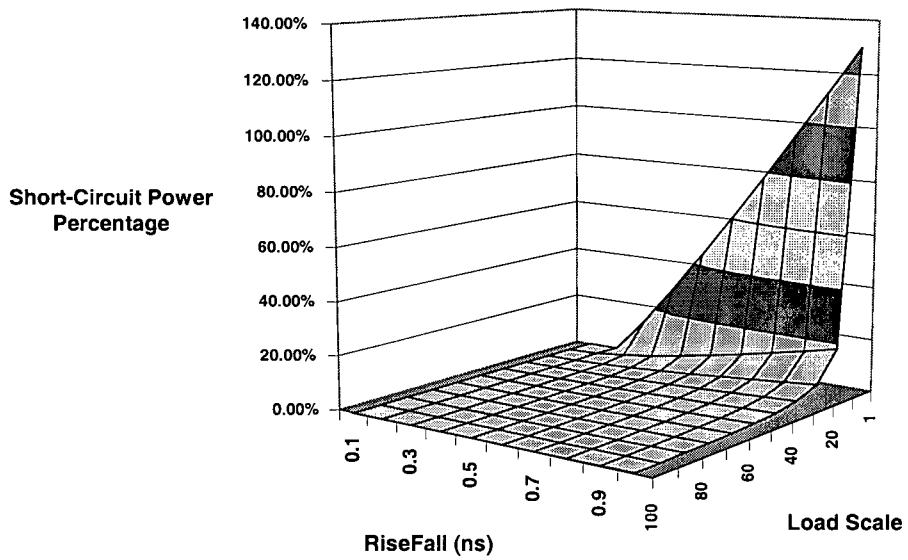


Figure 4.17 Short-circuit power percentage for the 1X inverter

Consider an inverter with a large load. During a rising transition at the input, the output voltage remains almost the same as the supply voltage until the input transition is completed. Therefore, although both the PMOS and NMOS transistors are on, the current flow through the PMOS transistor is very small. Similarly, during a falling transition at the input, the output voltage remains at approximately zero until the input voltage completes its swing. As a result, even though both transistors are on simultaneously, the current conducted through the NMOS transistor is small because of the small drain-to-source voltage [32]. In other words, a larger

load capacitance for an inverter leads to a smaller amount of short-circuit current flowing through the transistors during the rise and fall time. Thus, an inverter with a larger load capacitance dissipates less short-circuit power.

In FPGAs, the load of the inverters depends on the length of the wire segment and the internal population of wire switches and connection switches. As the majority of the dynamic power comes from the routing, we focus on the short-circuit power dissipated by the routing. To investigate the worse case scenario, the wiring assumption in Figure 4.4 is used to load one wire of segment length one. Figure 4.18 illustrates the short-circuit power as a percentage of the dynamic power for the 5X tri-state buffer as a function of the rise and fall time. According to the datasheets of Altera APEX20KE and Xilinx Virtex-E, the range for the setup time for the input signals to the register in each logic block is from 0.16ns to 1.1ns. We assume that the rise and fall time of the signals in the FPGA model matches the setup time specified in the datasheets. The short-circuit power percentage for the tri-state buffer is 8.5% at 1.1ns.

The short-circuit power can be incorporated in the dynamic power as shown Equation (4.16) , in which ϕ represents the short-circuit power percentage relative to the dynamic power estimated in Section 4.2.2. To be conservative, the short circuit power dissipated for our model is set to be 10% of the dynamic power.

$$\text{Total Dynamic Power} = (1+\phi)\cdot \text{Dynamic\_Power} \qquad (4.16)$$

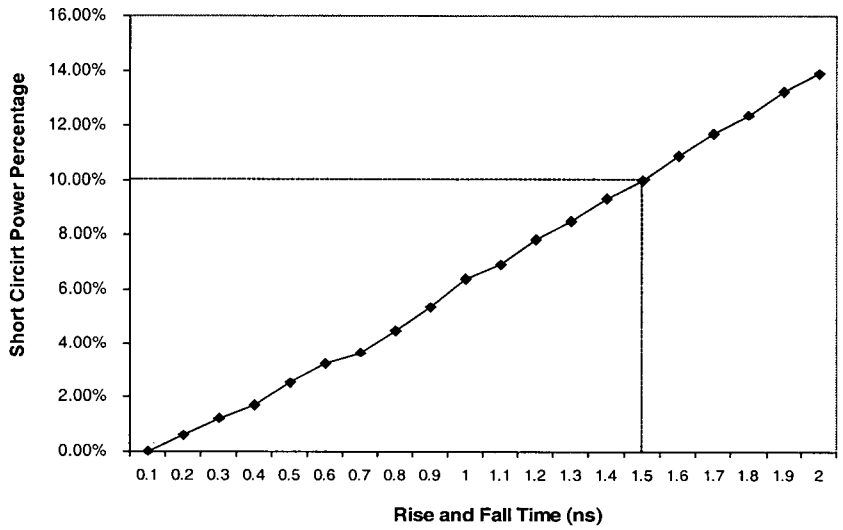


Figure 4.18 Short-circuit power percentage for one segment length

4.2.4 Leakage Power

The leakage power dissipation comes from two sources: reverse-bias leakage power and sub-threshold leakage power. As the majority of the leakage power is from the sub-threshold current [33], the reverse bias leakage current is assumed to be negligible. A first-order estimation model is applied to estimate the sub-threshold current [32]:

$$I_{drain}(\text{weak inversion}) = I_{on} \cdot \exp\left[\frac{(V_{gs} - V_{on})q}{nkT}\right] \tag{4.17}$$

where V_{on} is the boundary between the weak and strong inversion regions. The following equation is used to calculate V_{on} :

$$V_{on} = V_t + \frac{nkT}{q} \tag{4.18}$$

$$n = 1 + \left(\frac{qN_{FS}}{C_{ox}}\right) + \left(\frac{C_d}{C_{ox}}\right) \tag{4.19}$$

where I_{on} is the drain current at the boundary when V_{gs} is equal to V_{on}. The velocity saturation model [63] is employed to calculate I_{on}:

$$I_{on} = \frac{W v_{sat} C_{ox} (V_{gs} - V_t)^2}{(V_{gs} - V_t) + E_c L_{eff}} \tag{4.20}$$

where

- W is the device width
- v_{sat} is electron velocity
- E_c is the piecewise carrier drift velocity
- L_{eff} is the effective source-drain channel length
- V_{gs} is the gate-source voltage
- V_t is the threshold voltage
- C_d is the drain capacitance.
- k is the Boltzman's constant
- q is the elementary charge
- T is the temperature in degree Kelvin

N_{FS} is the number of fast surface states. It is a current fitting parameter that determines the slope of the sub-threshold current-voltage characteristic [32]. Each temperature has a specific N_{FS} value. To determine the N_{FS} values of NMOS and PMOS transistors, HSPICE simulations have been run for both types of transistors, with sizes ranging from 5 times the minimum transistor width to 20 times the minimum transistor width, over the temperature range from -40 to 100 °C [4][72]. To be conservative, the V_{gs} value is assumed to be half of the threshold—0.2V. Figure 4.19 and Figure 4.20 show the estimated and simulated leakage current as a function of different temperature. The average error between the estimated values and the

simulated results is 13.4%. The leakage power is calculated by multiplying the subthreshold current with the supply voltage:

$$\text{Leakage Power} = I_{drain(\text{weak inversion})} \cdot V_{supply\_voltage} \tag{4.21}$$

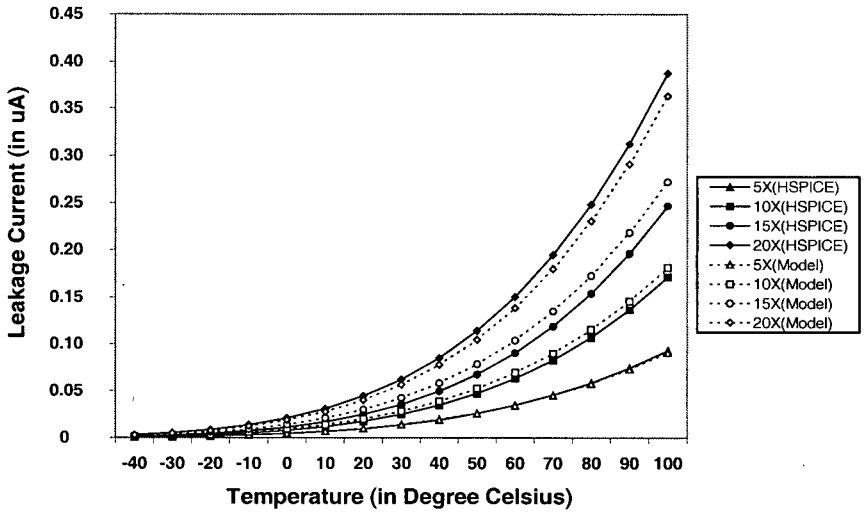


Figure 4.19 Leakage current versus temperature for different sizes of NMOS transistors

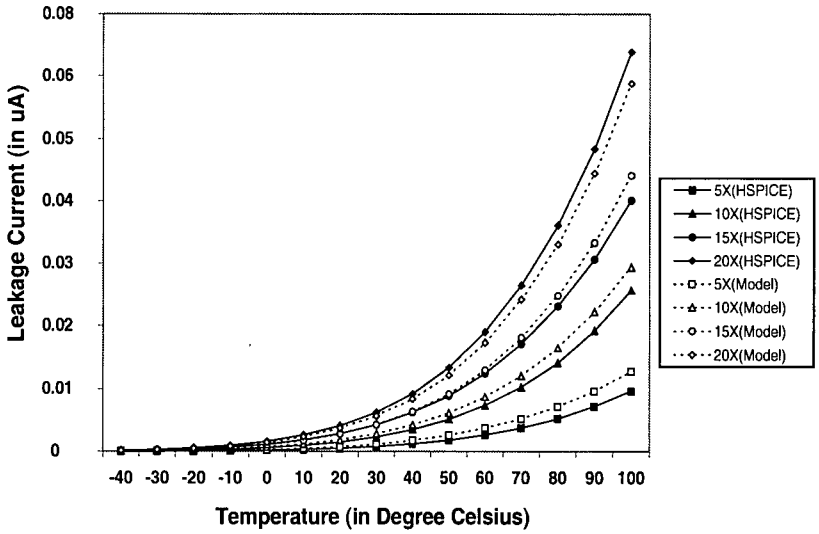


Figure 4.20 Leakage current versus temperature for different sizes of PMOS transistors

4.3 Summary

This power model has been integrated into the VPR CAD tool. The power estimation procedure is performed in two steps. In the activity generation step, a transition density model is employed to determine the switching activities of nodes within the circuit. A filter mechanism has also been incorporated into the model to provide a better estimate on switching activities by eliminating the inertial delays in the logic gates. In the power estimation step, the formulas for power dissipation of each component inside the FPGA architecture has been evaluated by HSPICE simulations. The average error for dynamic power estimation is 4.8% for routing, and 8.4% for the logic block. Short-circuit power is modeled as 10% of the dynamic power dissipation. Leakage current is calculated using a first-order equation with the velocity-saturation model; the average difference between the estimates and the HSPICE results for the leakage current is 13.4%.

Chapter 5

5 EXPERIMENTAL RESULTS AND SENSITIVITY

In the previous chapters, we have discussed our modifications to the parameterized architectural model for VPR and developed a power model to generate activity and power estimates. In this chapter, we will apply the power model to explore the energy dissipation within FPGAs. At the beginning of this chapter, the experimental methodology is discussed. Then, a set of benchmark circuits is used to determine the effect of varying architectural parameters on energy consumption. Finally, the sensitivity of our experimental results is considered.

5.1 Experimental Methodology

The power estimation flow employed for this research project is very similar to the flow discussed in Chapter 2 (see Figure 5.1). First, 36 benchmark circuits from the Microelectronics Center of North Carolina (MCNC) were optimized and mapped using *SIS* [63] and *FlowMap* [18] respectively. An activity estimator is then used to generate activities for all nodes in the mapped circuits. All the primary inputs are assumed to have the same transition density. *TVPack* is applied to group the LUTs and registers into logic blocks based on the user-specified cluster size. Then, *VPR* is used to perform *placement* and *routing* for the circuits. Each circuit is mapped to the smallest square logic block array with sufficient logic blocks and pads. The *VPR router* then determines the minimum number of tracks per channel required to route

the circuit. The power estimation step has been incorporated in VPR to calculate power dissipation based on the specified architecture [10].

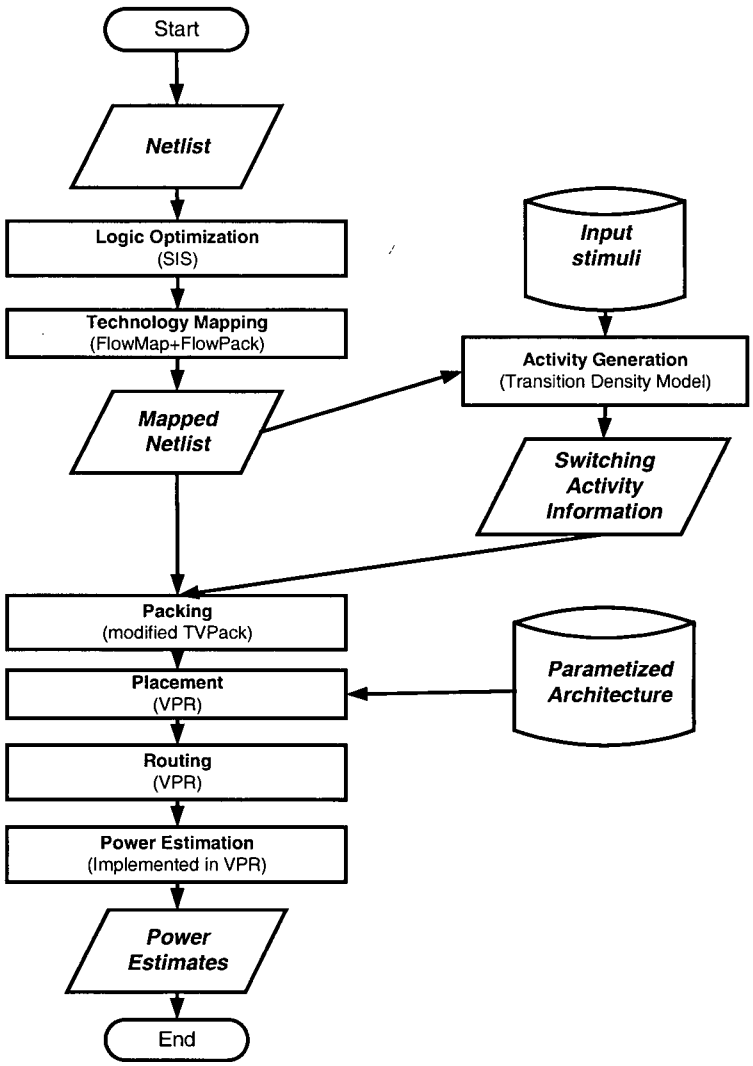


Figure 5.1 Power estimation flow for this project

Energy, instead of power, is used as the optimization goal to avoid bias by the clock frequency of the circuits. The clock frequency is determined by the critical path delay of the circuits after place and route. The formula for the energy is:

$$\text{Energy} = \text{Power} \cdot \text{Critical\_Path\_Delay} \tag{5.1}$$

To determine the parameters for our experiments, we have considered the findings from previous work. It has been shown that [10]:

- Routing architectures with a mixture of pass-transistor and tri-state buffer routing switches are better than those with only one type of switch. Architectures using 50% pass-transistor switched wires and 50% tri-state buffer switched wires result in the best delay.
- The optimal size for tri-state buffer is five times of the minimum buffer size.
- The optimal size for the pass-transistor is ten times its minimum size .
- Wires of length 4 achieves the best combination of low delay and high area-efficiency.
- The Disjoint switch block topology is faster in a FPGA in which segments span more than one logic block.

Therefore, 50% of the wires in the routing architectures used in our experiments are connected using pass-transistors while the rest are connected using tri-state buffers. Moreover, we fixed the wire length to four and made use of the Disjoint switch block topology. The internal population of the wire switches and connection switches is assumed to be 1.

In Section 5.2, the effects of three architectural parameters on energy consumption are shown. In Section 5.3.1, the experiments were repeated using different transition density values for the primary inputs to explore the sensitivity of the transition density. In Section 5.3.2, we re-run the experiments using a *breath-first routing algorithm* to investigate whether the power analysis is sensitive to the routing algorithm.

5.2 Experimental Results

We explored the effects of the three architectural parameters in Table 5.1 on the energy consumption of the FPGA:

Table 5.1 Parameters under investigation

Parameters	Description
Segment_length	The number of logic blocks spanned by each wire segment
Cluster_size	The number of LUTs per logic block
LUT_size	The number of inputs per look-up-table

The parameters are varied one at a time. All primary inputs are assumed to be *normalized random inputs* (with static probability of 0.5 and transition density of 0.5). Also the *timing-driven routing algorithm* in VPR is chosen for our experiments in the section.

5.2.1 Energy versus Segment Length

The *segment_length* parameter is varied from 1 to 16 to investigate the effect of the length of the wire segment on the routing energy. All the wires in each architecture are assumed to have the same length. The four switch block topologies (Disjoint, Universal, Wilton, and Imran) introduced in Chapter 3 are considered. Figure 5.2 shows the relationship between the average routing energy dissipation of the benchmark circuits and the length of each wire segment. The graph shows that circuits dissipate less energy when routed with shorter wires. This means short wire segments are favorable from the energy consumption point of view. Short wires provide the flexibility for routing; as a result, they require fewer wires for connections. Long wires are not preferred especially for short connections due to the large wire capacitance and unused switches that attached to the wire; the large wire capacitance consumes a significant amount of dynamic power and the unused switches contribute leakage power.

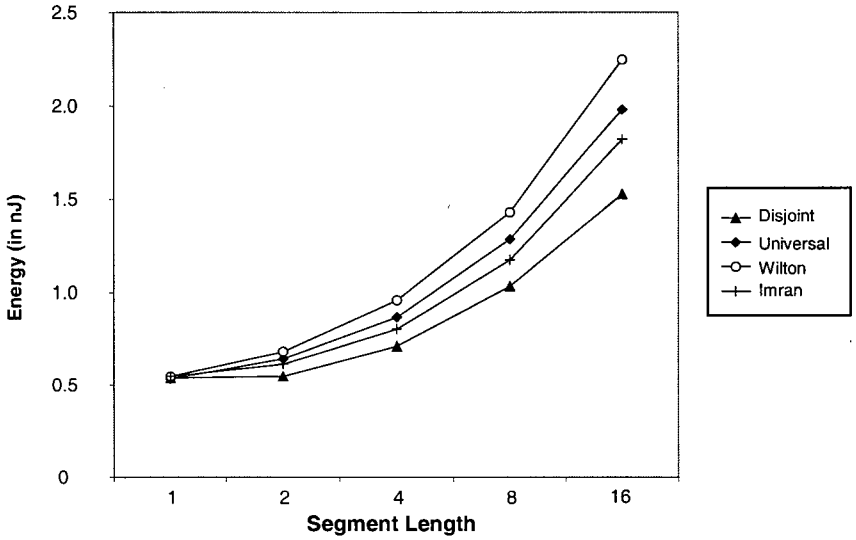


Figure 5.2 Energy vs. segment length

It has been shown that the Disjoint switch block is less flexible compared with other switch block topologies [10]. However, the Disjoint block requires only one switch connection for each wire that bypasses the switch block, while the Universal and Wilton topologies each require four connections. As a result, the Disjoint topology needs fewer routing switches for long wires. Thus, it is more energy-efficient for segment length longer than one, as shown in Figure 5.2. However, FPGAs employ the Disjoint switch box dissipate more routing energy compared to the FPGAs that contain other switch boxes when the segment length is 1. The Imran topology is a mixture of Disjoint and Wilton topology as explained in [46]. According to our experiments, FPGAs using the Imran topology disspates on average 12% more energy than those using the Disjoint topology.

5.2.1 Energy versus Cluster Size

This section studies the impact of cluster size on energy dissipated by the routing, logic blocks, and clock. Figure 5.3 shows the energy dissipation as a function of cluster size. As shown in the graph, increasing the cluster size does not cause any significant change in the routing

energy, but it slightly increases the logic block energy and significantly decreases the clock energy dissipation. This is because as a logic block contains more LUTs, more internal wires for local connections and longer routing wires for block-to-block connections are required, but the logic block can handle more complicated functions. As a result, the total number of logic blocks required decreases, and hence, reduces the number of clock branches and routing branches.

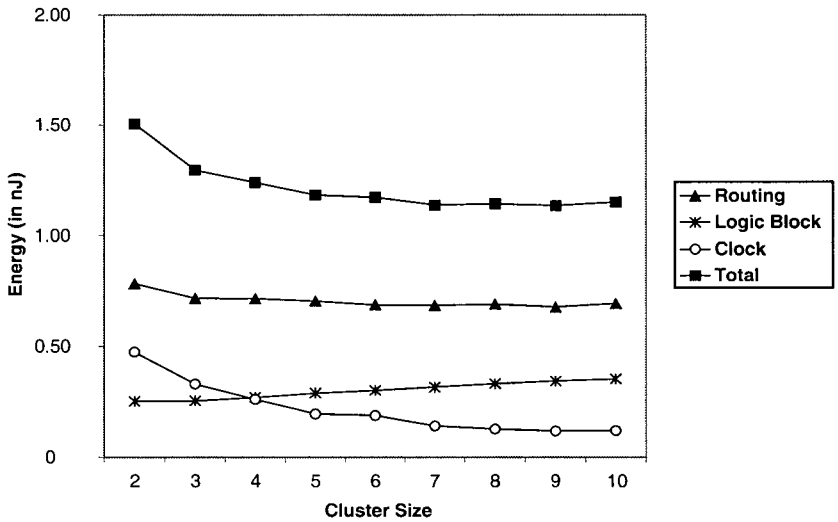


Figure 5.3 Energy versus cluster size

5.2.2 Energy versus Look-Up-Table Size

The energy dissipation as a function of LUT size is shown in Figure 5.4. Energy consumed by both the logic blocks and the routing fabric increase with the LUT size. This is because a larger LUT contains more internal nodes, and results in longer global wiring segments. The behavior of the clock energy is the result of two competing factors: smaller LUTs means more LUTs are required to implement a circuit, and hence, more clock branches are required; on the other hand, the length of each clock branch increases as the LUT size increases. Overall, a three-input lookup table appears to be a good choice.

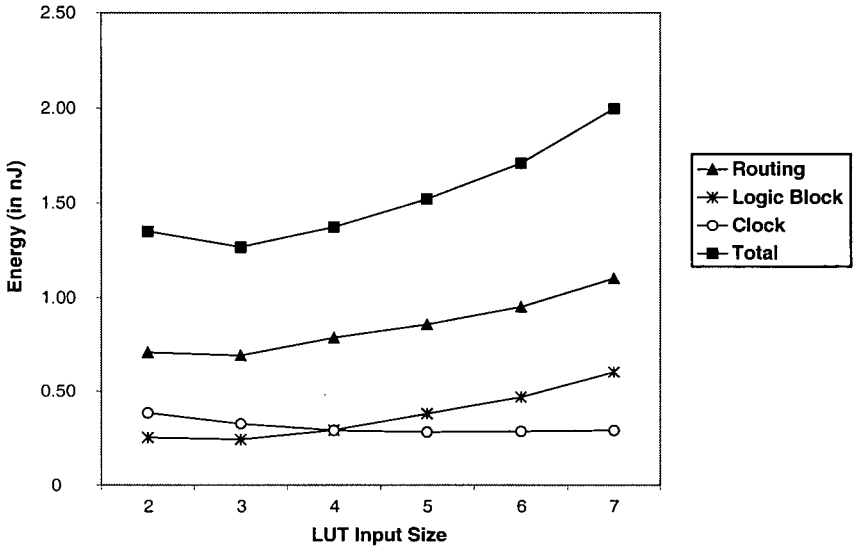


Figure 5.4 Energy versus look-up-table (LUT) size

5.2.3 Energy Distribution

This section analyzes how energy is distributed among the routing, logic blocks, and clock. The average value of the routing energy among the four types of switch blocks is used for comparison. The total energy of the logic blocks and routing wires has been separated into the dynamic portion (the dynamic and short-circuit energy) and the leakage portion for our analysis. As shown in Figure 5.5, most of the dynamic energy (58%) is dissipated in the routing fabric, while 18% is dissipated in the logic blocks and 19% is dissipated by the clock network. This finding is similar to the results obtained by previous researchers [25][37][55][59], except that these previous works did not consider the leakage energy. The leakage energy from logic blocks and routing wires is approximately 5% of the total power dissipation.

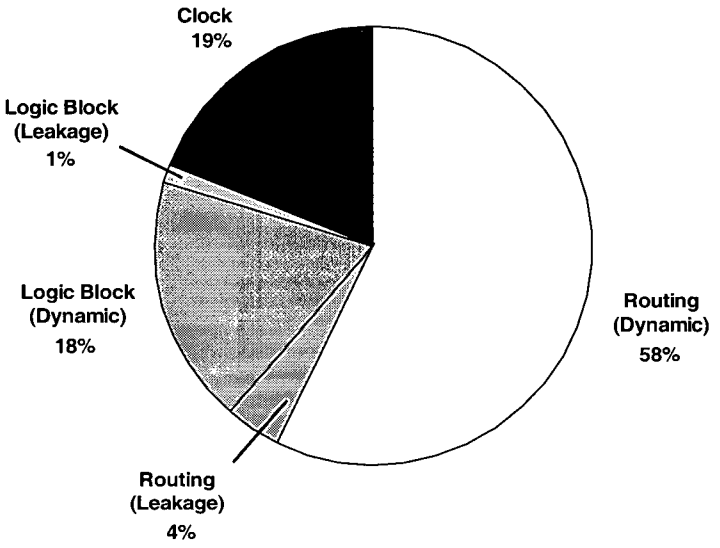


Figure 5.5 Energy distribution

5.3 Sensitivity of the Results

A previous study has demonstrated the importance of analyzing the sensitivity of architectural results to assumptions made during the experimentation [79]. For this research project, we have considered two criteria for the sensitivity analysis: *primary input transition density assumption* and *routing algorithm.*

5.3.1 Primary Input Transition Densities

The switching activities and densities of the primary inputs have a significant impact on the power estimation. Nevertheless, the primary input switching rate is often unknown at the design stage. Researchers normally assume normalized random signals (often with static probability of 0.5 and transition density of 0.5) for their primary inputs [53][49][50][51][59]. In fact, FPGA vendors suggest that the average switching rate of inputs typically ranges from 6% to 12% of the clock frequency, which can be represented by transition density from 0.12 to 0.24 [75][76]. We studied the effect of the transition density assumption on the following factors:

- Routing energy for different segment length
- Total energy dissipated within the circuit
- Our power analysis from previous experiments

5.3.1.1 Impact on Routing Energy

Figure 5.6 demonstrates that the routing energy increases with the primary input transition density. The curve is slightly parabolic because of the filter mechanism discussed in Chapter 4. If the transition densities propagated to the output of the LUTs are too high, the filter mechanism would eliminate the unrealistic glitches based on the inertial effect of the gates. This phenomenon happens more often when the primary input transition density is higher.

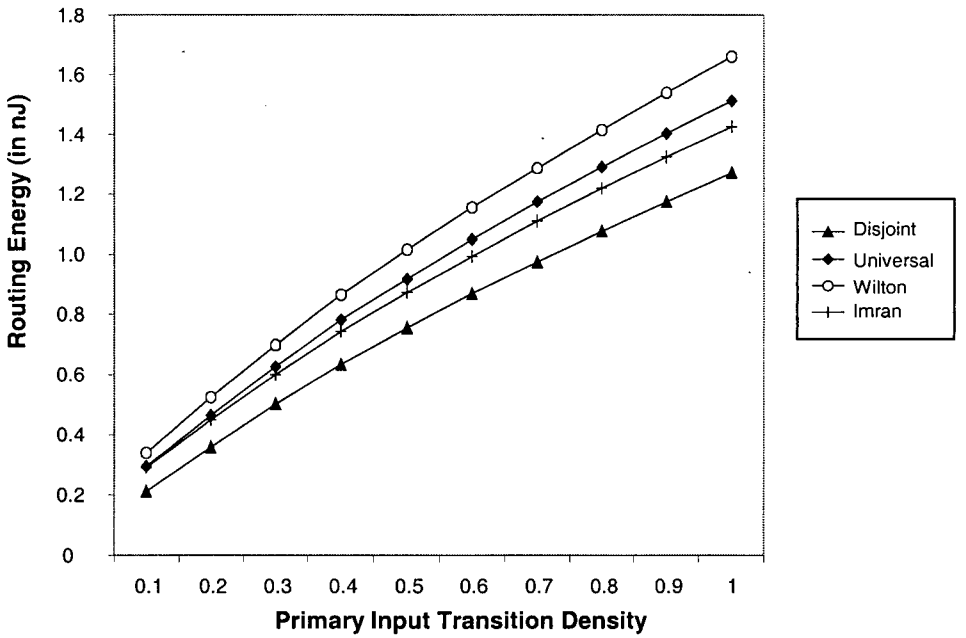


Figure 5.6 Routing energy versus primary input transition density

5.3.1.2 *Impact on Total Energy*

Figure 5.7 shows that the energy consumed by the routing and the logic blocks grows when a higher value of primary input transition density is used. The primary input transition density does not affect the clock energy since the clock network is separated from the segmented routing. Note that the primary input transition density has more effect on the routing energy than the logic block energy because the routing wires contribute more capacitance.

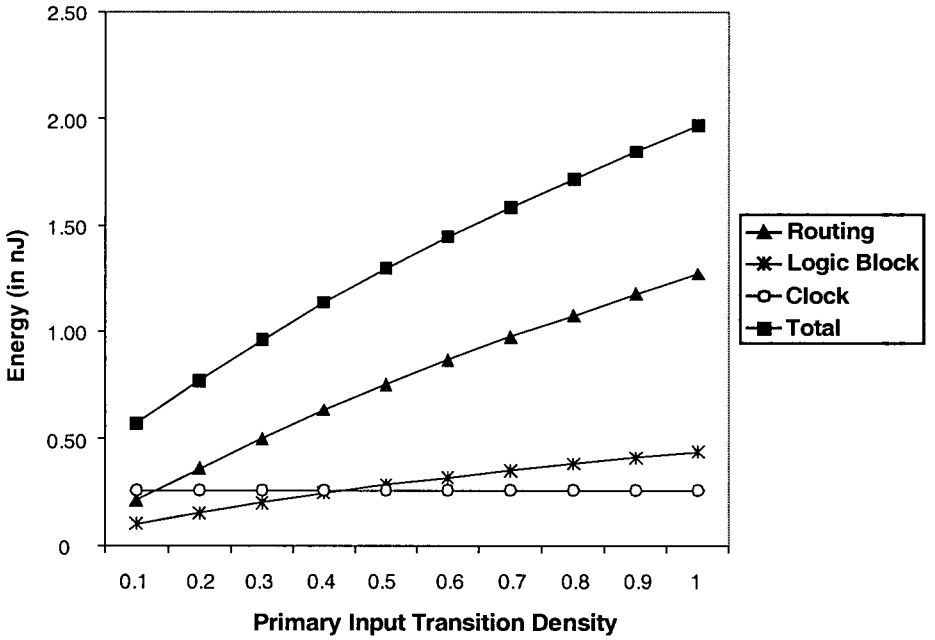


Figure 5.7 Energy versus primary input transition density

5.3.1.3 *Impact on Power Analysis*

All the experiments in Section 5.2 were repeated to investigate if our conclusions will change when a different primary input assumption is employed. To reflect the statistical values [75][76], the transition density for the primary inputs is set to 0.2 (10% of the clock frequency). Even though the overall energy dissipated by the routing and logic block has been dropped

approximately 40% compared to the previous results, the trends of the routing and logic block energy remain the same. Figures 5.8 to 5.10 show the results of our repeated experiments.

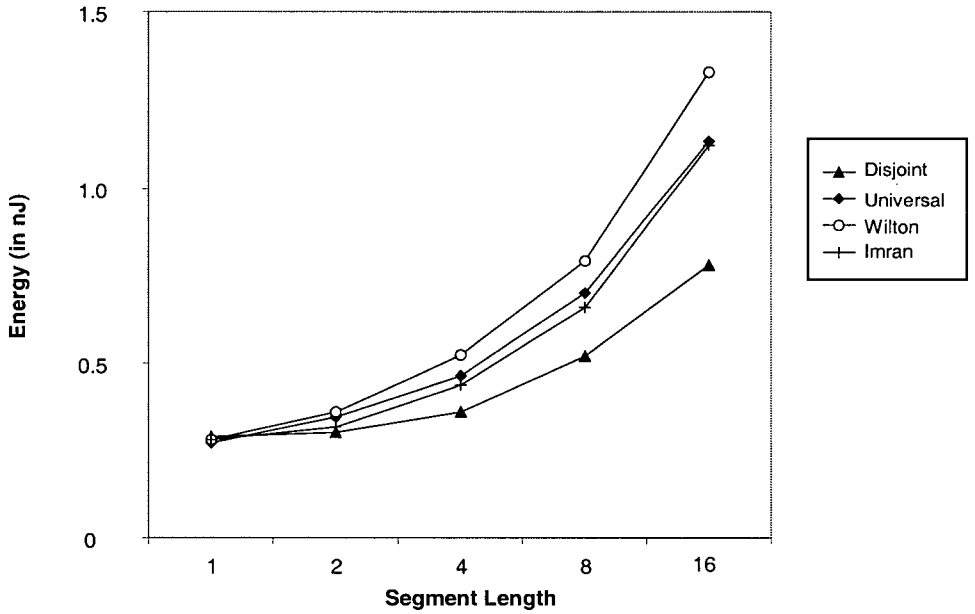


Figure 5.8 Energy versus segment length when input transition density is 0.2

Short wires are still preferable for lower energy dissipation in the FPGA interconnects (see Figure 5.8). Also as before, the Disjoint switch block topology dissipates more energy than the other topologies when used with length one segments, but dissipate less energy when the segment length is larger than one. Figure 5.9 also leads the same conclusions as before. The increase in the number of LUTs per logic block only slightly increases the energy dissipation by the logic block because of the additional internal interconnects. Nevertheless, fewer logic blocks are needed if the logic blocks are larger, resulting in fewer routing and clock branches. Note that the conclusion regarding the impact of LUT sizes on energy does change slightly (see Figure 5.10). The increase of logic block and routing is not as severe compared to the

experiment in Section 5.2.2, but the clock energy is the same as before. In this case, a four-input LUT seems to be a better choice than the three-input LUT identified before.

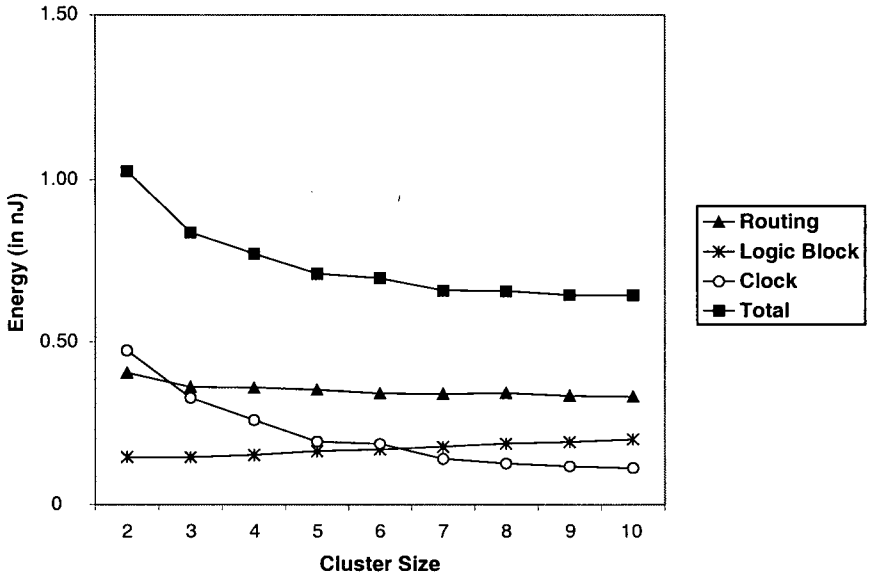


Figure 5.9 Energy versus cluster size when primary input transition density is 0.2

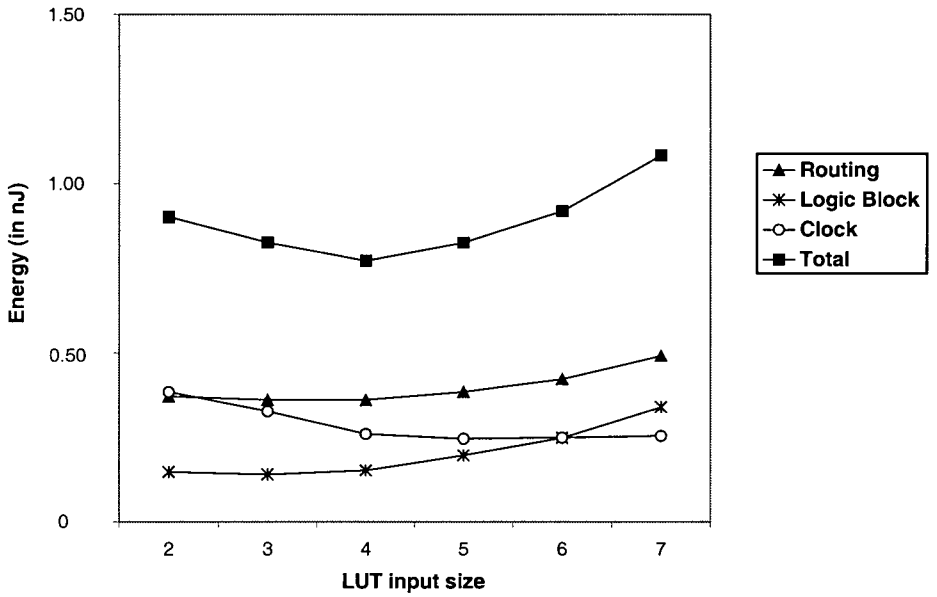


Figure 5.10 Energy versus look-up-table size when primary input transition density is 0.2

Finally, the energy distribution among the core components of FPGAs is revisited. The dynamic and short-circuit energy consumed by the routing and logic block is around 45% and 16% (see Figure 5.11). Because the clock network is a dedicated network, its energy consumption is unrelated to the primary input transition density. Moreover, the number of unused switches remains the same. Therefore, the leakage energy is the same as the previous attempt. Because of the reduction of energy dissipated by routing and logic blocks, the clock energy and leakage energy become more significant in this case. The percentage of clock energy and leakage energy is approximately 30% and 9% respectively.

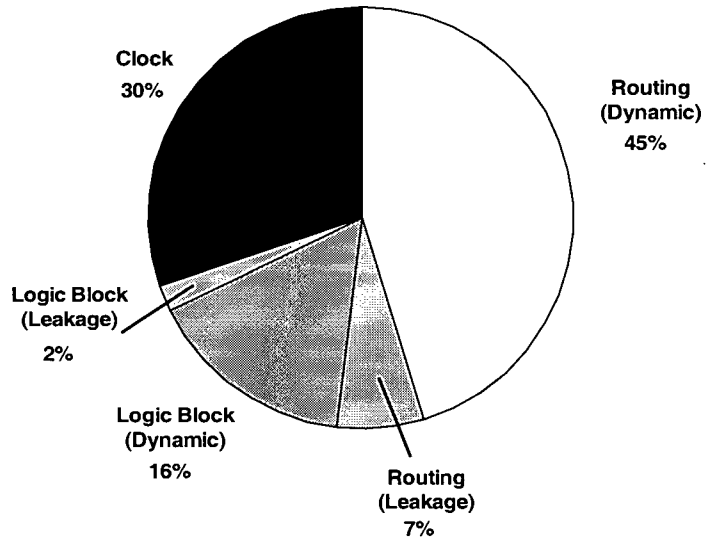


Figure 5.11 Energy distribution when primary input transition density is 0.2

5.3.2 Routing Algorithm

VPR supports two routing algorithms: *timing-driven* and *breath-first*. The timing-driven routing algorithm has been employed in Section 5.2. In this section, the breath-first routing algorithm is used. Compared to the timing-driven algorithm, the breath-first algorithm is capable of reducing the number of routing tracks required to route a circuit; therefore, it leads to a lower

routing capacitance. However, circuits generated by the breath-first algorithm are usually slower than those produced by the timing-driven algorithm [11]. As a large portion of the energy is dissipated in the routing, the routing algorithm may have strong influence on our results. Our objective here is to determine if our conclusion regarding the effect of segment length to energy consumption changes when a different routing algorithm is used. Also, we would like to know which algorithm is better for low-energy FPGAs.

The relationship between the routing energy and segment length obtained using the breath-first routing algorithm is very similar to that generated by its timing-driven counterpart (see Figure 5.12). Shorter segments are still more favorable for lower energy consumption within the FPGAs. Since the breath-first algorithm attempts to optimize the number of routing tracks, FPGAS employing the less-routable Disjoint topology consume more energy than FPGAs containing segments of length one. However, the Disjoint switch block remains a more suitable choice when the segment length is larger than one. Therefore, our conclusion has not been altered by the change of routing algorithm.

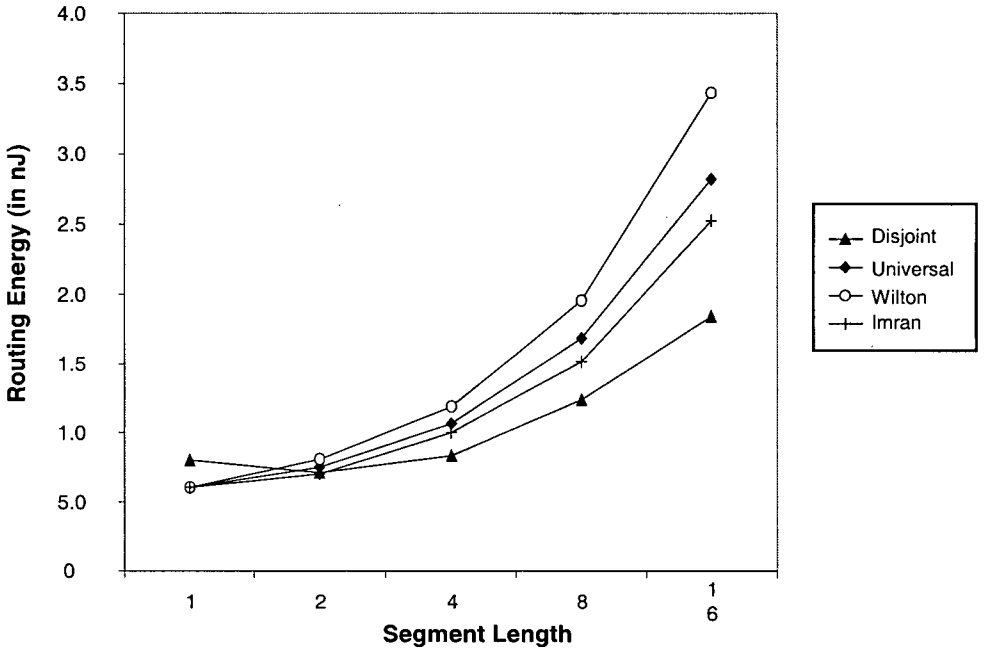


Figure 5.12 Routing energy versus segment length using breath-first algorithm

For a comparison between the timing-driven and breath-first algorithms, the average routing energy among the four types of switch block was calculated. The timing-driven algorithm yields circuits that consumed 2 times more power, but the average critical path delay of the circuits is 61% less than that using the breath-first algorithm. This is because the timing-driven router attempts to use fewer tracks for critical nets while the breath-first router attempts to minimize the total number of tracks within the FPGAs. Therefore, the critical nets routed by the breath-first router have to be mapped to longer routes since the number of tracks per channel in the FPGA is less than that using the timing-driven router. As a result, circuits obtained by timing-driven algorithm dissipated 22.2% less energy than those generated from the breath-first algorithm (see Figure 5.13). According to the results, the timing-driven algorithm is more suitable for low-energy FPGAs.

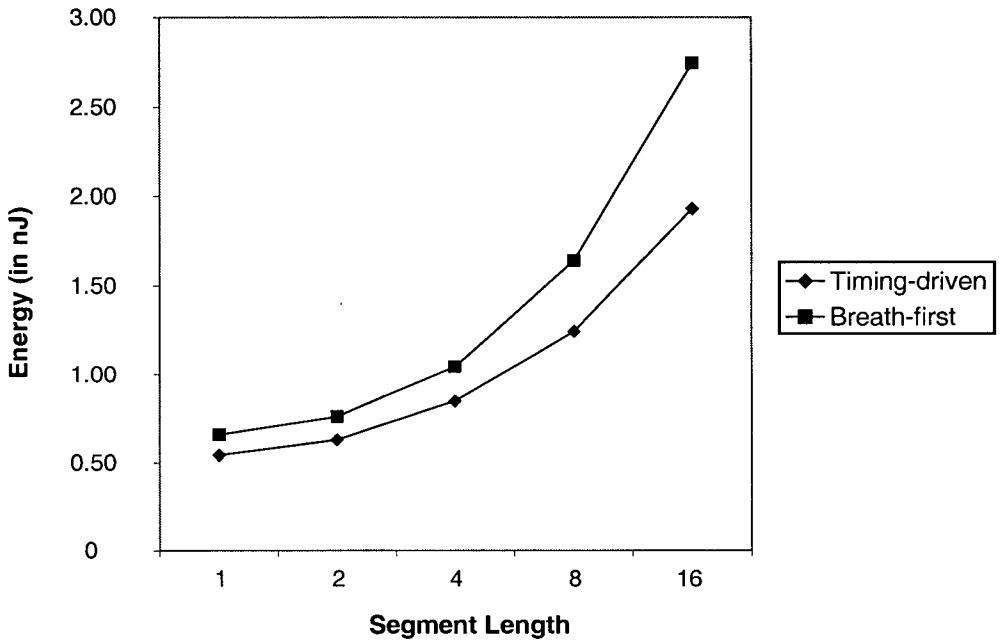


Figure 5.13 Average routing energy from timing-driven and breath-first algorithms

5.4 Summary

In this chapter, we have considered three architectural parameters: segment length, cluster size, and LUT size. Energy rather than power is used for comparison. According to the analysis, short wire segments are more suitable for low energy FPGAs. Moreover, the increase in cluster size and LUT size leads to a competition between two factors: larger logic blocks consume more power internally, but with larger logic blocks, fewer clock and routing branches are required.

Our results indicate that 58% of the energy is from the dynamic portion of the routing energy, 18% from the dynamic portion of the logic block energy, and 19% from the clock network. The leakage energy within the FPGA circuits is approximately 5% of the total energy consumption. The sensitivity of the underlying assumptions, including the primary input transition density and the routing algorithm, has been investigated. The result of the sensitivity

analysis further confirms the conclusions. It was also found that the timing-driven algorithm is better at optimum for low energy than the breath-first routing algorithm.

Chapter 6

6 CONCLUSIONS

This thesis has described a power model for FPGAs; the model has been incorporated to the existing VPR framework and provides a powerful tool to evaluate different architectural impacts on power dissipation within FPGAs.

Power dissipation has become a major concern in the semiconductor industry because it is closely related to the packaging cost, device reliability, and battery life (for handheld devices) [30]. Since FPGAs are programmed after fabrication to carry out a variety of applications, the power dissipated within the FPGAs must be determined dynamically. Gate-level power estimation techniques are suitable for determining the power dissipated within FPGAs because of their capability to handle large circuits. Chapter 2 examines several simulation and probabilistic power estimation techniques. Probabilistic techniques are preferred for this project since they are less computationally intensive. However, as processors get faster, simulation techniques may become more attractive because of their accuracy. Chapter 3 proposes modifications to the VPR framework and discusses a set of additional parameters for the architectural model used in VPR needed to estimate the power dissipated by internal connections within a logic block and by the clock network. Chapter 4 presents the details of the power model. The power model consists of two parts: activity generation and power estimation. The activity estimator generates the switching activity of each node in the circuit, and the power estimator calculates the power dissipated using the activity and architecture information. Among the existing probabilistic methods, the Transition Density Signal Model

was chosen for the activity generator. A previous study shows that the absolute error for the transition density model is within 23%. For the power estimation, equations have been defined and verified to calculate the dynamic, short-circuit, and leakage power for each basic component (such as LUT, multiplexer, flip-flop, routing wire, and buffer). The accuracy of the power estimates is within 15% of the HSPICE results.

To understand the power consumption within a FPGA, the power model has been used to evaluate different architectures by varying three parameters (segment length, cluster size, and LUT size) in Chapter 5. The results indicate that circuits dissipate less energy if routed using shorter wire segments. Larger cluster and LUT sizes yield larger logic blocks and lengthen each wire segment, but reduce the number of logic blocks required for a given circuit, and hence reduce the number of routing and clock branches. The increase of energy due to increased logic block size and the decrease in energy due to the smaller number of logic branches roughly balance out each other. As a result, varying the cluster and LUT sizes does not cause a significant effect on the total energy consumption. Sensitivity analysis was performed by repeating the experiments with different transition density values for the primary inputs and a different routing algorithm. Even through the total energy increases with the transition density values for the primary inputs, the results exhibit the same trends as before. On the other hand, it has been found that circuits routed by the breath-first routing algorithm consume more energy than those by the timing-driven routing algorithm used in the earlier part of the project. Nevertheless, the conclusions stay the same.

6.1 Future Work

The accuracy of power estimates is highly dependent on the input stimuli. Unfortunately, the existing benchmark circuits do not have documented input stimuli. Therefore, researchers normally assume random values for the primary inputs. In the future, benchmark circuits with input stimuli are needed for realistic power estimations.

Modern commercial FPGAs contain clock management circuitry, which manages multiple clocks in one system, and allows part of the global clock network to be disabled in order to reduce the dynamic power dissipation. The current model considers only a simple single global clock network. It can be extended to handle multiple clock networks.

The dynamic and short-circuit power dissipated by the SRAM cells is assumed to be negligible in the current model. In fact, such power consumption relies on the design and distribution of the SRAM cells. Since there are thousands of SRAM cells on an FPGA, even a small amount leakage current in each SRAM cell can make a big difference to the total power consumption of the chip during operation. A careful study on the power dissipation in the SRAM cell will be useful.

More experiments are necessary to further investigate the impacts of the architectural parameters on the power consumption. For instance, it has been assumed that the FPGAs only contain wire segments with the same length in this project. But in reality, commercial FPGAs normally contain wire segments with different lengths. Short wires are used for local or direct connections to the neighbor logic blocks; long wires are used for nets spanning half or the entire chip. There can be new power-driven algorithms for handling different wire segment lengths.

6.2 Contributions of this Work

The work as made the following contributions:

1. A detailed power consumption model for FPGAs has been designed, and integrated into an existing CAD tool. The model is significantly faster than HSPICE, but gives estimates that are within 15% of HSPICE for a given activity assumption. An error 15% is significant and may mean that the model is not suitable for an FPGA user to estimate the power of his/her circuit, however, the model will be of use in the design of the FPGAs themselves, where comparisons between alternative architectures or alternative CAD algorithms are more important than absolute power measurements.

2. The model was validated using HSPICE simulations.

3. Experiments were performed to investigate the impact of segment length, cluster size, and LUT size on the power dissipation of FPGAs.

4. A sensitivity analysis was performed

Most importantly, this research project has furthered the FPGA community's understanding of the power dissipation within FPGAs. It has confirmed that interconnects are the key to reducing power dissipation, and has raised awareness of the leakage power caused by the large number of unused switches on today's FPGAs.

REFERENCES

[1] Actel, Design for Low Power in Actel Antifuse FPGAs, September 2002.

[2] Actel, Efficient Use of ProASIC Clock Trees, August 2001.

[3] A. Allan, D. Edenfeld, W. Joyner Jr, A. Khang, M. Rogers, Y. Zorian, "2001 Technology Roadmap for Semiconductors", Computer, Vol. 35, Issue 1 , Jan 2002, pp. 42 –53

[4] Altera, APEX 20K Programmable Logic Device Family Data Sheet, ver 4.1, September 2001.

[5] Altera, CPLDs vs. FPGAs: Comparing High-Capacity Programmable Logic, February 1995.

[6] Altera, Stratix Programmable Logic Device Family Data Sheet, ver. 2.0, April 2002.

[7] Altera, Evaluating Power for Altera Devices, ver. 3, May 1999.

[8] Altera, Altera Device Package Information Datasheet, ver. 10.2, April 2002.

[9] Altera, Power Analysis in the Quartus II Development Tool, Technical Brief 72, ver. 1.0, January 2001.

[10] V.Betz, J.Rose, A, Marquardt, "Architecture and CAD For Deep-Submicron FPGAs," Kluwer Academic Publishers, 1999.

[11] V.Betz, VPR and T-VPack User's Manual, ver 4.30, March 2000.

[12] D. Brand, C. Visweswariah, "Inaccuracies in Power Estimation During Logic Synthesis," IEEE International Conference on Computer-Aided Design, pp. 388-394, November 1996.

[13] D. Brooks, V. Tiwari, M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," 27th Annual International Symposium on Computer Architecture, June 2000.

[14] S.D. Brown, R.J. Francis, J. Rose, Z.G. Vranesic, Field-Programmable Gate Arrays, Kluwer Academic Publishers, June 1992.

[15] R. Burch, F. Najm, P. Yang, T. Trick, "McPOWER: A Monte Carlo Approach to Power Estimation," IEEE/ACM International Conference on Computer-Aided Design, pp. 90-97, November 8-12, 1992.

[16] J.A. Butts, G.S. Sohi, "A Static Power Model for Architects," 33rd Symposium on Microarchitecture (MICRO-33), December 10-13, 2000.

[17] Y.W. Chang, D. Wong, and C.Wong, "Universal switch modules for FPGA design," ACM Transactions on Design Automation of Electronic Systems, vol. 1, pp 80-101, January 1996.

[18] J. Cong and Y.Ding, "Flowmap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, No. 1, pp.1-12, January 1994.

[19] J. Cowling, "Programmable Logic Trends in DSP Design," Electronic News, April 10, 2000.

[20] B. Dipert, "EDN's First Annual Programmable-Logic Directory," EDN, August 17, 2000.

[21] B. Dipert, "Programmable Logic: Beat the Heat on Power Consumption," EDN Access, August 1, 1997.

[22] D. Eckerbert, P. Larsson-Edefors, "Interconnect-Driven Short-Circuit Power Modeling," Proceedings of Euromicro Symposium on Digital Systems Design, pp. 412-421, September 4-6, 2001.

[23] E.G. Friedman, "Clock Distribution Design in VLSI Circuits – an Overview," Proceedings of IEEE International Symposium on Circuits and Systems, pp. 1475-1478, May 1993.

[24] V. George, H. Zhang, J. Rabaey, "The Design of a Low Energy FPGA," International Symposium on Low Power Electronics and Design, pp. 188-193, 1999.

[25] A. Garcia, W. Burleson, J.Danger, "Power Modelling in Field Programmable Gate Arrays (FPGAs)," in International Workshop on Field-Programmable Logic and Applications, pp. 274-281, September 1999.

[26] S. Gupta, F.N. Najm, "Analytic Models for RTL Power Estimation of Combinational and Sequential Circuits," IEEE Transactions on Computer-Aided Design, vol. 19, no. 7, pp. 808-814, July 2000.

[27] S. Gupta, F.N. Najm, "Power Macromodeling for High Level Power Estimation," 34th Design Automation Conference, pp. 365-370, June 9-13 1997.

[28] S. Gupta, F.N. Najm, "Power Modeling for High Level Power Estimation," IEEE Transactions on VLSI, vol. 8, no. 1, pp. 18-29, February 2000.

[29] J-M. Hwang, F-Y. Chiang, T-T. Hwang, "A Re-Engineering Approach to Low Power FPGA Design Using SPFD," Proceedings of Design Automation Conference, pp. 722-725, 1998.

[30] The International Technology Roadmap for Semiconductors, 2001 Edition, International Sematech, Austin, Texas, 2001.

[31] X. Jiang, S. Horiguchi, "Statistical Skew Modeling for General Clock Distribution Networks in Presence of Process Variations," IEEE Transactions on VLSI Systems, vol. 9, no. 5, pp. 704-717, October 2001.

[32] S.M. Kang, Y. Leblebici, CMOS Digital Integrated Circuits: Analysis and Design, McGraw-Hill, 1999.

[33] A. Leshavarzi, K.Roy, C.Hawkins, "Intrinsic Leakage in Low Power Deep Submicron CMOS ICs," in Proceedings of the IEEE International Test Conference, pp 146-155, 1997

[34] K.Poon, A.Yan, S.J.E. Wilton, "A Flexible Power Model for FPGAs," in International Workshop on Field-Programmable Logic and Applications, September 2002.

[35] J. Kozhaya, F.N. Najm, "Power Estimation for Large Sequential Circuits," IEEE Transactions on VLSI, vol. 9, no. 2, pp. 400-407, April 2001.

[36] R. Kumar, C.P. Ravikumar, "Leakage Power Estimation for Deep Submicron Circuits in an ASIC Design Environment," Proceedings of the 15th International Conference on VLSI Design (VLSID'02), January 2002.

[37] E.Kusse, J.Rabaey, "Low-Energy Embedded FPGA Structures," ACM International Symposium on Low Power Electronics and Design (ISLPED), August 1998.

[38] Lattice Semiconductor, Lattice ORCA Series 4 Data Sheet, April 2002.

[39] Lattice Semiconductor, ispMACH 4000B/C Power Consumption, TN1005, March 2002.

[40] Lattice Semiconductor, Power Estimation in ispMACH 5000B Devices, TN1023, May 2002.

[41] Lattice Semiconductor, Power Estimation in ispMACH 5000VG Devices, TN1002, November 2001.

[42] G.G. Lemieux and S.D. Brown, "A detailed router for allocation wire segments in field-programmable gate arrays," in Proceedings of the ACM Physical Design Workshop, April 1993.

[43] A. Lesea, M. Alexander, Xilinx, Powering Xilinx FPGAs, XAPP158, ver. 1.4, February 6, 2001.

[44] H. Li, W-K. Mak, S. Katkoori, "LUT-Based FPGA Technology Mapping for Power Minimization with Optimal Depth," IEEE Computer Society Workshop on VLSI, pp. 123-128, 2001.

[45] G. Lim, R. Saleh, "Trends in Low Power Digital Systems on Chip Design", in International Symposium on Quality of Electronic Design, March 2002.

[46] M.I. Masud, S.J.E. Wilton, "A New Switch Block for Segmented FPGAs," in International Workshop on Field-Programmable Logic and Applications, pp. 274-281, September 1999.

[47] Model Technology Corp., ModelUser, Q2, 2002.

[48] F.N. Najm, "Feedback, Correlation, and Delay Concerns in the Power Estimation of VLSI Circuits," ACM/IEEE Design Automation Conference, pp. 612-617, 1995.

[49] F.N. Najm, "Improved Estimation of the Switching Activity for Reliability Prediction in VLSI Circuits," IEEE Custom Integrated Circuits Conference, pp. 429-432, May 1-4, 1994.

[50] F.N. Najm, "Low-pass Filter for Computing the Transition Density in Digital Circuits," IEEE Transactions on Computer-Aided Design, vol. 13, no. 9, pp. 1123-1131, September 1994.

[51] F.N. Najm, "Power Estimation Techniques for Integrated Circuits," IEEE/ACM Internation Conference on Computer-Aided Design, 1995.

[52] F.N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," IEEE Transactions on VLSI Systems, vol. 2, no. 4, pp. 446-455, December 1994.

[53] F.N. Najm, "Transition Density, A New Measure of Activity in Digital Circuits," Texas Instruments Technical Report #7529/0032, August 1991.

[54] F.N. Najm, M.Y. Zhang, "Extreme Delay Sensitivity and the Worst-Case Switching Activity in VLSI Circuits," ACM/IEEE Design Automation Conference, pp. 623-627, 1995.

[55] T. Osmulski, et al., "A Probabilistic Power Prediction Tool for the Xilinx 4000-Series FPGA," Proceedings of International Workshop on Embedded/Distributed HPC Systems and Applications," pp 776-783, May 2000

[56] J.M. Rabaey, Digital Integrated Circuits: A Design Perspective, Prentice-Hall, 1996.

[57] K. Roy, "Power-Dissipation Driven FPGA Place and Route Under Timing Constraints," IEEE Transactions on Circuits and Systems, vol. 46, no. 5, pp. 634-637, May 1999.

[58] E.M. Sentovich, K.J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A.L. Sangiovanni-Vincentelli, "Sequential Circuit Design Using Synthesis and Optimization," ICCD, pp.328-333, 1992.

[59] L. Shang, A.S. Kaviani, K. Bathala, "Dynamic Power Consumption in Virtex-II FPGA Family," Tenth ACM International Symposium on Field-Programmable Gate Arrays, pp. 157-164, February 2002.

[60] L. Shang, N.K.Jha, "High-Level Power Modeling of CPLDs and FPGAs, " Proceedings of IEEE International Conference on Computer Design (ICCD), pp. 46-51, September 2001.

[61] A. Singh, M. Marek-Sadowska, "Efficient Circuit Clustering for Area and Power Reduction in FPGAs," Proceedings of International Symposium on Field-Programmable Gate Arrays, February 2002.

[62] E. Todorovich, M. Gilabert, G.sutter, S.Lopez-Buedo, E. Boemo, "A Tool for Activity Estimation in FPGAs," in International Workshop on Field-Programmable Logic and Applications, September 2002.

[63] K.Y. Toh, P.K. Ko, R.G. Meyer, "An Engineering Model for Short-Channel MOS Devices", IEEE Journal of Solid-State Circuits, Vol. 23, No. 4, August 1988.

[64] C.Y.Tsui, M. Pedram, A. M. Despain, "Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs," Proceedings of Design Automation Conference, pp. 18-23, 1994.

[65] T. Tuan, J. Rabaey, "Reconfigurable Fabric for Low-Energy Protocol Processing," ICASSP 2001.

[66] Q. Wang, S.B.K. Vrudhula, "A New Short Circuit Power Model for Complex CMOS Gates," Proceedings of IEEE Alessandro Volta Memorial Workshop on Low Power Design (Volta99), pp. 98-106, March 1999.

[67] Q. Wang, S.B.K. Vrudhula, "On Short Circuit Power Estimation of CMOS Inverters", Proceedings of IEEE International Conference on Computer Design (ICCD), pp. 70-75, October 1998.

[68] F. G. Wolff, M.J. Knieser, D.J. Weyer, C.A. Papachristou, "High-Level Low Power FPGA Design Methodology," National Aerospace and Electronics Conference (NAECON), pp.554-559.

[69] S. Wenande, R. Chidester, "Xilinx Takes Power Analysis to New Levels with Xpower," Xcell Journal, pp. 26-27, Fall/Winter 2001.

[70] S.J.E. Wilton, Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memory, PhD thesis, University of Toronto, 1997.

[71] S.J.E. Wilton, N.P. Jouppi, "CACTI: An Enhanced Cache Access and Cycle Time Model," IEEE Journal of Solid-State Circuits, vol. 31, no. 5, pp. 677-687, May 1996.

[72] Xilinx, Virtex-E 1.8V Field Programmable Gate Arrys Datasheet, ver 2.2, November 9, 2001.

[73] Xilinx, Obtaining Accurate Power Estimation for CoolRunner XPLA3 CPLD Using Xpower, XAPP360, ver. 1.0, August 15, 2001.

[74] Xilinx, "FPGAs, Power and Packages," XCell, 1997

[75] Xilinx, Understanding XC9500XL CPLD Power, ver. 1.1, January 22, 1999.

[76] Xilinx, Virtex Power Estimator User Guide, XAPP152, ver. 1.1, February 18, 2002.

[77] Xilinx, Virtex-II Pro Platform FPGAs: Functional Description, ver. 2.0, June 13, 2002.

[78] Xilinx, XC4000XLA/XV Field Programmable Gate Arrays Product Specification, DS015, ver. 1.3, October 18, 1999.

[79] A.Yan, R.Cheng, S.J.E. Wilton, "On the Sensitivity of FPGA Architectural Conclustions to Experimental Assumptions, Tools, and Techniques," in the proceeding of ACM International Symposium on Field-Programmable Gate Arrays, pp147-156, February 24-26, 2002

[80] G. Yeap, Practical Low Power Digital VLSI Design, Kluwer Academic Publishers, 1998.

Appendix A

FILTER MECHANISM CALCULATION FOR THE TRANSITION DENSITY MODEL

The transition density signal model is a probabilistic model to estimate power dissipation. The algorithm takes two pieces of information: the static probability, P(x), and transition density, D(x). Based on Equation (4.2), the transition density for the output of each logic module is computed. However, this approach does not restrict the maximum transition density, and assume short pulses to propagate through the logic gates. It does not capture the fact that the gates are not fast enough to respond to such short pulses. To account for the filtration effect of the circuit inertial delays, the model illustrated in Figure A.1 is introduced. Each logic module contains a low pass filter, and a zero-delay boolean block. A transition at *out'* is propagated to the *out* only if the signal is stable for longer than a delay (τ) [50].

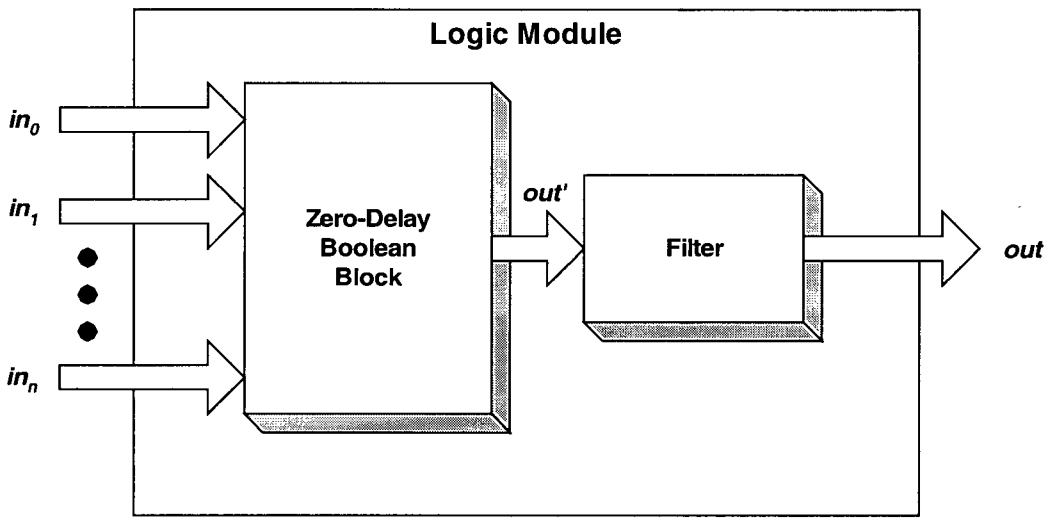


Figure A.1 Timing Model for Each Logic Module (from [50])

According to [50], the filter can be considered as a state machine with four states. Two states are defined as stable states when the input is either logic 0 or 1; the other two states are classified as unstable states when the input switch from logic 0 to 1, or from 1 to 0. If a transition happens, the filter will transit to an unstable state and wait for a time period of τ. If the input is in an unstable state and a transition occurs, no transition will occur at the output and the filter will return to a stable state. Based on the state machine assumption, the following formulas are derived for the static probability and the transition density at the output [50]:

$$D(y) = P_F \cdot D(x) \tag{A.1}$$

$$P(y) = P_F \cdot \left\{ \frac{1}{1 - F_0(\tau_0)} - P(x) + \left(E[r_1 - \tau_1 \mid r_1 > \tau_1] - E[r_0 - \tau_0 \mid r_0 > \tau_0] \right) \cdot \frac{D(x)}{2} \right\} \tag{A.2}$$

where:

- τ_0 and τ_1 can be set by the user, or derived from the propagation delay and rise/fall time of the module
- P(x) is the static probability of the input signals
- P_F is the *transmission probability* of the filter:

$$P_F = \frac{[1 - F_0(\tau_0)] \cdot [1 - F_1(\tau_1)]}{[1 - F_0(\tau_0) \cdot F_1(\tau_1)]} \tag{A.3}$$

- $F_0(\tau_0)$ and $F_1(\tau_1)$ are the *cumulative distribution functions*. $F_0(\tau_0)$ represents the fraction of the low pulses which are shorter than or equal to τ_0. $F_1(\tau_1)$ represents the fraction of the high pulses which are shorter than or equal to τ_1. These equations are derived on the basis of the following exponential distribution function, where μ is the mean pulse width:

$$F(z) = \int_0^t \frac{1}{\mu} e^{\frac{-z}{\mu}} dz = 1 - e^{\frac{-t}{\mu}} \tag{A.4}$$

As a result, the $F_0(\tau_0)$ and $F_1(\tau_1)$ are:

$$F_0(\tau_0) = 1 - e^{\frac{-\tau_0}{\mu_0}} \tag{A.5}$$

$$F_1(\tau_1) = 1 - e^{\frac{-\tau_1}{\mu_1}} \tag{A.6}$$

μ_0 and μ_1 represent the average low pulse width and the average high pulse width of the input signal respectively.

- For very small τ_0 and τ_1, the expected values $E[r_1-\tau_1 \mid r_1>\tau_1]$ and $E[r_0-\tau_0 \mid r_0-\tau_0]$ are:

$$E[r_0 - \tau_0 \mid r_0 > \tau_0] \approx \frac{(\mu_0 - \tau_0) + \tau_0 \cdot F_0(\tau_0)/2}{1 - F_0(\tau_0)} \tag{A.7}$$

$$E[r_1 - \tau_1 \mid r_1 > \tau_1] \approx \frac{(\mu_1 - \tau_1) + \tau_1 \cdot F_1(\tau_1)/2}{1 - F_1(\tau_1)} \tag{A.8}$$

Assume the gate has zero propagation delay and the rise/fall time is related to the clock period, τ_0 and τ_1 can be expressed as:

$$\tau_0 = \tau_1 = \beta\left(\frac{1}{f_{clk}}\right) \tag{A.9}$$

- β is the rise and fall time relative to the signal period
- f_{clk} represents the clock frequency

The average low and high pulse width of the signal, μ_0 and μ_1, are:

$$\mu_0 = \frac{2}{fclk \cdot D(x)} \cdot (1 - P(x)) \tag{A.10}$$

$$\mu_1 = \frac{2}{fclk \cdot D(x)} \cdot P(x) \tag{A.11}$$

Based on Equations (A.5) to (A.11) above, $E[r_0-\tau_0 \mid r_0>\tau_0]$ and $E[r_1-\tau_1 \mid r_1>\tau_1]$ are:

$$E[r_0-\tau_0 \mid r_0>\tau_0]=\frac{\mu_0-\tau_0\cdot\left(\frac{1+e^{\frac{-\tau_0}{\mu_0}}}{2}\right)}{e^{\frac{-\tau_0}{\mu_0}}}=\frac{1}{f_{CLK}}\left\{\frac{\beta-\frac{(1-P(x))}{D(x)}\cdot\left[1+e^{\frac{-\beta\cdot D(x)}{2\cdot(1-P(x))}}\right]}{e^{\frac{-\beta\cdot D(x)}{2\cdot(1-P(x))}}}\right\} \tag{A.12}$$

$$E[r_1-\tau_1 \mid r_1>\tau_1]=\frac{\mu_1-\tau_1\cdot\left(\frac{1+e^{\frac{-\tau_1}{\mu_1}}}{2}\right)}{e^{\frac{-\tau_1}{\mu_1}}}=\frac{1}{f_{CLK}}\left\{\frac{\beta-\frac{P(x)}{D(x)}\cdot\left[1+e^{\frac{-\beta\cdot D(x)}{2\cdot P(x)}}\right]}{e^{\frac{-\beta\cdot D(x)}{2\cdot P(x)}}}\right\} \tag{A.13}$$

According to the datasheets [4][72], the range of clock frequencies for commercial FPGAs is between 25Mhz and 200Mhz. Therefore, the clock period is around 5ns to 40ns, which is very small. As a result, the Equation (A.2) can be approximated as:

$$P(y)\approx P_F\cdot\left[\frac{1}{1-F_0(\tau_0)}-P(x)\right] \tag{A.14}$$

The following formulas for the static probability and transition density are derived from the above equations and used in the power model:

$$D(y) = P_F \cdot D(x) \tag{A.15}$$

$$P(y) \approx P_F \cdot \left[\frac{1}{e^{\frac{-\beta \cdot D(x)}{2 \cdot (1-P(x))}}} - P(x) \right] \tag{A.16}$$

where:

$$P_F = \frac{e^{\frac{-\beta \cdot D(x)}{2} \left[\frac{1}{P(x) \cdot (1-P(x))} \right]}}{e^{\frac{-\beta \cdot D(x)}{2P(x)}} + e^{\frac{-\beta \cdot D(x)}{2 \cdot (1-P(x))}} - e^{\frac{-\beta \cdot D(x)}{2} \left[\frac{1}{P(x) \cdot (1-P(x))} \right]}} \tag{A.17}$$