

DESIGN OF LOW POWER STATE MACHINES ON FPGAs

BY

M. Adil Saleem



2008-NUST-MS PhD-ComE-02

MSc-ComE-57

Submitted to the Department of Computer Engineering in fulfillment of the
requirements for the degree of

Masters of Science
In
Computer Engineering

THESIS SUPERVISOR

Dr. Shoab A. Khan

Department of Computer Engineering
College of Electrical and Mechanical Engineering,
National University of Sciences and Technology,
Rawalpindi

October 2010

ACKNOWLEDGMENTS

I express my humblest gratitude to ALLAH Almighty who has given me the direction and ability to accomplish the dissertation. I gratefully acknowledge the support, assistance and encouragement of all those people who have contributed towards the preparation of this thesis.

Special thanks to

- My thesis supervisor Dr. Shoab A Khan, for his guidance and support through the project
- The Head of Department Dr. Khalid Iqbal, for all his encouragement and motivation
- All my teachers specially Dr. Younus Javed, Dr. Khalid Iqbal, Dr. Shoab, Dr. Rashid Ahmed, Dr. Almas Anjum, Dr. Ghalib and Dr. Shaliza for their excellent teaching throughout the degree program.
- My class fellows Jawad, Nadeem, Usman, Obaid, Sir Zahid, Sir Jawed

And a very special Thank You to my Parents who have always been an inspiration for me.

10 October, 2010

ABSTRACT

With the massive growth and developments in digital systems, there is an increasing need of digital systems that consume lesser power. Low power applications have many potential target areas including multimedia applications, DSP applications, embedded controllers etc. Especially, mobile devices focus on low power consumption for longer battery life and usability. Methods and techniques have been developed on different levels for reducing power consumption in a digital circuit. The power consumed by a circuit can be divided into static power consumption and dynamic power consumption. Static power consumption is due to leakage currents in transistors and comes under the hardware domain. Whereas dynamic power consumption is due to switching of clock and other signals in the system and comes under the software/algorithmic domain. Clock gating is one good technique for reducing dynamic power consumption. In this technique we disable the clock signal to a circuit or a part of it when its functionality is not required, thus reducing dynamic power consumption. In this thesis, one such method based on clock gating is proposed. Traditionally, clock gating is a method applied in ASIC domain. Application of this technique in FPGA domain is relatively new. In this method we apply clock gating technique on a digital circuit at FSM level. For achieving better results, we have also leveraged the architectural benefits of the Virtex-5 FPGA device. So, this method is a combination of clock gating and a suitable device applied on FPGA based circuits. We tested this method on two circuits, a simple adder/multiplier circuit and more complex MD5 circuit. We achieved 5% and 7% reduction in dynamic power consumption respectively.

List of Publications

Adil Saleem, Shoab A Khan, “**Low Power State Machine Design on FPGAs**”.
Proceedings of 3rd International Conference on Advanced Computer Theory and
Engineering (ICACTE), Chengdu, China, August 2010, Vol. 4, pp. 442 – 445

Table of Contents

Chapter 1: INTRODUCTION	3
1.1 Background	3
1.2 Applications.....	4
1.3 Motivation	4
1.4 Scope of work.....	5
1.5 Structure	5
Chapter2: FPGA, FSM, Design/Implementation Cycle	6
2.1 FPGA (Field Programmable Gate Array).....	6
2.2 FSM (Finite State Machine)	9
2.3 Design Cycle of an FPGA Based Circuit	10
Chapter 3: Existing Techniques & Previous Work	13
3.1 Static versus Dynamic Power Consumption	13
3.2 Hardware Technology for Optimization.....	14
3.3 Software / Algorithm Based Techniques.....	16
3.4 Routing & Placing algorithms	21
3.5 Architecture Features for Power Optimization.....	21
3.6 Power modes	22
Chapter 4: Design / Implementation.....	23
4.1 Clock Gating.....	23
4.2 Synthesis of Circuit	27
4.3 Simulation and Power Analysis.....	28
Chapter 5: Results	31
5.1 Experiment Environment.....	31
5.2 Experiment Parameters.....	32
5.3 Circuits under Study	32
5.4 Discussion	36
Chapter 6: Conclusion & Future Work	38
6.1 Conclusion.....	38
6.2 Future work	39

APPENDIX A	40
APPENDIX B.....	42
REFERENCES.....	46

LIST OF FIGURES

Figure 2.1 Internal Architecture of an FPGA.....	5
Figure 2.2 FPGA Programming Process.....	6
Figure 2.3 A sample FSM.....	7
Figure 2.4 Example: Sequence Detector.....	8
Figure 2.5 Design Cycle of an FPGA based circuit.....	10
Figure 3.1 Dynamic Power Consumption.....	12
Figure 3.2 Comparison between Flash based FPGAs and SRAM based FPGAs.....	13
Figure 3.3 Clock Gating.....	15
Figure 3.4 State Splitting.....	15
Figure 3.5 State Splitting example.....	16
Figure 3.6 State merging.....	17
Figure 3.7 FSM Decomposition.....	18
Figure 3.8 Clock spine optimization.....	19
Figure 4.1 Major steps in our proposed method.....	21
Figure 4.2 Linear circuits.....	22
Figure 4.3 Linear circuit example.....	22
Figure 4.4 Branching circuit.....	23
Figure 4.5 Hybrid circuit.....	24
Figure 4.6 System Decomposition flow.....	25
Figure 4.7 Simulation flow of experiment.....	26
Figure 4.8 Power analysis flow of experiment.....	27
Figure 5.1 Arithmetic circuit.....	31
Figure 5.2 Clock gating.....	31
Figure 5.3 Arithmetic circuit result.....	32
Figure 5.4 Communication circuit.....	33
Figure 5.5 Communication circuit result.....	34
Figure A.1 Clock regions in Virtex-5 FPGA.....	38
Figure A.2 A clock region.....	39
Figure B.1 HDL code.....	40
Figure B.2 Generation of simulation file.....	41
Figure B.3 XPA input parameters.....	42
Figure B.4 XPA results.....	43

LIST OF TABLES

Table 5.1 Arithmetic Circuit Result.....	32
Table 5.2 Communication circuit result.....	33

LIST OF ABBREVIATIONS

1. FPGA – Field Programmable Gate Array
2. FSM – Finite State Machine
3. CAD – Computer Aided Design
4. EDA – Electronic Design Automation
5. ASIC – Application Specific Integrated Circuit
6. CPLD – Complex Programmable Logic Device
7. HDL – Hardware Description Language
8. XPA – Xpower Analyzer
9. DSP – Digital Signal Processing
10. CLB – Combinational Logic Block
11. IOB – Input Output Block
12. PLL – Phase Locked Loop
13. LUT – Look Up Table
14. HD – High Definition
15. SRAM – Static Random Access Memory
16. ROM – Read Only Memory
17. FEC – Forward Error Correction
18. ISE – Integrated Synthesis Environment

CHAPTER 1

INTRODUCTION

1.1 Background

An FPGA is a Field Programmable Gate Array. FPGA is a programmable device for implementing digital circuits. Different contemporary digital devices are IC (Integrated Circuit), Microcontroller, Microprocessor, DSP Processor, ASIC (Application Specific Integrated Circuit), CPLD (Complex Programmable Logic Device), FPGA. All these devices have their own characteristics.

CPLDs and FPGAs are of more interest for digital applications because we can create our custom hardware on these devices. These applications can range from a small circuit to very complex circuits. FPGAs are extensively used in applications in digital imaging, control applications, military equipment, space equipment, communications equipment, wireless equipment etc. FPGAs are used very extensively in virtually all the domains of digital systems. Circuits are implemented on FPGAs through an HDL (Hardware Description Language). Different HDLs are Verilog, VHDL, Abel, System C. Since applications are implemented on an FPGA through programming, so deployment time and debugging on an FPGA is very quick and easy. This is one of the reasons why FPGAs are so popular for digital applications.

While use of programmable devices is convenient, this convenience comes with a cost. The programmability of an FPGA chip is achieved through programmable SRAM cells in hardware. There are logic blocks and programmable interconnects to connect those blocks. Then there are channels to connect the I/Os of the chip to internal circuitry. All this surplus hardware leads to higher power consumption in FPGAs as compared to other devices.

With more and more devices becoming wireless and running on batteries, need for power efficient applications is very much there. A system may become useless if it fails to work within a given power budget. If a cell phone, a camera or any other portable device cannot work efficiently on its batteries, then its usability is seriously hampered.

Also, digital applications especially multimedia (imaging and video) applications and communications applications are getting more and more heavy in terms of clock speed requirements, memory requirements. [4] has worked specifically on the FSM on H264 codec.

With increasing need of powerful and processor hungry applications, we need power efficient methods for developing these applications on FPGAs. Power optimization, low power routing and placement, power aware applications are the areas on which lot of research is going on.

1.2 Applications

Some typical applications that require low power consumption are

DSP Systems DSP (Digital Signals Processing) systems typically consist of adders, multipliers, decimators, interpolators etc. They work on high data rates and clock speeds.

Multimedia (Encoding/Decoding) applications are typically CPU intensive and require more power. E.g Graphics cards, digital imaging devices etc.

Portable Devices need maximum battery life.

Going Green Technology companies are now conscious about environment. So naturally, low power consumption is always welcome on every level.

And many more

1.3 Motivation

With massive growth in development of portable devices, need for a *Greener* environment, attempts to put HD (High Definition) multimedia on portable phones, need for low power digital circuits is more than ever. Our motivation is to work in the area of low power circuits and to develop effective techniques and methods that can reduce power consumption in any digital circuit irrespective of its nature. Techniques that are independent of a particular kind of applications are more effective. This thesis is an endeavor in this direction.

1.4 Scope of work

Main focus of our work is to develop a method on FSM design level for reducing power consumption of digital circuits on FPGAs. FPGAs consume more power in general because of the programmable cells. Usually with a tight power budget, FPGA designs and applications require minimum possible power consumption. Hence, techniques for reducing power reduction are always an important part of digital designs. This work is one such attempt in this direction.

1.5 Structure

Chapter 2 describes some basic concepts about FPGAs, digital design, FSM. Chapter 3 discusses existing techniques of reducing power consumption in digital circuits. In Chapter 4 we have discussed the method and technique that we have applied. Chapter 5 presents the results of the experiments. Chapter 6 gives conclusion and future work in this direction. After the chapters, we have two appendices. Appendix A discusses the architecture of Virtex-5 FPGA. Appendix B consists of a small tutorial of XPA software by us for our readers.

CHAPTER 2

FPGA, FSM, Design/Implementation Cycle

In this chapter we will take a look at some of the basics about FPGA, FSM and the design cycle through which applications are mapped on FPGAs.

Programmable Devices are digital devices that can be configured (programmed) for a specific functionality fall into the category of Programmable Devices. Example of such devices are programmable memories (EEPROMs, Flash memories), CPLD, FPGA.

The primary advantage of programmable devices is that they make the deployment time for a design faster. Complex and large scale designs can be implemented on programmable devices very quickly with lesser cost. They greatly reduce the cost and time taken for debugging and experimentation at design time.

2.1 FPGA (Field Programmable Gate Array)

Field Programmable Gate Array (FPGA) is a programmable logic chip that consists of logic gates that can be programmed through some Hardware Description Language (HDL) to implement a digital circuit. It is called field programmable because an FPGA is 'Field Programmable' and not dependent on factory manufacturing for its configuration. This characteristic of FPGA has made it an excellent choice for all kinds of research work as well as industrial applications.

2.1.1 Architecture of FPGA

The internal architecture of FPGA is given in figure 2.1.

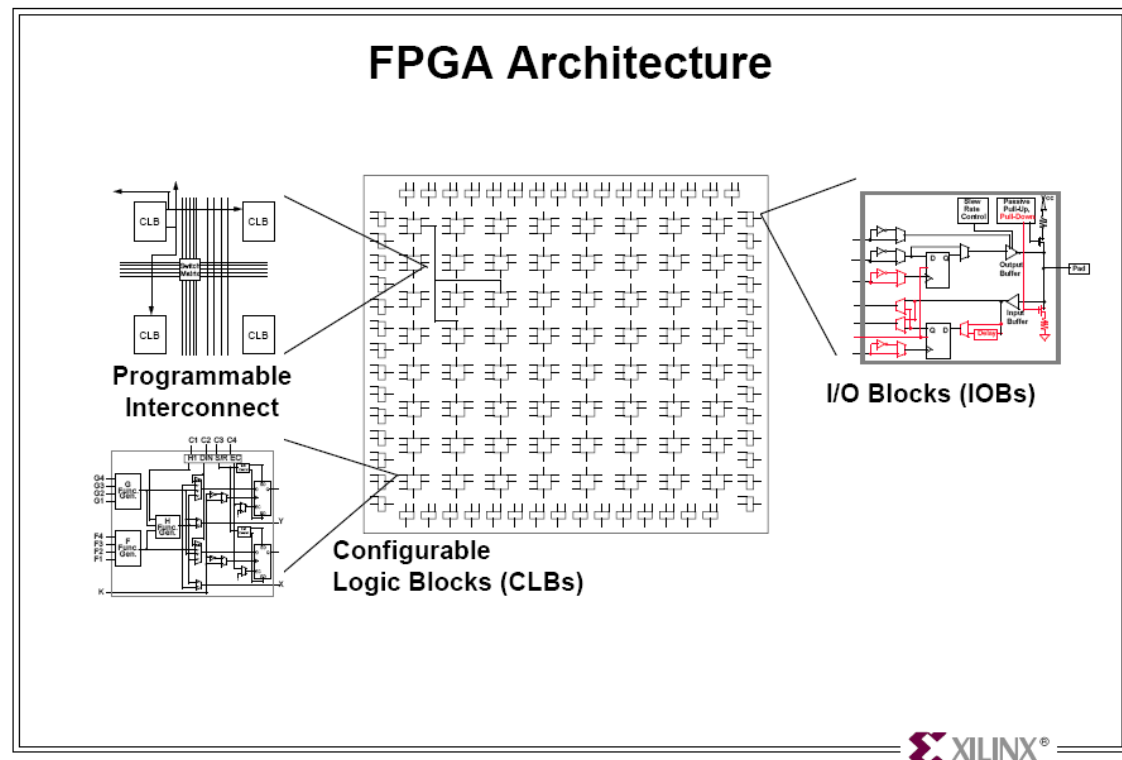


Fig.2.1 Internal Architecture of an FPGA. (source: [12])

The FPGA chip consists of the following.

CLB (Combinational Logic Block) CLBs are modules for implementing combinational logic. They are made up of LUTs (Look Up Table). They store the logic part of the circuit that is to be implemented.

IOB (Input Output Block) IOBs are modules for utilizing the I/Os of the FPGA. They serve as the interface for external I/O pins. They implement necessary circuitry to synchronize the input/output with external source.

Programmable Interconnects are the programmable switches used for routing and connecting the CLBs and IOBs.

Clock Manager & PLL (Phase Locked Loop) are responsible for clock signal generation and propagation in the chip with equal delay. Different clock frequencies can be generated using the clock manager with or without external clock signal.

Hard Cores & Coprocessors FPGAs can have built-in cores and co-processors that can be used in the digital designs.

The CLBs and other components are made up of SRAM cells for programmability. They can store values in them this way.

2.1.2 FPGA Programming Process

Programming an FPGA chip for a particular function/application is as follows.

1. Write code in some HDL (Hardware Description Language). Popular HDLs are Verilog, VHDL, System C, System Verilog.
2. After the FPGA design is created in HDL, **Synthesis** process is done on the HDL code. Synthesis converts the HDL blocks/modules into circuit primitives that are present in an FPGA.
3. After synthesis, **Place and Route** is the next step. In this step the component netlist that was generated by synthesis process is actually mapped to the CLBs, IOBs and other components. Basically placing and routing is the process of mapping components in the list to actual device. For this purpose some component library is provided by the Vendor with the FPGA.
4. After place and route, a binary file is generated. This is called programming file. This is downloaded on to the FPGA chip to make the FPGA work according to intended functionality. The downloading of file on chip can be done through one of the three methods
 - i. Through serial interface
 - ii. Through JTAG interface
 - iii. Stored on some ROM (Read Only Memory) and loaded onto the FPGA when it is powered on.

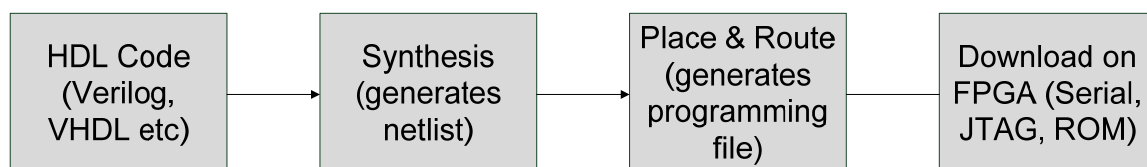


Fig.2.2 FPGA Programming Process

2.1.3 FPGA Vendors

FPGA vendor companies are Xilinx, Altera, Actel, Achronix, AMI Semiconductors and more. Among these Xilinx and Altera are the two leading FPGA manufacturers.

2.1.4 Software Tools

While synthesis tools and design tools are provided by FPGA vendors, 3rd party software are popular too. Software tools provided by Synplicity, Exemplar, Cadence, Synopsis and Mentor Graphics are popular.

2.2 FSM (Finite State Machine)

FSM (Finite State Machine) is a graphical method to represent state transitions within a digital system. FSM is used to model the signal flow of sequential circuits. FSM is a useful tool in the design process of a digital circuit.

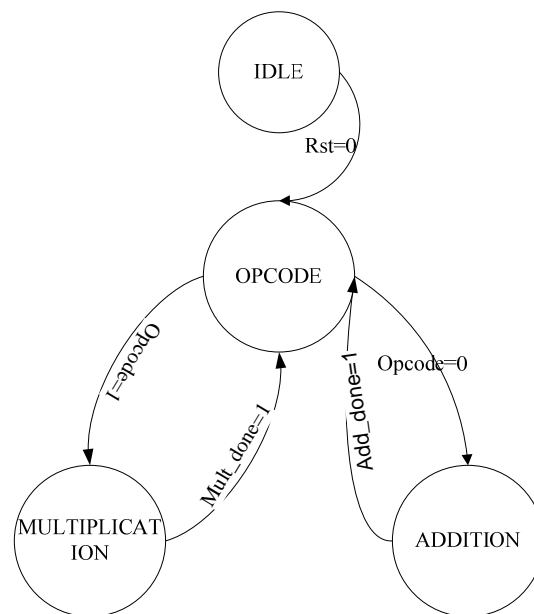


Fig.2.3 A sample FSM

An FSM consists of following components.

State A state is represented by a bubble. Values of the output that are linked with state are mentioned in the state.

Transition A transition is between two states or with the state itself. Transition is usually associated with the input signals. The value of input/output is mentioned with the state transition.

A **Mealy** machine's output at any given time is dependent on current state as well as the input signal values.

A **Moore** machine's output at any given time is dependent on current state only (e.g. Counters, pattern detectors etc).

The useful thing about FSM is that it is a direct mapping of states and state transitions. Each state in FSM represents a state in state register in actual circuit. Hence, size of the state register can be calculated from no. of states. Program logic can be directly inferred from it. Even HDL code can be generated from an FSM.

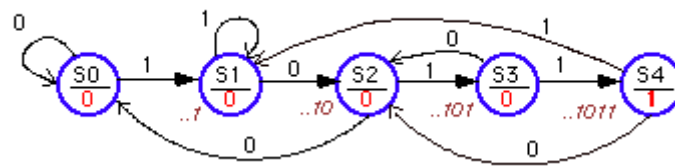


Fig.2.4 Example: Sequence Detector (source [11])

2.3 Design Cycle of an FPGA Based Circuit

The design cycle of an FPGA based circuit is as follows.

1. Requirement/need of a digital circuit
2. Modeling of the required application/functionality using FSM, pseudocode
3. Development of FSM, optimizations applied
4. HDL code generated based on the FSM
5. Functional simulation of the HDL code using test vectors/test bench
6. Synthesis of the HDL code
7. Timing simulation for checking timing constraints
8. Place & Route (device mapping)
9. Programming on FPGA chip
10. Running the circuit physically through FPGA

These are some the basic steps involved in the process. Of course, debugging and optimization is a constant part of the whole design flow. Optimization can be done at many levels. Optimization can be based on speed, power, area.

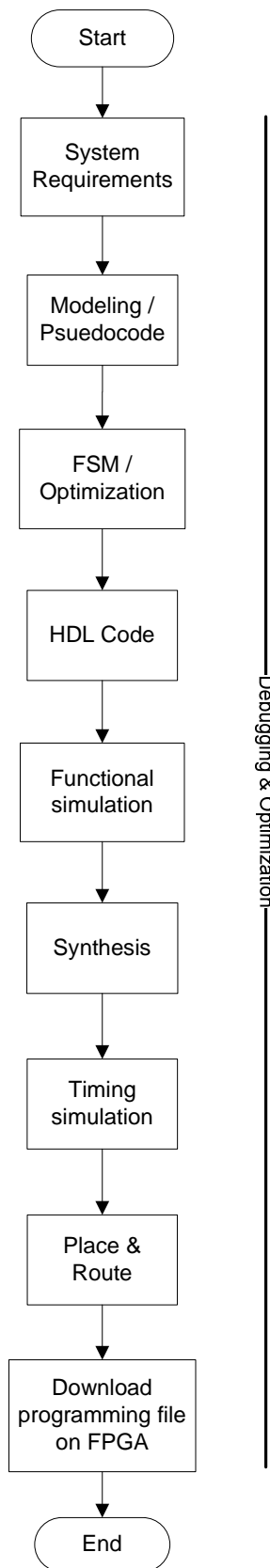


Fig.2.5 Design Cycle of an FPGA based circuit

CHAPTER 3

Existing Techniques & Previous Work

The problem of reducing power consumption is approached at many different levels. For example, at hardware level, system architecture level, software/algorithm level etc. Each type of power optimization has its own characteristics and limitations. They all can be applied singly or in combination to get best possible results.

However, from a digital designer's perspective it is easier and more flexible to adopt techniques at algorithmic level, since hardware and architecture level changes require more time and higher costs to implement even small changes. So naturally, much work has been done at the algorithmic level for designing low power systems. These techniques provide faster designing and deployment/debugging time.

Before we discuss the techniques for reducing power consumption we will discuss the concept of static and dynamic power. It will enable us to distinguish between hardware and software based approaches.

3.1 Static versus Dynamic Power Consumption

The total power consumption of a digital circuit can be divided into two parts.

1. Static power consumption
2. Dynamic power consumption

3.1.1 Static power consumption is the power consumed by the integrated circuits and the transistors. This is dependent on the hardware technology used (CMOS, TTL). It is also dependent on the manufacturing quality of the circuit. Static power consumption is always there and there is little we can do about it as digital designers.

Since static power consumption is in the hardware domain, so we will not focus on them. Instead, we will focus on the dynamic power consumption and techniques to reduce it.

3.1.2 Dynamic power consumption refers to the power consumption of a digital circuit due to switching of the logic signals. That includes the clock signals and other (data, control) signals in the circuit. So we can say that dynamic power consumption is directly linked to the logic of the circuit and how the signals make transitions during execution.

Actually, switching of a digital signal is switching of voltage levels. The transition between signal states ($0 \rightarrow 1$ or $1 \rightarrow 0$) consumes current and basis for power dissipation.

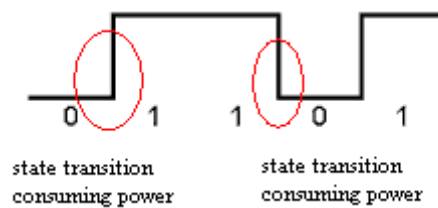


Fig.3.1 Dynamic Power Consumption

3.2 Hardware Technology for Optimization

The programmability of an FPGA chip is achieved through SRAM cells. The first and very basic level where we can reduce power consumption is the hardware itself. FPGA vendors are trying to work upon hardware technologies that consume lesser power.

One example is the technology of Flash FPGAs. These FPGAs are built by flash programmable cells. Figure 3.2 shows a comparison between flash FPGAs and SRAM based FPGAs.

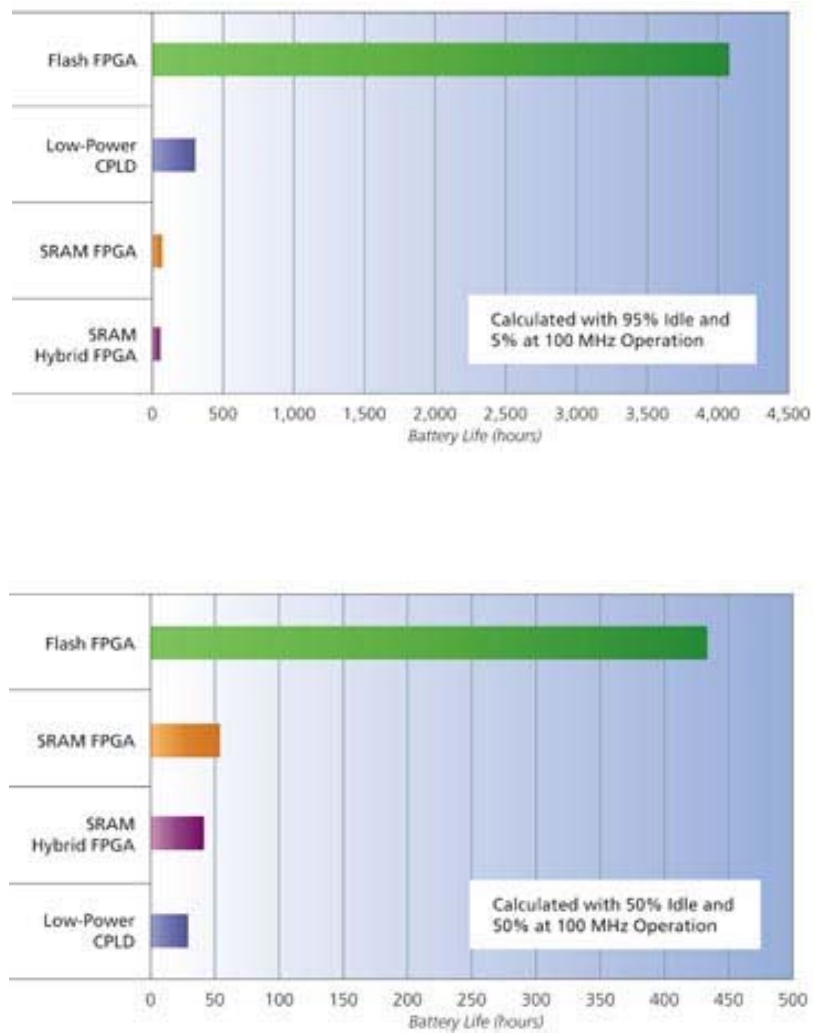


Fig.3.2 Comparison between Flash based FPGAs and SRAM based FPGAs (source: [13])

Some more hardware based schemes are discussed in detail in [8].

3.3 Software / Algorithm Based Techniques

This category is also referred to as EDA (Electronic Design Automation). The techniques in this category mainly focus on the circuit logic and signal flows rather than hardware technologies.

This implies that these techniques focus on state machine of the circuit. As discussed earlier, the FSM of the target application is the basis for all the signal flows, outputs, state transitions etc. For this very reason design of the state machine has a direct impact on the hardware produced as a result. Any changes made at the FSM level reflect directly in the resulting hardware. These techniques are being extensively worked upon.

Also note that the software based techniques apply on dynamic power reduction of a circuit. It is due to the fact that at application/algorithm level we can control the signal flow and hence the dynamic power consumption.

3.3.1 Clock Gating

Clocking gating is a simple but very effective technique to reduce power consumption of a sequential circuit. In this technique we cut off the clock signal to a circuit or a part of a circuit when its functioning is not required in the system.

Cutting off the clock signal does two things.

1. Reduce the signal switching of the clock signal itself
2. Reduce the signal switching activity in the circuit that runs on that particular clock

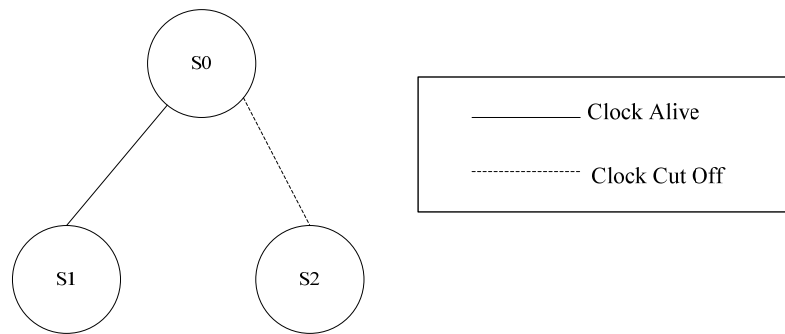


Fig.3.3 Clock Gating

Traditionally, clock gating technique is applied mainly in the ASIC domain. However, the technique is also valid in the FPGA domain. The reason it is not being extensively applied on FPGA circuits is that due to the programmable interconnects in the FPGA, clock gating technique is not that much effective as it is on ASIC circuits.

3.3.2 State Splitting

This technique is basically an extension of clock gating. When applying clock gating technique, we may come across a state which is a single transition state between multiple incoming and outgoing states.

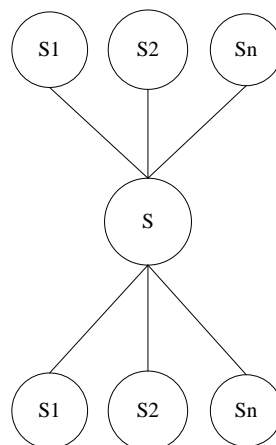


Fig.3.4 State Splitting

Since all the state transitions in that section of FSM are dependent on that single state, we cannot apply the clock gating technique directly. Because gating clock on that state means

disabling that state. Being a single state, that is not possible without disabling whole circuit. That is where concept of state splitting comes in.

State splitting means that we split a state into two or more states. The resulting states are different but functionally equivalent states. The idea is to create replica of the state so that we have redundant states and hence we can apply clock gating technique on it.

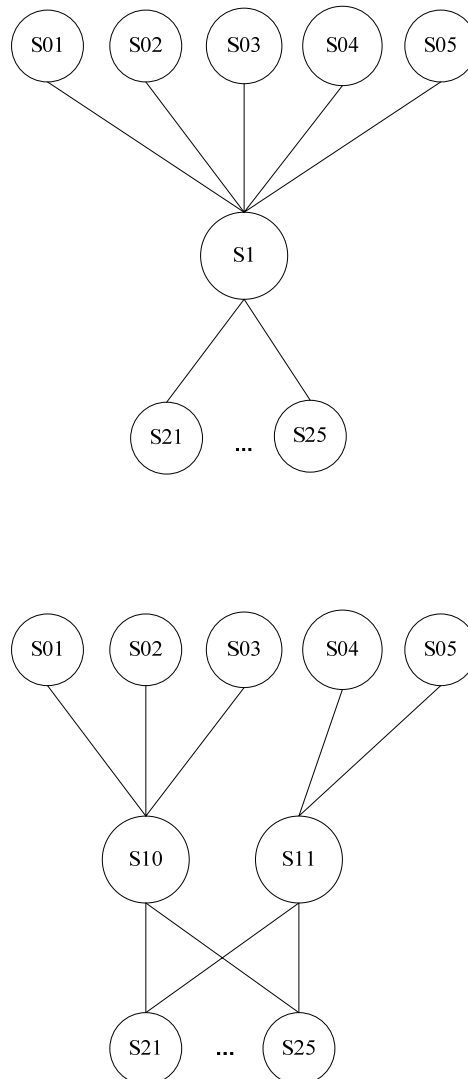


Fig.3.5 State Splitting example

For example, in figure 3.5 state S1 is a single state between multiple incoming and outgoing states. So to apply clock gating, we split S1 into two states S10 and S11.

The no. of states in which to split depends on the no. of incoming and outgoing states. There is not hard and fast rule about it. Depending upon the situation, it can vary. The state splitting method is discussed in detail in [2].

3.3.3 State Merging

On the contrary to state splitting, in this technique equivalent states are merged to form a single state. It reduces the size of state register and hence reduces power consumption. This technique is not applied with clock gating technique.

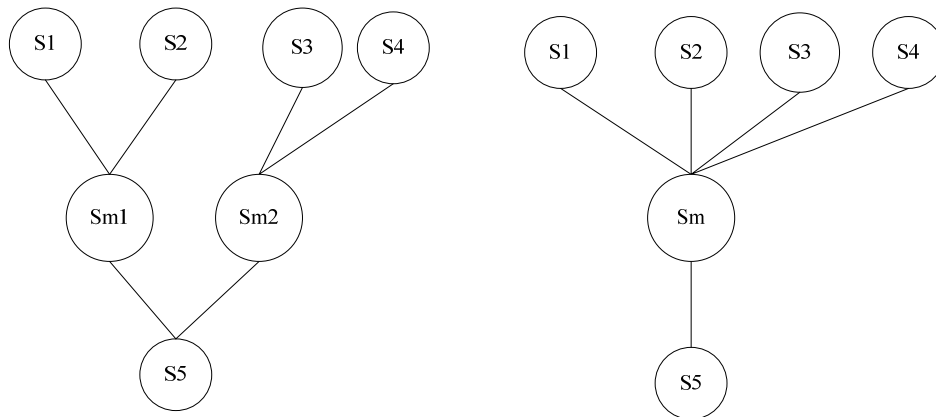


Fig.3.6 State merging

For example, in figure 3.6 Sm1 and Sm2 are merged to form a single state Sm.

3.3.4 State Assignment

At the time of synthesis of the circuit, states are assigned to the state registers. Switching activity can be reduced in the circuit by assigning states in such a way that the hamming distance (the no. of bits that are different between two binary numbers) is lowest between two states.

One such scheme is Grey coding where only 1 bit is changed between two successive states. The reduced switching activity leads to lower power consumption.

3.3.5 Don't Care Optimization

The don't care states are redundant in a circuit. They can be restructured to improve reduce the hardware and hence power consumption in a circuit.

3.3.6 FSM Decomposition

If the FSM of a circuit is large enough, we can divide it into sub-FSMs. The advantages that we can get out of it are that we apply different power optimization techniques on different sub-FSMs. Also, we can run different portions of circuit on different clocks.

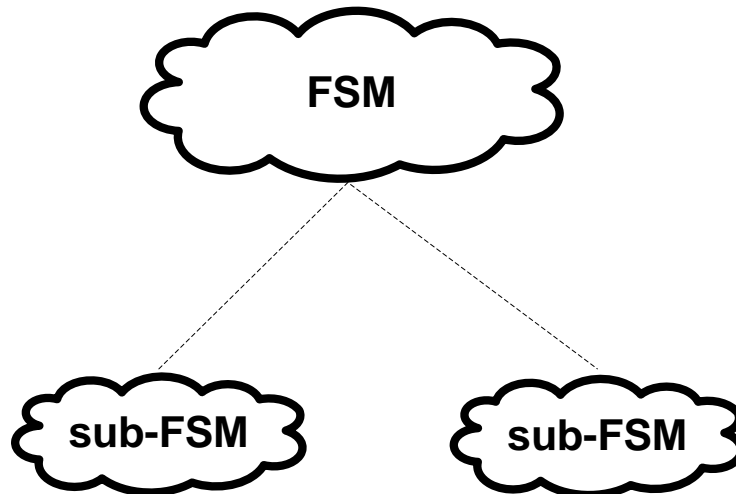


Fig.3.7 FSM Decomposition

3.3.7 Symbolic Synthesis

In [3] power optimization is applied at the time of synthesis of the circuits. It uses statistical techniques for the purpose. This method is specially good for large sized circuits.

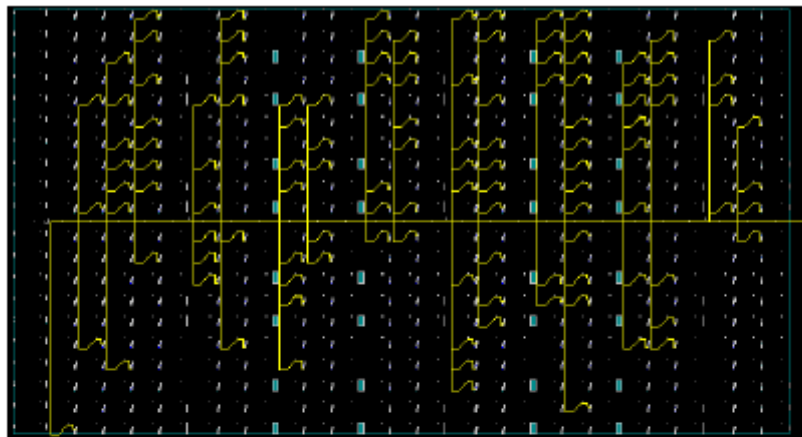
3.4 Routing & Placing algorithms

Special routing and placement algorithms are being developed by the synthesis software tools (e.g Xilinx ISE Design Suite). The Xilinx 11 and Xilinx 12 versions incorporate special placement algorithms that place the components so as to give minimum clock distances from the components to the clock.

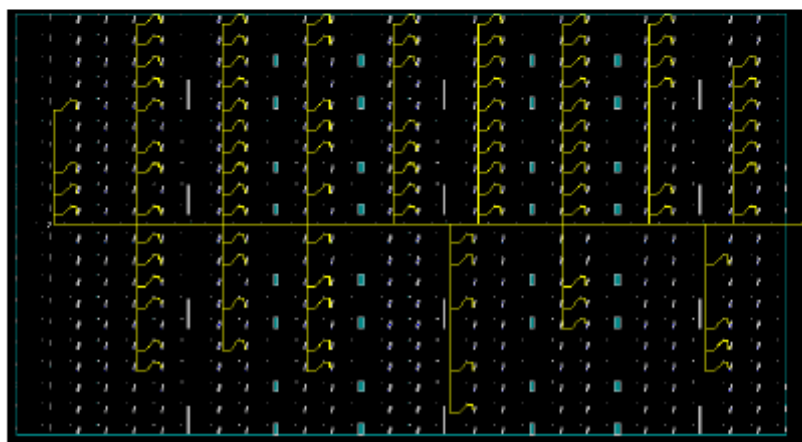
The routing and placing algorithms try to efficiently place the FFs so as to reduce the physical clock distances within the FPGA chip. [5] also discuss CAD algorithms for power efficient routing and placement.

3.5 Architecture Features for Power Optimization

Special FPGA internal architectures have been proposed and designed for the purpose. Example is the Virtex-5® device by Xilinx Inc. The internal architecture of the Virtex-5 device has been designed to allow minimum clock latencies, clock spine reduction and efficient clock distribution. The detailed features of the architecture can be found in [1] and [9] and the Appendix A.



a) Clock region placed without clock spine optimization



b) Clock region placed with clock spine optimization

Fig.3.8 Clock spine optimization (source: [1])

[6] and [7] have also discussed low power solutions on architectural level.

3.6 Power modes

One solution to a power optimization problem is to create different modes of operation for a circuit. In low power mode, it can apply various optimization techniques (like clock gating) described above. Also, it can define the degree to which quality (or accuracy) of the circuit can be compromised for low power consumption. There can be a quality-power tradeoff. For

example, for a particular algorithm, we can reduce no. of iterations for a particular operation.

Or we can limit the no. of bits (the precision) of the circuit.

- Lesser clock cycles
- Tradeoff b/w performance/power , accuracy/power

CHAPTER 4

Design / Implementation

Our work falls in the category of software/algorithm based techniques. The main idea is to apply the clock gating technique on a given circuit. Also we leveraged the architectural advantages that Virtex-5 device provides. So, in a nutshell we combine the clock gating technique with advantages of Virtex-5 FPGA device architecture.

The method that we applied consists of the following major steps.

1. Apply clock gating technique on the given circuit
2. Synthesize the circuit with target device as Virtex-5 FPGA
3. Simulation and Power Analysis

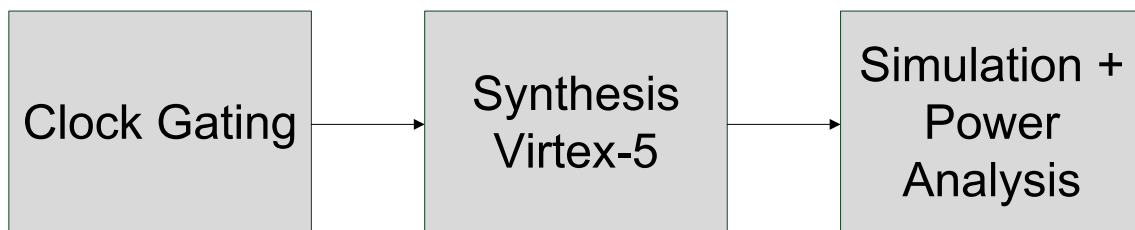


Fig.4.1 Major steps in our proposed method

4.1 Clock Gating

When applying clock gating on a circuit, we classify the circuit into 3 types of circuits. This classification is based on the pattern of state transitions in the FSM of that circuit.

1. Linear Circuits
2. Branching Circuits
3. Hybrid Circuits

4.1.1 Linear Circuits

We define linear circuits as circuits whose state transitions are in a linear fashion. Typically, signal processing systems and communication systems are linear circuits.

Signal processing systems usually operate on an input data at different stages. At each stage some algorithm is applied on the data fed from previous stage. So, in the FSM these stages appear in a linear fashion.

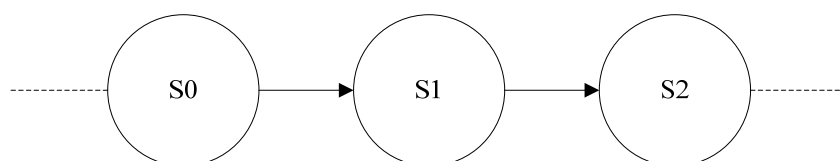


Fig.4.2 Linear circuits

Applying clock gating on a linear circuit is simple. While one stage is active, the clock can be cut off to other stages which are not required at that time, hence reducing clock switching and its consequent dynamic power consumption. For example, in figure 4.3 during the encryption process we can kill the clock to the other blocks like FEC (Forward Error Correction) and source coding.

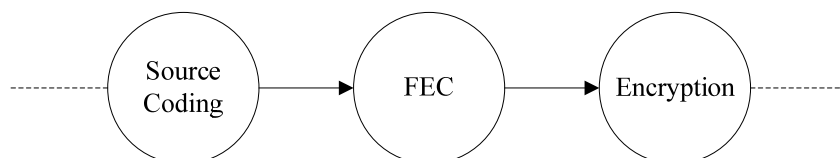


Fig.4.3 Linear circuit example

However, this approach has one disadvantage. Cutting off the clock in this manner prevent us from using pipelining technique in which all the units are functional at all times. Not using pipe-lining reduces the performance of the system. The trade-off between performance and power consumption can be made depending upon the system requirements.

4.1.2 Branching Circuits

Branching circuits are the kind of circuits whose states in the FSM are in the pattern of a spanning tree. Typically, encoders and decoders of different kinds fall in this category.

Examples are audio/video codecs, data compression algorithms, error correction codes etc. Clock gating when applied to such circuits comprises of two steps.

- a) Identifying a cluster/group of states that are independent of other branches in the system.
- b) Cut off the clock to that whole cluster/group when that branch is not active

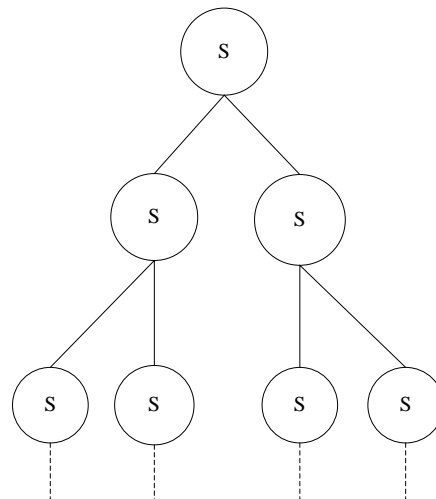


Fig.4.4 Branching circuit

In some cases, we may come across such a state where it is a single transition state between multiple incoming and outgoing states. In that case, we cannot apply this method directly. Then we will have to split that state using state splitting method and then apply this method. Depending on the density of incoming states, the state is split into two or more different but functionally equivalent states.

4.1.3 Hybrid Circuits

Sometimes we may come across larger and complex circuits which cannot be classified into one category (linear or branching). In such a case, we can divide the circuit into sub-circuits. And then apply clock gating accordingly on the sub-circuits.

No. of subsystems depends on the complexity of the system. Different subsystems can run on different clocks or they can run on the same clock. In case of different clocks, synchronization of clocks would also be required to ensure proper working.

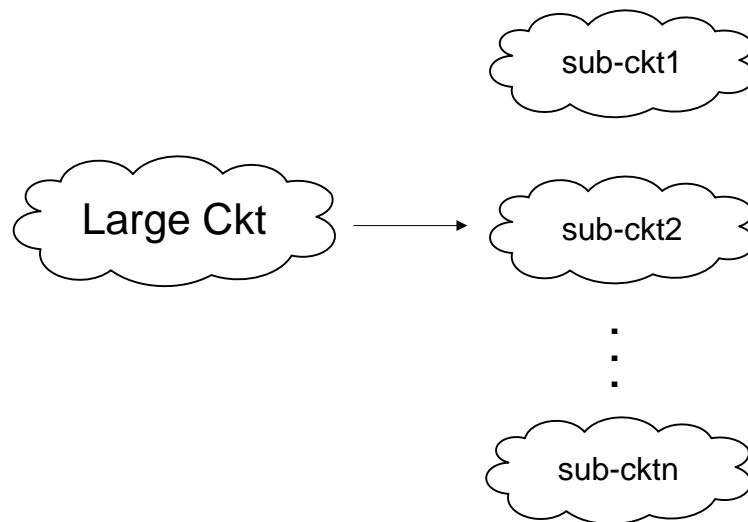


Fig.4.5 Hybrid circuit

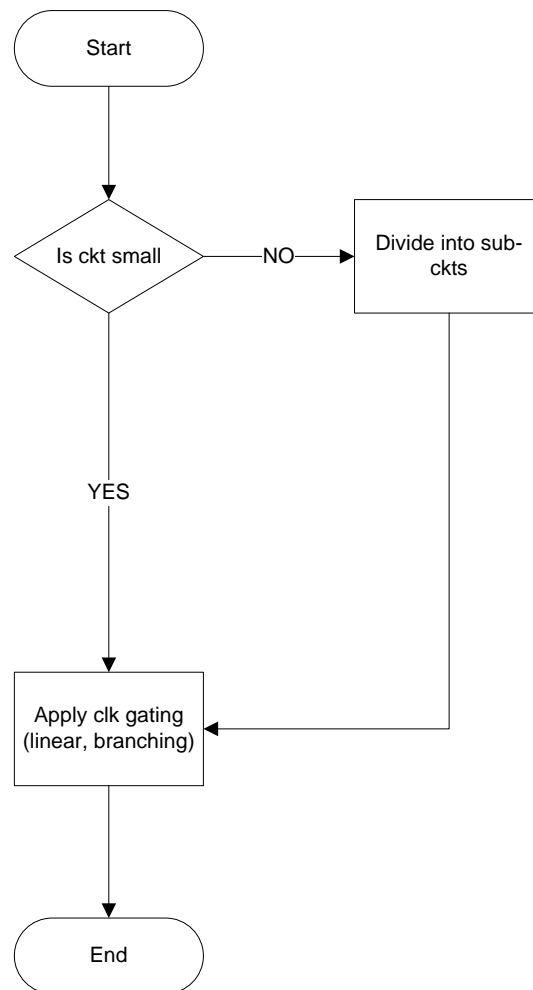


Fig.4.6 System Decomposition flow

4.2 Synthesis of Circuit

After applying clock gating in code, we synthesize the circuit. We used Xilinx ISE software for the purpose. The synthesized version of the circuit is necessary for power analysis at later stage.

Here, selection of the target device is important. As described earlier, we are getting the architectural advantages that Virtex-5 FPGA provides. The details of the Virtex-5 device are given in Appendix A.

4.3 Simulation and Power Analysis

After applying clock gating and synthesis of the circuit comes the experiments. The experiments consist of simulation of the circuits and then logic-level power estimation for comparison.

4.3.1 Simulation

1. For each circuit we considered, we took its Verilog code.
2. Developed and run the test bench module for that circuit. The simulations were run in ModelSim® software.
3. Developed the modified version for that circuit that implemented clock gating.
4. Again run the simulation with same test bench that was used for normal version of the circuit.
5. During each simulation, the output of the simulation (signal transitions, clock timings etc) were dumped in a simulation file. This file referred to as the simulation file was used later on for the power analysis part.

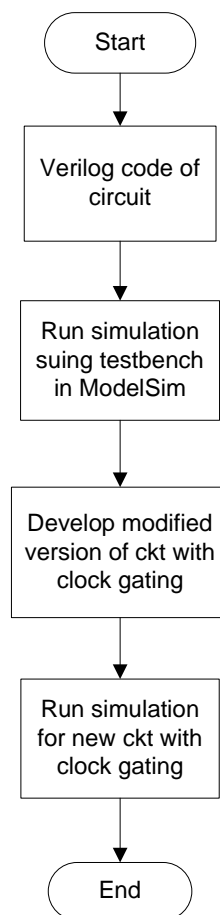


Fig.4.7 Simulation flow of experiment

4.3.2 Power Analysis

1. Create a new project in Xilinx ISE software for each version (simple and clock gating version both) for the circuit under study.
2. Selected Virtex-5 device as the target device. (xc5v1x30 specifically)
3. Synthesized both versions of the circuit.
4. Performed power analysis using XPower Analyzer (XPA) software. The XPA uses synthesized version of the circuit and the simulation file generated earlier for analysis.
5. Compare the results.

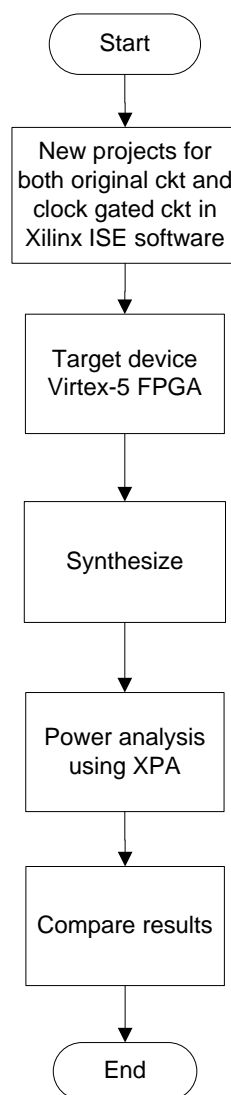


Fig.4.8 Power analysis flow of experiment

One important point to discuss here is that such kind of power analysis (using simulation files) is called Logic-Level Power Estimation. Power estimation is not as accurate as real power measurement (using board level or circuit level measurements). But the estimation is sufficient to serve our purpose. Since in clock gating we are working with logic levels and signals, so such an estimation is sufficient to give us an idea how much power are we reducing or increasing.

CHAPTER 5

RESULTS

In this chapter we will discuss the experiments that we performed and the results of those experiments.

5.1 Experiment Environment

Following are specifications of the software that we used for our simulations.

ModelSim SE 6.3f was used for simulation of verilog code. It is a very good simulation software. We can compile and import libraries from other vendors like Xilinx, Altera, Actel etc. for simulation of their devices.

Xilinx ISE 10.1 was used for synthesis of verilog code. Since we were using Xilinx Virtex-5 as the target device, so Xilinx ISE Design suite was a natural choice for us. Xilinx design suite consists of many good software tools that help in overall design and experimentation on FPGAs.

XPA (XPower Analyzer) 10.1 was used for power analysis. It uses the synthesized version of the circuit and the simulation file for power analysis.

Some of the **circuit cores** used for experiments were taken from [14]. The website is dedicated to open source cores. There are numerous good quality circuit cores available in almost all the mainstream hardware description languages.

5.2 Experiment Parameters

Since this method is aimed at reducing the power consumption of the circuit, so **Average Power Consumption** of the circuit is our main parameter. This parameter is being commonly used for such power related experiments. The average power consumed is in milli watts (mW).

5.3 Circuits under Study

We chose two circuits for evaluation of our proposed method. They are

- An arithmetic circuit (branching type circuit)
- A simple communication system (linear type circuit)

These circuits are conceptually strong candidates for evaluation of the method. Their relative simplicity makes it easier for us to focus on power analysis part rather than getting into complex details of the circuitry itself.

5.3.1 Circuit # 1 (Arithmetic Circuit)

Figure 5.1 shows the flow of arithmetic circuit. Normally the state is Opcode. When opcode 0 is selected, next state is addition and when opcode 1 is selected next state is multiplication. The multiplier and adder are both 4 input 8 bit circuits.

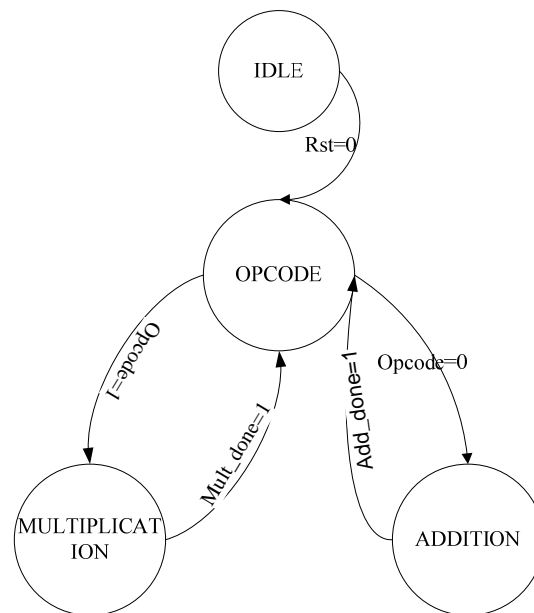


Fig.5.1 Arithmetic circuit

Now here is point where we apply clock gating.

- When multiplication state is active, we cutoff clock to addition circuit (the data registers).
- Similarly when addition is selected we disable clock to multiplication circuit's registers.
- And when we are at opcode state, we disable clock to both addition and multiplication.

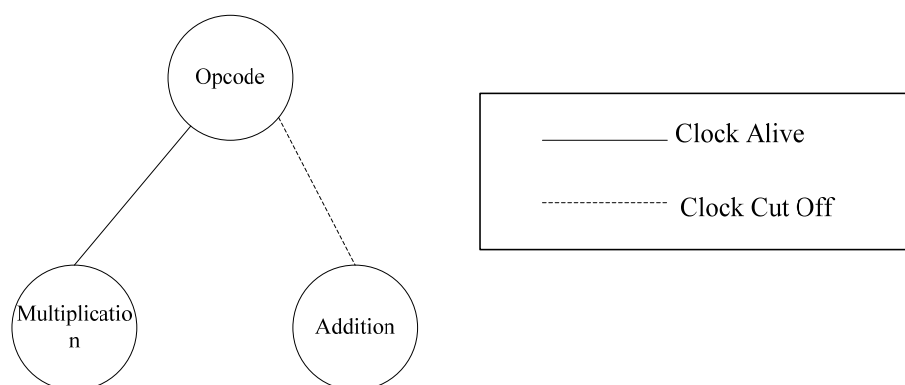


Fig.5.2 Clock gating

This cutting of clock when a state is not functioning reduces clock signal switching and the switching in data registers, thus reducing power consumption. The enabling and disabling of clock is achieved through two internal signals **ce_mult** and **ce_add**.

We run the simulation both with and without clock gating and compare the results after power estimation on XPA. The stimuli for both clock gated and non-gated circuits are the same. Each result is a mean of 3 runs.

The average power consumption in mW is as following.

Non - Clock Gated	Clock Gated	Improvement
336	320	5.00%

Table.5.1 Arithmetic circuit result

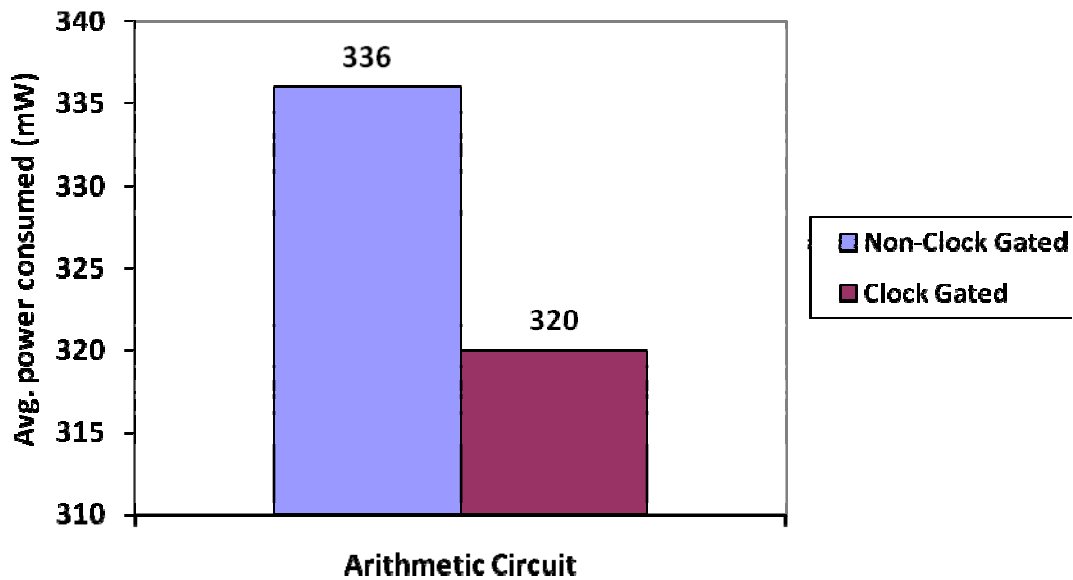


Fig.5.3 Arithmetic circuit result

5.3.2 Circuit # 2 (Simple Communication System)

The second circuit is actually a small part of a communication system. Figure 5.4 shows the flow of circuit. Typical communication transceivers are modeled like this, stage by stage processing of data/voice/video etc.

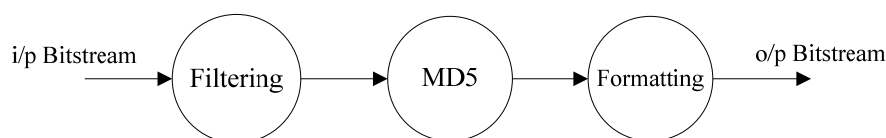


Fig.5.4 Communication circuit

Clock gating is applied on the MD5 module. When filtering or formatting is in progress, clock signal to MD5 module is disabled. It is disabled through a signal **ce_md5**.

It is worth noting here that we didn't have to go into the internal details of the MD5 algorithm here. The clock enabling/disabling is applied on the macro level. Of course we can apply the technique on a fine-grain level also, but even at this level it works. This is also important because it enables a digital designer to apply such optimizations without going into the details of individual algorithms.

We run the simulation both with and without clock gating and compare the results after power estimation on XPA. The stimuli for both clock gated and non-gated circuits are random. Each result is a mean of 3 runs.

The average power consumption in mW is as following.

Non - Clock Gated	Clock Gated	Improvement
337	314	7.32%

Table.5.2 Communication circuit result

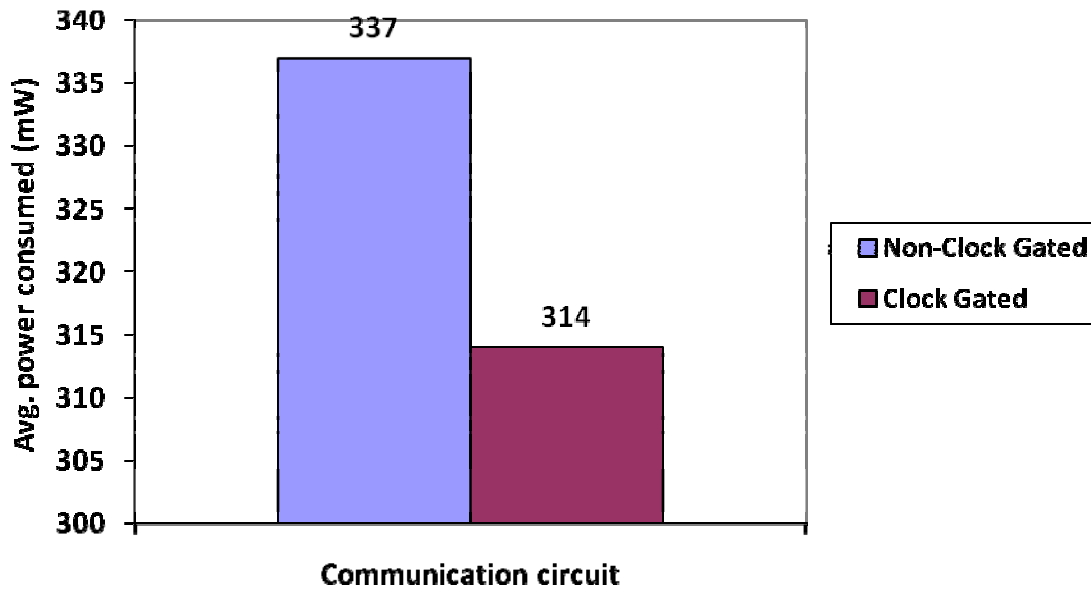


Fig.5.5 Communication circuit result

5.4 Discussion

The quantitative improvement power consumption is directly dependent on nature of application and the circuit. Since methods on signal flow level deal with signals and their manipulation, so their results can vary from case to case. In general, dynamic power reduction can be done upto a certain limit. After that either we have to compromise on the functionality or start optimizing on other levels.

In our case the experimental results have shown upto 7% dynamic power reduction using this method. This is a good trend since we have observed dynamic power reduction with no compromise on the functionality of the circuits.

This percentage can further improve if we apply clock gating hierarchically on multi-levels. With each level of detail more clock buffers and clock signals will spring out which can complicate the design process. So deciding the level of detail is a decision based on the target system and complexity that the designers want to maintain.

Another important aspect in the context of clock gating is the size of the circuit in terms of no. of registers and memory connected with it. Actually we just disable a clock signal to a specific part of the datapath. Actual saving of power is multiplied by the fact that how many

registers and other sequential circuit elements are driven on that particular clock. The more the count, more the power saving.

CHAPTER 6

Conclusion & Future Work

6.1 Conclusion

Low power applications are currently an active area of research among scientists and researchers because of obvious advantage that a low power consuming application provides. Low power consumption is a key consideration when working on digital systems. This requirement becomes a necessity in some cases. Possible areas of applications are multimedia applications, clock cycle intensive applications, systems for portable devices etc.

In this thesis, we proposed and experimented with a method for reducing power consumption in a digital circuit. We apply the clock gating on a given circuit. Two kinds of circuits are good candidates for clock gating. Linear circuits whose state transitions in the FSM are in a linear fashion. And branching circuits whose state transitions in the FSM are like a tree. Larger circuits can be divided into small sub-circuits like linear and branching circuits and then apply clock gating. Also, we exploited the architectural features of Virtex-5 FPGA for our experiments. Internal architecture of Virtex-5 FPGA is specifically designed to improve clock placement and routing. In summary, we combined clock gating with architectural features of Virtex-5 FPGA.

The method was applied on two circuits and after the power analysis of simple and gated versions of the circuit we saw up to 7% reduction in dynamic power of the given circuit. During experiments the functionality and precision of the circuit was not compromised. So applying this technique at signal level and without compromise on precision, 7% reduction in power consumption is a good improvement.

6.2 Future work

The next step in the direction of this work is to automate the process of applying the clock gating to an FSM. Currently, we did the analysis and re-structuring of FSM manually. This is why it was applied on relatively simpler circuits. Algorithms can be developed for parsing the state machine, look for potential areas of clock gating and modify the signal flow of control signals accordingly.

For branching circuits statistical methods can be used to apply clock gating technique according to the input data patterns. The input data patterns effect directly the branching within a circuit. Hence, an intelligent algorithm that takes into consideration the occurrence of certain input patterns can potentially give better results for the clock gating technique.

APPENDIX A

VIRTEX-5[®] FPGA

Virtex 5 FPGAs are high performance FPGAs by Xilinx Inc. They have upto 330,000 logic cells and can run upto 500 MHz clock speed [12]. Complete specifications of Virtex-5 FPGA can be seen in [9]

The main feature that distinguishes it from other FPGAs is its internal architecture. Due to its typical architecture it is more power efficient with a better clock distribution. [1] has described the features and their usage in detail.

Virtex-5 is divided internally into clock regions according to the clock distribution network. Clock distribution to each clock region can be controlled via 32 global clock buffers available.

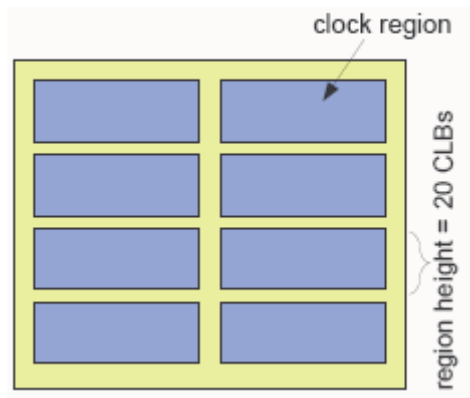


Fig. A.1 Clock regions in Virtex-5 FPGA

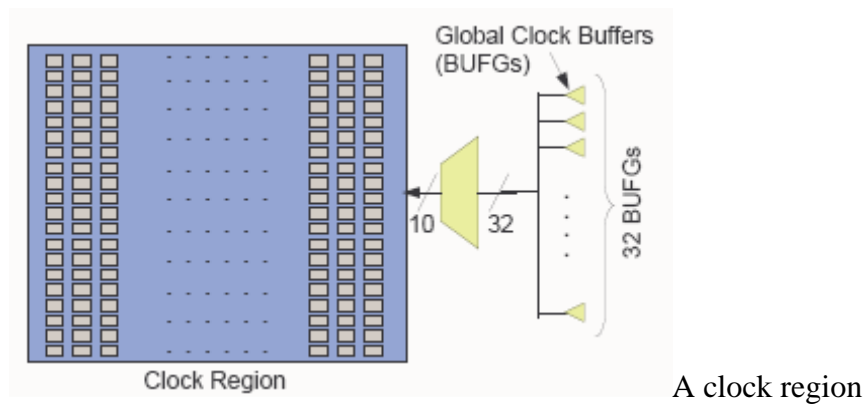


Fig. A.2 A clock region

The advantage that these clock regions provide is that clock signal can be enabled/ disabled to a whole clock region instead of a single flip flop. This is particularly useful in clock gating. When clock is gated (disabled), the switching activity of clock is stopped for whole clock region and the interconnects within that region. It results in overall lesser clock switching and hence lesser power consumption.

The clock distribution within clock region can further be optimized by reducing the length of clock spines. This kind of optimization is described in [1]. We will not discuss it further since it is beyond the scope of our work.

APPENDIX B

XPA (XPower Analyzer) Tutorial

XPA (XPower Analyzer) is a software program part of the Xilinx ISE Design Suite. It is a useful tool for quick analysis of logic level power estimation for digital circuits.

Following is a basic walkthrough of whole process of power estimation and analysis using XPA. More details can be found in [10]

HDL Code

```
36         takeMD5Hash = 'b1;
37
38         load_i = 'b0;
39         newtext_i= 'b0;
40         data_i=64'h0000806100000000;
41         @(posedge clk);
42         rst = #1 'b0;
43         @(posedge clk);
44         rst = #1 'b1;
45         @(posedge clk);
46         load_i = #1 'b1;
47         @(posedge clk);
48         load_i = #1 'b0;
49         data_i= 64'h0;
50         @(posedge clk);
51         load_i = #1 'b1;
52         @(posedge clk);
53         @(posedge clk);
54         load_i = #1 'b0;
55         data_i= 64'h8000000000000000;
56         @(posedge clk);
57         load_i =#1 'b1;
```

Fig. B.1 HDL code

First of all we take the HDL code of the circuit for which we want to do power analysis. Important thing to know is that we analyze power consumption of a ‘module’ (which represents a circuit in Verilog).

Test bench & Simulation

Then we write a test bench module for the circuit to perform simulation. Important thing in test bench is that it should dump its variables to produce VCD file. .VCD file is the file used by XPA. In test bench code, this file should be generated like this

```
initial
begin
$dumpfile("vcd_file_md5_gated.VCD"); ← Filename
$dumpvars(1, tb_top.uut); ← Module to analyze
end                                     (highlighted in red)
```

Fig. B.2 Generation of simulation file

- The VCD file should be generated by using \$dumpfile("filename.VCD")
- And the variables of the module that we want to analyze should be dumped in the above created VCD file using \$dumpvars(1,<modulename>.<instance_name>)

For example, in the above code we are analyzing module **tb_top** and particularly its instance **uut**

Then run the simulation in some HDL simulator (e.g. ModelSim)

Synthesis

After simulation, we synthesize the circuit using some synthesis software (e.g. Xilinx ISE). After synthesis, placing and routing of circuit we are done with synthesis part.

Analysis using XPA

- Open XPA software (under Xilinx ISE Design Suite 10.1 → ISE → Accessories → XPower Analyzer)
- In XPA, File → Open Design
- It will open following dialog box

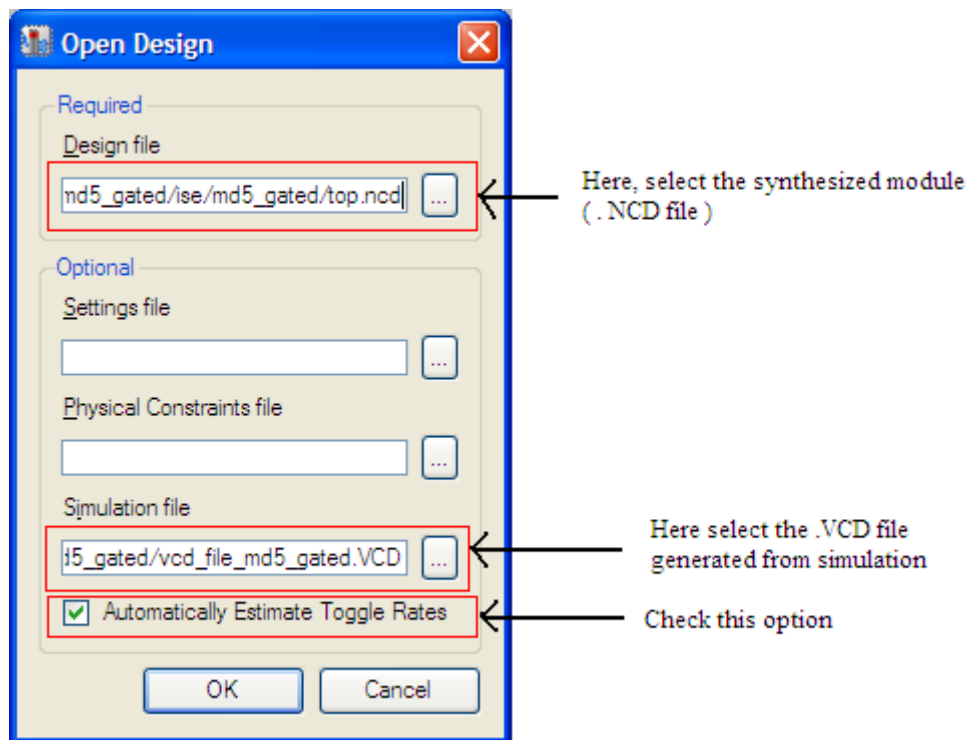
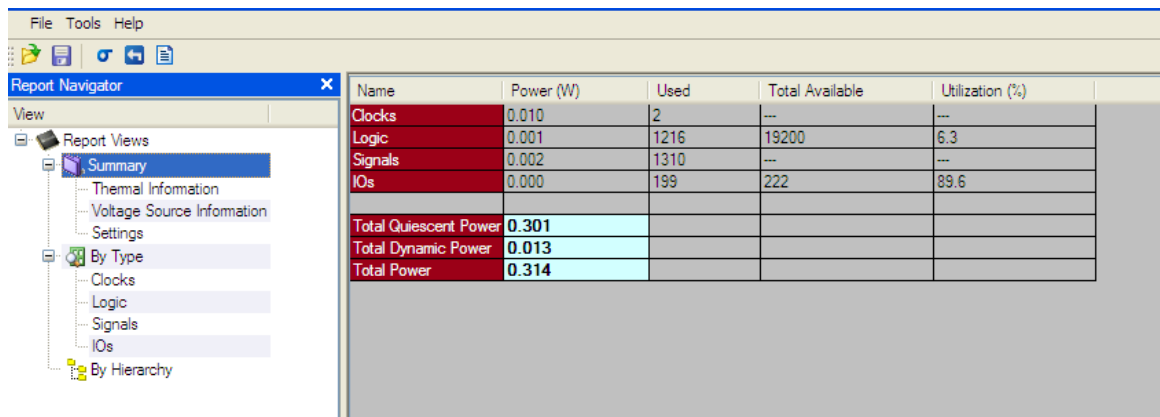


Fig. B.3 XPA input parameters

- Press OK. It will start process of power estimation and display the results



Name	Power (W)	Used	Total Available	Utilization (%)
Clocks	0.010	2	---	---
Logic	0.001	1216	19200	6.3
Signals	0.002	1310	---	---
IOs	0.000	199	222	89.6
Total Quiescent Power	0.301			
Total Dynamic Power	0.013			
Total Power	0.314			

Fig. B.4 XPA results

In the results we can see Total power, power by each signal type, thermal information.

REFERENCES

- [1] Qiang Wang, Subodh Gupta and Jason Anderson, “Clock power reduction for virtex-5 FPGAs”. ACM/SIGDA Int’l Symposium on Field Programmable Gate Arrays, pages 156–165, Monterey, CA, 2007.
- [2] Lin Yuan, Gang Qu, Tiziano Villa and Alberto Sangiovanni-Vincentelli, “An FSM reengineering approach to sequential circuit synthesis by state splitting”. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 27, No. 6, June 2008
- [3] L. Benini, P. Siegel, De Micheli, “Automatic synthesis of gated clocks for power reduction in sequential circuits.” IEEE Dcsirn and Test of Computers, pp. 32-40, fiec. 1994.
- [4] Ke Xu, Chiu-Sing Choy, Cheong-Fat Chan and Kong-Pang Pun, “Power-Efficient VLSI Realization of a Complex FSM for H.264/AVC Bitstream Parsing”. IEEE Transactions on Circuits and Systems - II: Express Briefs, Vol. 54, No. 11, November 2007.
- [5] Lamoureux, J. Wilton, S.J.E. “On the interaction between power-aware FPGA CAD Algorithms”. International Conference on Computer Aided Design, pages 701-708, ICCAD 2003.
- [6] Deming Chen Cong, J. Yiping Fan, “Low-power high-level synthesis for FPGA architectures”. Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003. ISLPED '03.
- [7] Siozios, K. Tatas, K. Soudris, D. Thanailakis, A. “Platform-based FPGA architecture: designing high-performance and low-power routing structure for realizing DSP applications”. 20th International Symposium on Parallel and Distributed Processing, IPDPS 2006.

- [8] Hichem Belhadj, Vishal Aggrawal, Ajay Pardhan and Amal Zerrouki, “Power-Aware FPGA Design”. Actel Corporation, White Paper, February 2009
- [9] Xilinx Inc., Virtex-5 FPGA Data Sheet, San Jose, CA. 2007.
- [10] Xilinx ISE design suite software manuals
- [11] Finite State Machine implemented as a Moore Machine. Available:
<http://www.seas.upenn.edu/~ese201/abel/abelMoore.html>
- [12] Getting started with FPGAs. Xilinx Inc. Available:
<http://www.xilinx.com/company/gettingstarted/index.htm>
- [13] Low Power Solutions. Actel Corporation. Available:
<http://www.actel.com/products/solutions/power/default.aspx>
- [14] OpenCores, <http://www.opencores.org>
- [15] FSM http://en.wikipedia.org/wiki/Finite_State_Machine