DESIGN AND SIMULATION OF LOW POWER FLIP FLOPS

PROJECT REPORT

Submitted by

V.NANDHINI

Register No: 13MAE10

in partial fulfillment for the requirement of award of the degree

of

MASTER OF ENGINEERING

in

APPLIED ELECTRONICS

Department of Electronics and Communication Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

(An autonomous institution affiliated to Anna University, Chennai)

COIMBATORE - 641 049

ANNA UNIVERSITY: CHENNAI 600 025

APRIL-2015

ACKNOWLEDGEMENT

First I would like to express my praise and gratitude to the Lord, who has showered his grace and blessing enabling me to complete this project in an excellent manner. He has made all things in beautiful in his time.

I express my sincere thanks to our beloved Joint Correspondent, **Shri. Shankar Vanavarayar** for his kind support and for providing necessary facilities to carry out the project work.

I would like to express my sincere thanks to our beloved Principal **Dr. R.S.Kumar M.E., Ph.D.,** who encouraged me with his valuable thoughts.

I would like to express my sincere thanks and deep sense of gratitude to our HOD, **Dr. Rajeswari Mariappan M.E., Ph.D.,** for her valuable suggestions and encouragement which paved way for the successful completion of the project.

I am greatly privileged to express my deep sense of gratitude to the Project Coordinator **Ms.Sasikala S, M.E.**, Associate Professor, Department of Electronics and Communication Engineering for her continuous support throughout the course.

In particular, I wish to thank and express my everlasting gratitude to the Supervisor **Prof.K.Ramprakash,** Department of Electronics and Communication Engineering**,** for his expert counselling in each and every steps of project work and I wish to convey my deep sense of gratitude to all teaching and non-teaching staff members of ECE Department for their help and cooperation.

Finally, I thank my parents and my family members for giving me the moral support in all of my activities and my dear friends who helped me to endure my difficult times with their unfailing support and warm wishes.

ABSTRACT

Power consumed by clocking has taken a major part of the whole design circuit. Given a design, we can reduce its power consumption by replacing several flip-flops with some multi-bit flip-flop. This may affect the performance of the original circuit because of its timing and placement capacity constraints. To overcome this problem efficiently, a technique combination table is built to enumerate possible combinations of flip-flops provided by a library. Finally, merging of flip-flops is done with help of co-ordination transformation and combination table. We can achieve better area reduction and power reduction by 37.65%. The implementation of flip flop merging is done in ModelSim software and the power analysis through Quartus II.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The circuits stored information about the previous history of inputs are called storage or memory elements. A primitive storage element can be constructed from a small number of gates connecting the outputs back as inputs. These circuits are binary cells capable of storing one bit of information. They have two outputs, one for the normal value and one for the complement value of bit stored in it. Primitive memory elements actually fall into two board classes:

- latches
- flip-flop

If a latch has only data inputs, it is called an unlocked latch (or only latch). Level-sensitive latches have an additional enable input, sometimes called the clock. Level-sensitive latches continuously sample their inputs when they are enabled. Any change in the level of the input is propagated through to the output. When the enable signal is unasserted, the last value of the inputs is determines the state held by the latch.

Flip-flops differ from latches in that their output change only with repeat to the clock, whereas latches change output when their inputs change. Flip-flops are characterized on the basis of the clock transition that cause the output change. There are positive edge-triggered, negative edge-triggered, and master/slave flip-flops. A positive edge-triggered flip-flop samples its inputs on the low-to-high clock transition. A negative edge-triggered flip-flop works in a similar fashion, with the input sampled on the high-to-low clock transition. A master-slave flip-flop is constructed from two stage separate flip-flops. The first stage (first flip-flop) samples the inputs on the rising edge of a clock signal. The second stage transfer them to the output on the falling edge of the clock signal. These circuits have two additional control inputs. These are Preset and Clear, which force the output of the flip-flop or latch to the logic-1 or logic-0 state, respectively, independent of the flip-flop or latch inputs.

1.2 MULTI BIT FLIP FLOPS

Multi-Bit Flip-Flops are capable of reducing the power consumption because they have shared inverter inside the flip-flop. It also minimizes the clock skew at the same time. Both single and multi-bit flip-flop have the same clock condition and same set and reset condition. Fig. 1.1 shows the example of multi-bit flip-flops. By replacing the two 1-bit flip-flops as one 2-bit flip-flop it share the clock buffer based on that we can achieve the power reduction. Based on the number of minimum size inverters that it can drive on a given rising or falling time we can calculate the driving capability of a clock buffer. Based on this shown in Fig 1.1 it share the clock buffer by replacing the two 1-bit flip-flop as one 2-bit flip-flop so it reduce the power consumption.

1.2.1 ADVANTAGES OF MERGING MULTI BIT FLIP-FLOPS

1. Smaller design area due to shared clock drivers and clock gating cells.
2. Less delay and power of the clock network due to fewer clock sinks and smaller capacitive load on the clock net;
3. Controllable clock skew because of common clock.
4. The required routing resource for a scan chain is greatly reduced because of fewer cells in a scan chain.

.

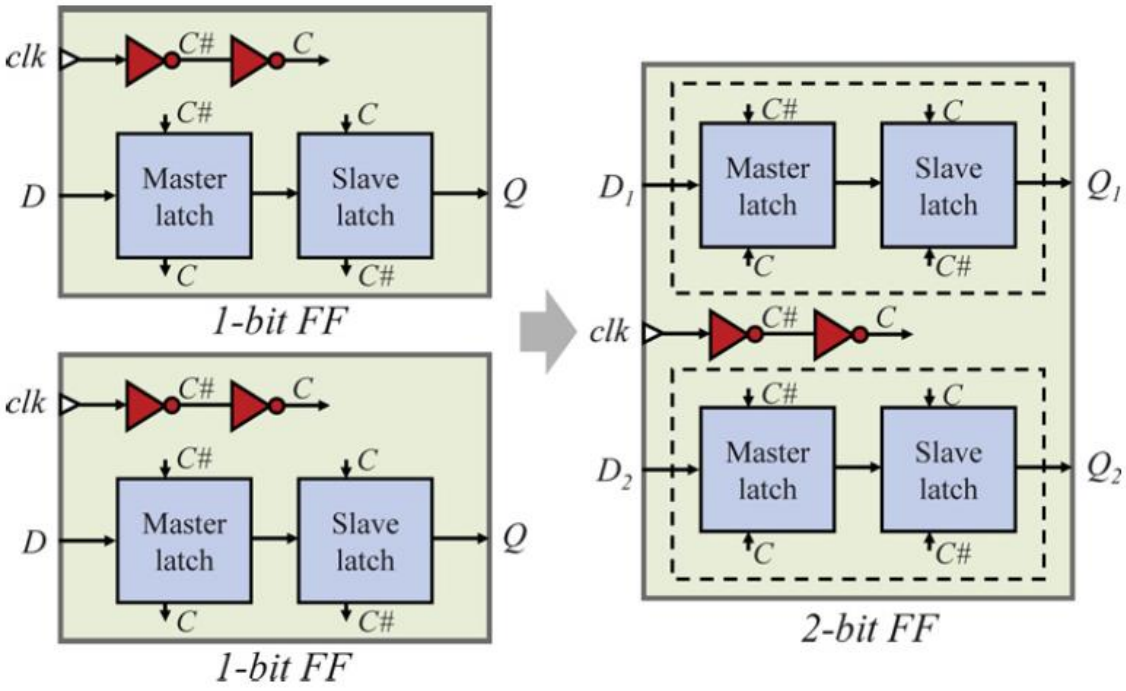


Fig 1.1 Example of merging two 1-bit flip-flops into one 2-bit flip-flop.

1.3 SOFTWARE USED

- ModelSim SE 6.4A
- Quartus II

1.4 ORGANIZATION OF THE REPORT

- **Chapter 2** discusses about literature survey
- **Chapter 3** discusses about clock distribution networks
- **Chapter 4** discusses about heuristic algorithm
- **Chapter 5** discusses about work flow
- **Chapter 6** gives the simulation results
- **Chapter 7** gives the conclusion

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

This chapter presents the literature survey for reduce the power and area and interconnecting wire length. Clock gating is a popular technique used in many synchronous circuits for reducing dynamic power dissipation. Clock gating saves power by adding more logic to a circuit to prune the clock tree. Pruning the clock disables portions of the circuitry so that the flip-flops in them do not have to switch states. Switching states consumes power. When not being switched, the switching power consumption goes to zero.

[1] P. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R.L.Allmon, "Post placement power optimization with Multi Bit Flip- Flop," IEEE J. Solid-State Circuits, vol. 33, no. 5, pp. 676–686, May 2012.

This paper presents a novel power optimization method by incrementally applying more multi-bit flip-flops at the post-placement stage to gain more clock power saving while considering the placement density and timing slack constraints, and simultaneously minimizing interconnecting wire length. Replacing several one bit flip flop with one Multi Bit Flip Flop to reduce the total area and dynamic power and it can be reduced upto 50%.Windows optimization technique is larger so that flip flop will perform slowly which is a major disadvantage .

[2] D. Duarte, V. Narayanan, and M. J. Irwin, "Power aware placement," in Proc. IEEE VLSI Comput. Soc. Annu. Symp., Pittsburgh, PA, Apr. 2005, pp. 52–57.

We present a power-aware placement method that simultaneously performs (1) activity-based register clustering that reduces clock power by placing registers in the same leaf cluster of the clock trees in a smaller area and (2) activity-based net weighting that reduces net switching power by assigning a combination of activity and timing weights to the nets with higher switching rates or more critical timing. Focuses on calculating the idle period of different flip

flop and inserting the gating logic into netlist to achieve the total power by 25.3%.The net switching power can be achieved by 25.4 % and then wire length can also be reduced which is a major disadvantage .

[3] H. Kawagachi and T. Sakurai, "Impact of technology scaling in the clock power," in VLSI Circuits Dig. Tech. Papers Symp., Jun. 2003, pp. 97–98.

The clock distribution and generation circuitry is known to consume more than a quarter of the power budget of existing microprocessors. A previously derived clock energy model is briefly reviewed while a comprehensive framework for the estimation of system wide (chip level) and clock sub-system power as function of technology scaling is presented. This framework is used to study and quantify the impact that various intensifying concerns associated with scaling (i.e., increased leakage currents, increased inter wire capacitance) will have on clock energy and their relative impact on the overall system energy. The results obtained indicate that clock power will remain a significant contributor to the total chip power, as long as techniques are used to limit leakage power consumption. Increase the flexibility that covers the clock distribution and clock generation circuit to consume total power by 40%. Clock skew problem is only reduced by 30% which is a major disadvantage.

[4] W. Hou, D. Liu, and P.-H. Ho, "Automatic register banking for low power clock trees," in Proc. Quality Electron. Design, San Jose, Mar. 2010, pp. 647–652

Replacing some flip flop with multibit flip flop without affecting the performance and total wire length can be minimized by 20-30%. Using dual bit flip flop to save the clock power is 11 .22% and the replacement of flip flop during switching rate is 10.43% which is a major disadvantage.

[5] Y. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang, "High performance microprocessor design," in Proc. Design Autom. Conf., Jun. `1998, pp. 795–800.

Three generations of Alpha microprocessors have been designed using a proven custom design methodology. The performance of these microprocessors was optimized by focusing on high-frequency design. The Alpha instruction set architecture facilitates high clock speed, and the chip organization for each generation was carefully chosen to meet critical paths. Digital has developed six generations of CMOS technology optimized for high-frequency design. Complex circuit styles were used extensively to meet aggressive cycle time goals. Focus on high frequency design to achieve high performance and to improve the complexity of the circuit. In single supply voltage system the clock power can be reduced by 25.45 %, and in multiple supply voltage system the clock power can be reduced by 26.15 % which is a major disadvantage.

[6] W. Aloisi and R. Mita, "Gated-clock design of linear-feedback shift registers," IEEE Trans. Circuits Syst., II, Brief Papers, vol. 55, no. 5, pp. 546–550, Jun. 2008.

In the paper [6], the author described that the gated clock design approach for LFSRs which can lead to power reduction without unduly complicating the traditionally simple topology. The main drawback in traditional LFSR generators is high power Consumption. To analytically evaluate the power consumption of the gated clock approach applied to a LFSR, we have to take into account also the dissipation introduced by the extra gates that are employed to implement the gated clock circuits, as well as the load effects introduced by these gates with respect to the traditional one. LFSRs implemented with and without gated-clock design. Let us observe that the power consumption of the FFs is reduced of about 20% after applying the gating design, but the overall power reduction is about 10% due to consumption of the extra gates that are introduced to implement the gating circuit.

[7] Jhen-Hong He1, Li-Wei Huang2, Jui-Hung Hung3, Yu-Cheng Lin4, Guo-syuan Liou5,Tsai-Ming Hsieh "Clock Network Power Saving Using Multi-Bit Flip-Flops in Multiple Voltage Island Design" Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013).

In the paper [7], the author described for the single supply voltage system that the total power consumption is the sum of all flip-flops power consumption in the final design. For multiple supply voltage system, the total power consumption is the sum of the power consumption of all flip-flops and level shifters. Moreover, we can effectively reduce the clock power consumption by using the concept of EPC both single and multiple supply voltage designs. The experimental results show we can reduce the clock power 25.48% on the average in single supply voltage. In multiple supply voltage system, we can reduce the clock power up to 26.16%, and the number of level shifters is decreased 31.76% on the average.

[8] M. Donno, E. Macii, and L. Mazzoni, "Power-aware clock tree planning," in Proc. Int. Symp. Phys. Design, 2004, pp. 138–147.

In the paper [8], the author described that the clock signal driving a FF is disabled (gated) when the FFs state is not subject to change in the next clock cycle. Data driven gating is causing area and power overheads that must be considered. In an attempt to reduce the overhead, it is proposed to group several FFs to be driven by the same clock signal, generated by or ing the enabling signals of the individual FFs. This may however, lower the disabling effectiveness. It is therefore beneficial to group FFs whose switching activities are highly correlated and derive a joint enabling signal.

[9] S. Wimer and I. Koren, "The Optimal fan-out of clock network for power minimization by adaptive gating," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 10, pp. 1772–1780, Oct. 2012.

In the paper [9], the author described a model for data-driven gating which is developed based on the toggling activity of the constituent FFs . The optimal fan out of a clock gater yielding maximal power savings is derived based on the average toggling statistics of the individual FFs, process technology, and cell library in use. .The resulting clock gating methodology achieves 10% savings of the total clock tree switching power.

2.2 COMPARISON TABLE FOR VARIOUS METHODS

This table specifies the various implementation of flip-flop to optimize the power and to achieve the net switching activity. Although the drivers are very wide devices it was found that for all technologies the share of the clock power that is due to leakage is at most 2.5%. Technology optimizations and dynamic runtime techniques for leakage reduction will become standard for clock power and will remain a major contributor to the total system power.

Implementation	**Description**	**Motivation**
Post Placement Power Optimization	Progressive Windows Based Optimization	Reduce the power & Interconnecting wire length
Power Aware Placement.	Register clustering & net weighting	Reduced area &wire length
Impact Of Technology Scaling	Scaling interconnect & impact of leakage	Reduced leakage power
Flip-Flop Merging And Relocation	Net Switching technique	Switching rate less &save clock power
Clock Power Using Multibit Flip-flop	Three phase algorithm	Reduced power single & multiple supply voltage system
Clock Power Flip-Flop For Future Soc Applications	CDMFF + CPSFF Proposed in LCPTFF	Reduced power &area
A Low-Swing Clock Double-Edge Triggered Flip-Flop	LCDFF performed Charging &discharging Technique	Saving power in flip-flop operation & clock network

Table 2.1 comparison table for various methods

CHAPTER 3

CLOCK DISTRIBUTION NETWORKS

3.1 INTRODUCTION

In a synchronous digital system clock signal is used to synchronize the movement of data within the system. Clock signals are required to be distributed at physically remote locations of an integrated circuit. Clock signals transitions drive all the synchronous elements of a digital circuit like Flip Flops and Memories. These elements are referred to as Sinks. Clock Distribution Networks are circuits that distribute a clock signal from a central global clock source at the Centre of the Integrated circuit to all the sinks which use it.

In the process of Clock distribution, the clock signal traverses through a lot of interconnect networks and buffers which are a part of the clock distribution network. These elements introduce delay in the clock signal path. Ideally, a clock signal should arrive at all the sinks at the same time. But due to the variations in parameters like wire interconnect length, temperature variations, capacitive coupling and process variations, the arrival time of the clock transition at different sink locations varies. This spatial variation in the arrival time of the clock transition on an integrated circuit is commonly referred to as Clock Skew**.** For two points i and j, if the arrival times of the clock signals are a_i and a_j respectively then the clock skew between two points is given by $d(i,j) = a_i - a_j$ The figure below illustrates the reason for clock skew

Clock signals typically have the highest fan out and operate at the highest speeds in a synchronous digital system. Since the clock signals are used to synchronize the operations of the entire digital circuit the clock transitions should be sharp and should have minimum possible skew to avoid any data integrity errors or race conditions. As the frequency of operation of the synchronous circuit increases the circuit becomes more and more susceptible to clock skew i.e. the timing becomes more and more critical.

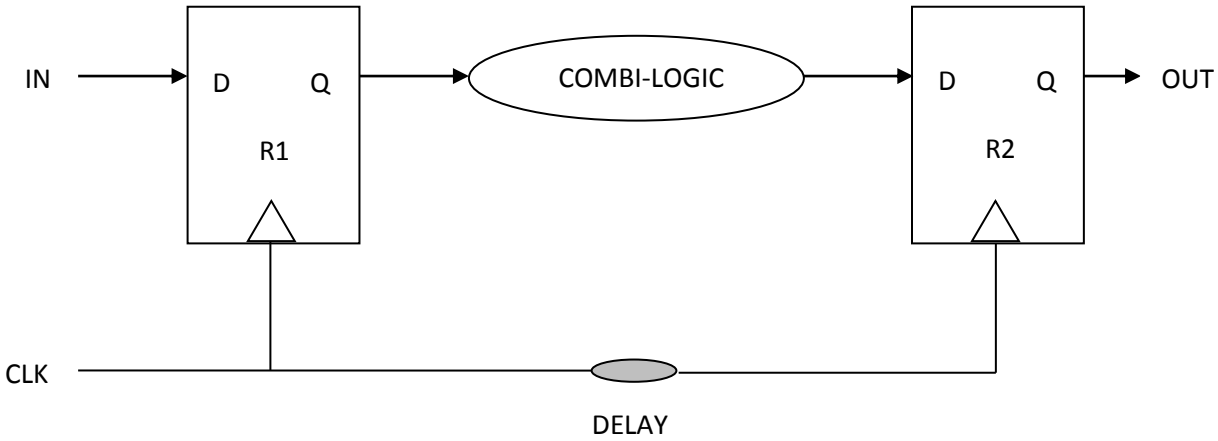


Fig 3.1 Clock Skew Illustration

At this point we introduce another term called Slew Rate. Slew is maximum rate of change of signal in a circuit. Slew depends upon the time it takes for a signal to rise (fall) from logic low (logic high) to logic high (logic low). More commonly, it depends on the time it takes for a signal to rise from 10% to 90% or fall from 90% to 10% of the supply voltage. For a clock distribution network, in addition to achieving minimum skew, it should also try to obtain as high slew rate as possible (i.e. minimum time to change from one logic level to another).

Another factor of prime importance in clock distribution network design is the power consumption. Clock distribution networks account for a significant component of power consumption on an integrated circuit. It is therefore absolutely essential to build clock distribution networks with minimum possible power consumption.

3.2 CLOCK DISTRIBUTION TECHNIQUE

There are different techniques employed in Integrated Circuit design to minimize the clock skew. A few of the techniques are listed below:

3.2.1 Clock Trees

3.2.2 Single Clock Mesh/Grid

3.2.3 Clock Trees with Multiple Local Meshes

3.2.1 CLOCK TREE

Clock trees work on the principle that relative phase of the clock at two sinks is more important than the absolute delay in the clock path from the source to the sink. Clock trees are balanced clock distribution networks. The most commonly used trees used for clock distribution are the H-TREE networks. A simple H-Tree structure is shown in the figure below

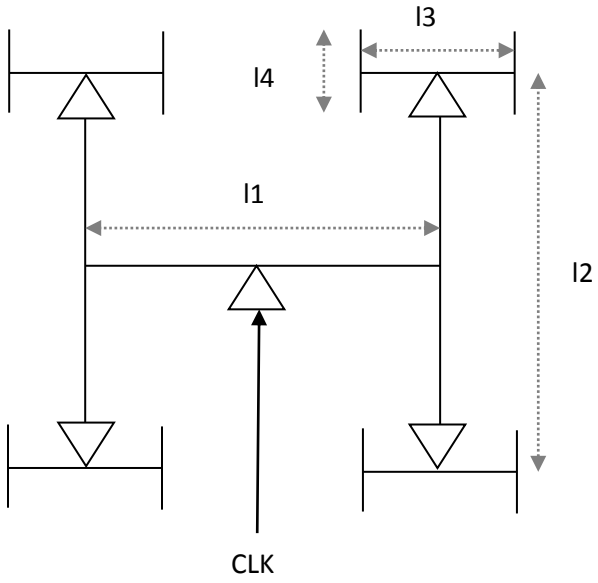


Fig 3.2 H-Tree Network

As seen from the above figure the H-Tree consists of a central buffer driven by the main clock source and a buffer at each vertex of the H-Tree. This H-Tree structure is a symmetric structure in terms of both the lengths of the H-tree segments as well as the buffer sizes. Therefore ideally, the skew between the different vertices of the H-Tree is zero. It might take a certain amount of delay for the clock signal to travel from the central clock source to the vertex due to the buffer delays. But since the buffers are identical the delay is the same on all the paths, resulting in zero skew. The only factors which may introduce the skew are environmental and process variations.

Clock distribution networks with H-Tree structure are constructed by recursively connecting H-Tree structures to each other as shown in

Fig 3.2 H-Tree Network. The H-Tree structure at each level has lengths equal to half of the previous level. For example if a level 1 H-Tree has lengths of l1 and l2, then the level 2 H-tree will have lengths l3=l1/2 and l4=l2/2.

ADVANTAGE

- H-Tree clock distribution network are ideally zero skew, low power, low area and ease of generation.

DISADVANTAGES

- The sink distribution in an integrated circuit may not be as uniform as the H-Tree structure.
- There might be a large concentration of sinks at one location whereas a very less sink concentration at another location.

In such a case the capacitive load distribution connected to the H-Tree vertices is highly irregular. This will result in non zero skew among sinks connected to different vertices. However techniques like introduction of dummy loads, wire snaking can be employed to overcome these disadvantages.

3.2.2 SINGLE CLOCK MESH

In this method a grid or mesh is used in the final stage of the clock distribution network. A balanced tree like an H-Tree is used to connect the main clock source to the local grid. The grid/mesh helps to achieve a low local skew i.e. the skew close to the sinks and the H-Tree structures help to achieve a low skew in the path from the clock source to the clock grid. Therefore this structure combines the advantages of both the Regular H-Tree structure as well as a grid.

ADVANTAGES

- The single clock mesh structure is that it helps to achieve very low skews as compared to the only H-Tree structure.

- This structure accommodates for late design as the grid is easily accessible from various points on the integrated circuit.

Therefore the clock mesh can be designed in the early stages.

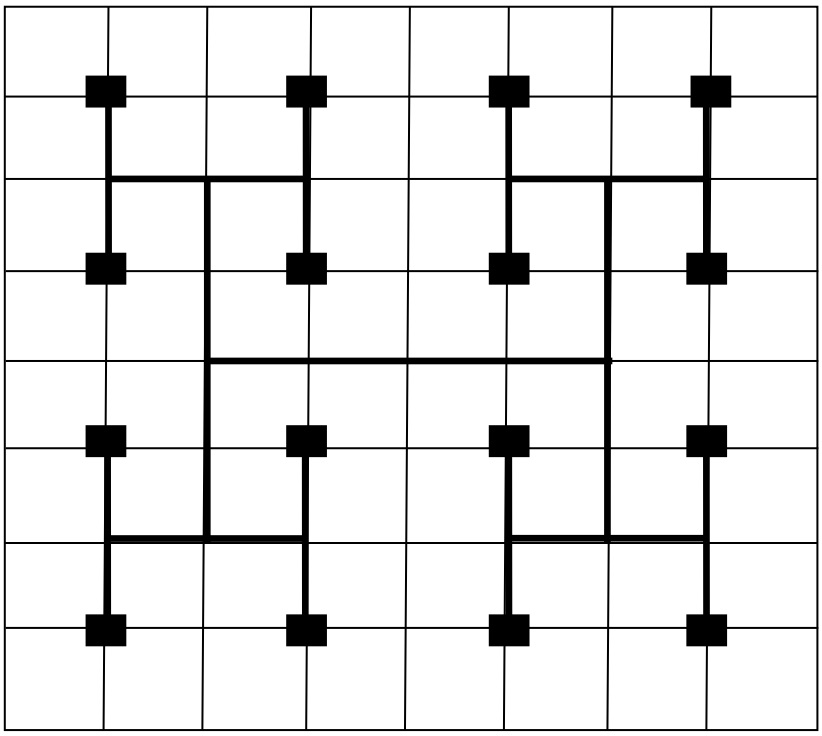


Fig 3.3 2-Level H-Tree with Single 8X8 Mesh

DISADVANTAGES

- The major disadvantages of the single clock mesh structure are its huge power consumption and wire length.
- It is not possible to selectively turn off the clock (for power reduction) to a certain area of the integrated circuit because of the single mesh structure i.e. ***clock gating*** is not possible in this approach.

3.2.3 CLOCK TREE WITH MULTIPLE LOCAL MESH

This is an improvement over the previous clock mesh scheme. In this case instead of having a single mesh, there will be a local mesh for a group of sinks. Thus we would have a number of local meshes. All these local meshes would be connected to the main clock source using balanced trees like H-Trees.

This architecture can be made reconfigurable by connecting two adjacent local meshes by transmission gates. The transmission gates will be turned on only if the two meshes connecting to it are turned on. The insertion of transmission gates brings this architecture close to the single mesh architecture while still allowing to independently turn off power supply to some of the local meshes.

ADVANTAGE

- The power consumption can be reduced by selectively turning off certain local meshes when not in use. This is clock gating.

DISADVANTAGE

- The clock skew is slightly worse when compared to single clock mesh scheme.

The structure of clock tree with local mesh is shown in the

Fig 3.4 Clock Tree with Local Mesh below. The figure also shows the transmission gates inserted in between the local meshes.

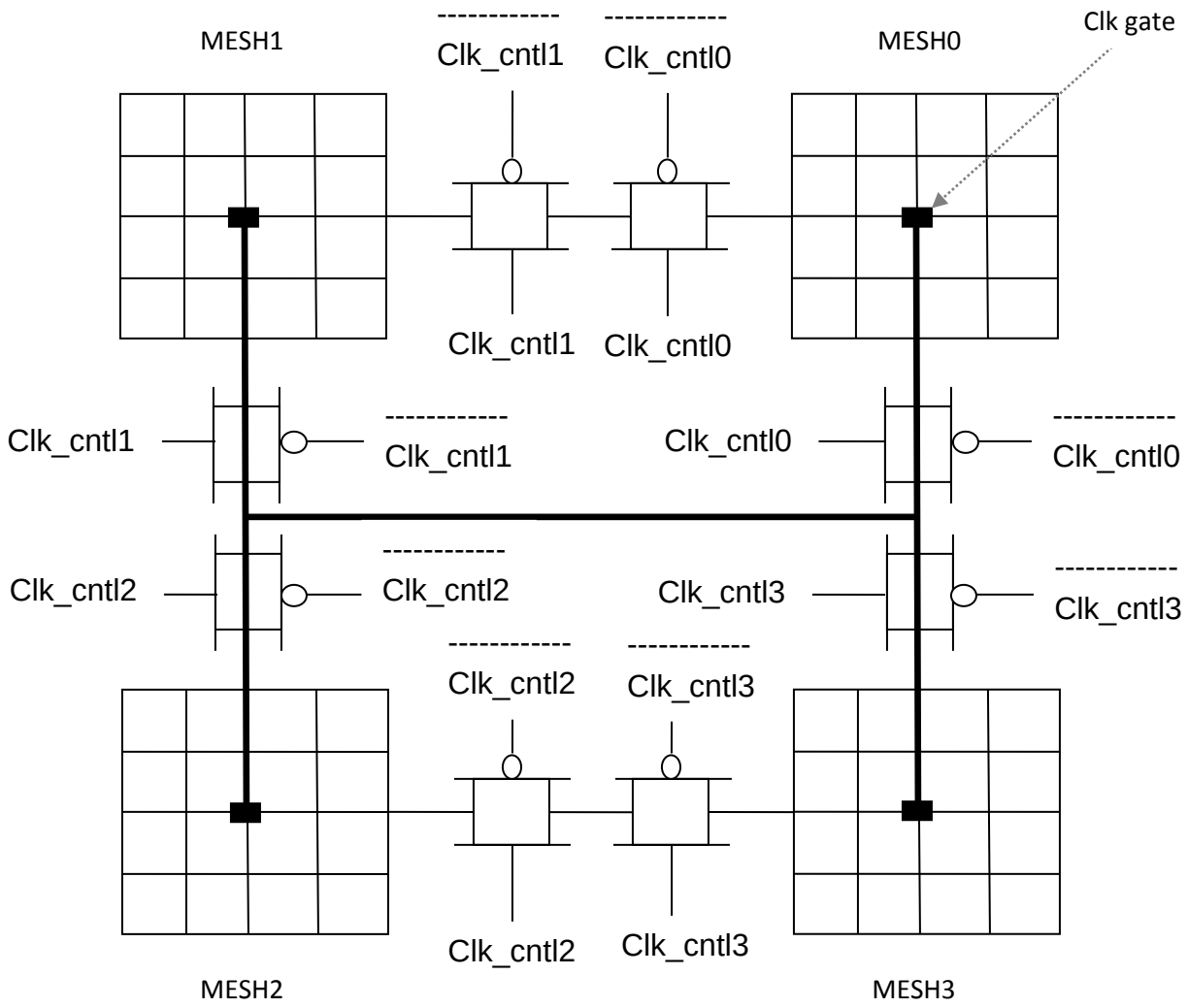


Fig 3.4 Clock Tree with Local Mesh and Transmission Gates

3.4 DRAM INTERNAL STRUCTURE

Dynamic random-access memory (DRAM) is a type of random-access memory that stores each bit of data in a separate capacitor within an integrated circuit. The capacitor can be either charged or discharged; these two states are taken to represent the two values of a bit, conventionally called 0 and 1. Since even "nonconducting" transistors always leak a small amount, the capacitors will slowly discharge, and the information eventually fades unless the capacitor charge is refreshed periodically. Because of this refresh requirement, it is a dynamic memory as opposed to static random access memory (SRAM) and other static types of memory.

DRAM, as the most cost effective solid-state storage device, is used in a wide variety of applications, from embedded appliances to the largest of supercomputers. Today "computers" span the spectrum of servers, workstations, desktops, notebooks, internet appliance, and game boxes. Each has unique set of design requirements for their respective memory systems. The variance in design requirements becomes even larger when graphics cards, routers and switches, mobile electronics, games and all other electronic devices using DRAM are included.

The advantage of DRAM is its structural simplicity: only one transistor and a capacitor are required per bit, compared to four or six transistors in SRAM. This allows DRAM to reach very high densities. Unlike flash memory, DRAM is volatile memory (vs. non-volatile memory), since it loses its data quickly when power is removed. The transistors and capacitors used are extremely small; billions can fit on a single memory chip. Due to the nature of its memory cells, DRAM consumes relatively large amounts of power, with different ways for managing the power consumption.

Dynamic Random Access Memories (DRAM) are based upon circuits which have a single capacitor-transistor pair for each bit of information stored in the device. With devices currently available with 256Mbits, and fabricated in laboratories with 1Gbits, the capacity and scale of these modern devices is enormous. As noted earlier, these circuits are characterized as dynamic because if they are not periodically refreshed, the data in the capacitors will be lost due to transistor leakage. All DRAM have an array of these single bit cells, which are addressed via the row(or word) lines and from which the data is accessed via the bit lines.

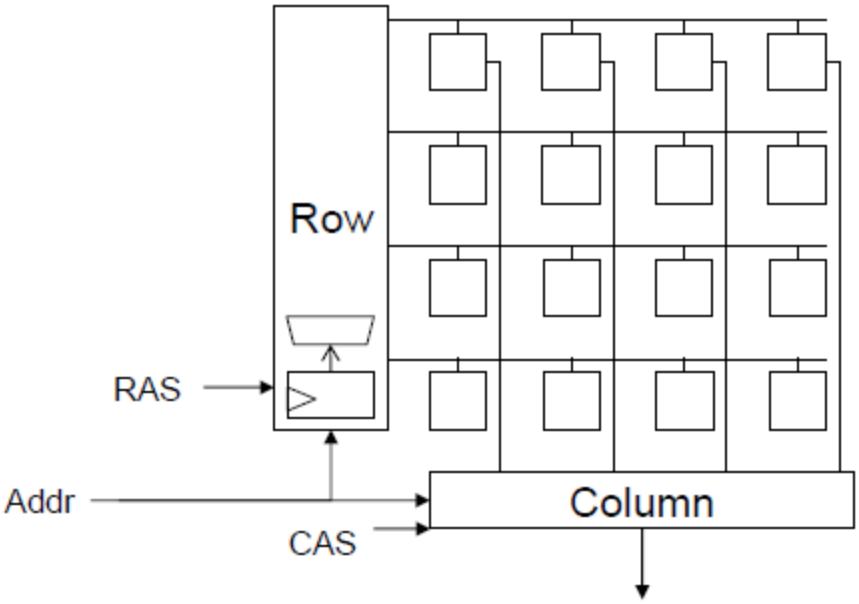


Fig 3.5 Simplified DRAM Internal Structure

This array is common among all DRAM devices, but the number of arrays which are located in each device is a design/performance trade-off which varies from architecture to architecture. Smaller arrays are generally lower latency because of the lower capacitance associated with the word and bit lines. A larger number of arrays for a fixed address space will generally yield a higher level of parallelism because of the limitation on pipelining consecutive accesses to the same array or bank. Trying to improve performance either through increasing the number of arrays in the device, or by reducing the size of the arrays in the device will increase the amount of die area per bit required for the layout of the device. Increasing the area per bit of the device increases the cost per bit of the device, and thus we have a situation where cost and performance correlate positively, while it is advantageous to increase performance and decrease cost.

CHAPTER 4

HEURISTIC ALGORITHM

Our algorithm is split into three stages as, First stage is to identify the merged flip-flops. In second stage we can build the combination table according to the identified overlapped region in the first stage. We can build the combination table in binary tree representation for easy representation. In the third stage, performing the merging flip-flops based on the combinational table.

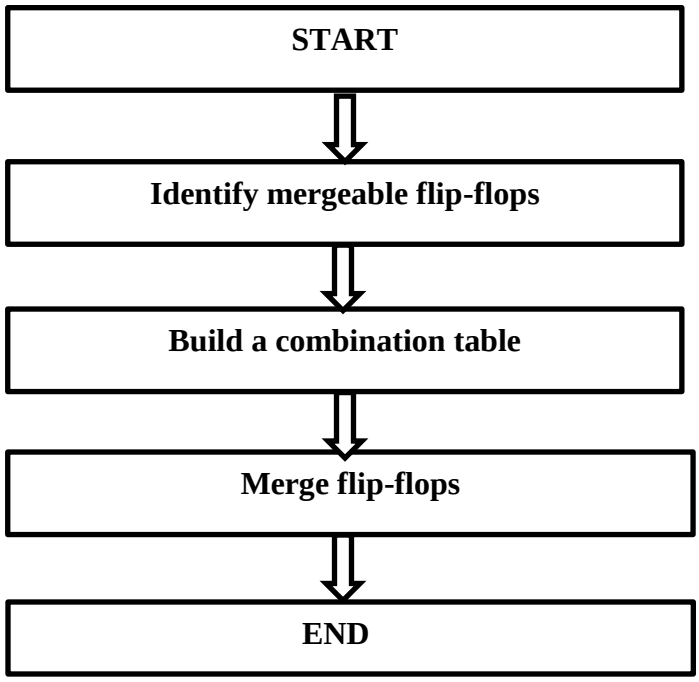


Fig 4.1 Flow chart of heuristic algorithm.

4.1 IDENTIFY MERGEABLE FLIP-FLOPS

In this algorithm, we have to identify a legal placement region for each flip-flop. First, the feasible placement region of a flip-flop associated with different pins are found based on the timing constraints defined on the pins. Then, the legal placement region of the flip-flop *fi* can be obtained by the overlapped area of these regions. However, because these regions are in the

diamond shape, it is not easy to identify the overlapped area. Therefore, the overlapped area can be identified more easily if we can transform the coordinate system of cells to get rectangular regions.

4.2 BUILD A COMBINATION TABLE

To perform the efficient process we build the combinational table. If without performing the combinational table we merged the flip-flops it will not be efficient because mergeable flip-flops is not in intersection value, it also time waste of designing. Building of combinational table is based on the library initialization value. To build a combination table, which defines all possible combinations of flip-flops in order to get a new multi-bit flip-flop provided by the library. The flip-flops can be merged with the help of the table. Maximum library size is initialized by library and minimum size is denoted as 1 bit because we are going merge the number of one bit flip-flops. Now a combination table is to be built, which records combining two n1s all possible combinations of flip-flops to get feasible flip-flops before replacements. Thus, we can gradually replace flip-flops according to the order of the combinations of flip-flops in this table.

Step1: Initialize the Library and the Combination Table

Consider a library *L* that provides two types of flip-flops, whose bit widths are 1 and 4 (i.e. bmin = 1 and bmax = 4), in Fig. 3(a). We first initialize two combinations *n*1 and *n*2 to represent these two types of flip-flops in the table *T* [see Fig. 4.2(a)].

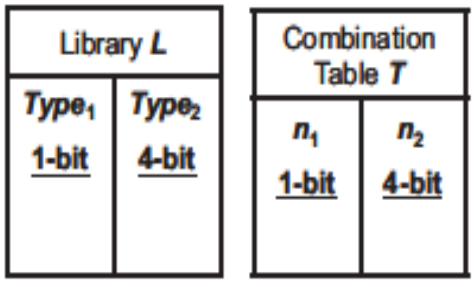


Fig. 4.2(a) Initialize the library L and the combination table T.

Step2: Insert Pseudo Type

Next, the function Insert Pseudo Type is performed to check whether the flip-flop types with bit widths between1 and 4 exist or not. Thus, two kinds of flip-flop types whose bit widths are 2 and 3 are added into *L* and all types of flip-flops in *L* are sorted according to their bit widths [see Fig. 4.3(b)]. Fig. 4.3(b) Pseudo types are added into *L* and the corresponding binary tree is also build for each combination in *T*.

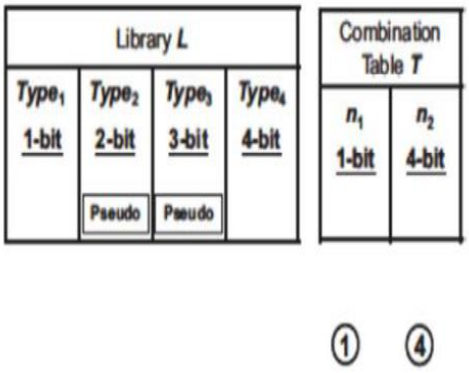


Fig. 4.3(b) Pseudo types are added into L, and the corresponding binary tree is also build for each combination in T.

Step 3: Create New Combination n3

By combining two1-bit flip-flops in the first and combination of two n1 and make a new Combination *n*3 can be obtained [Fig. 4.4(c)].

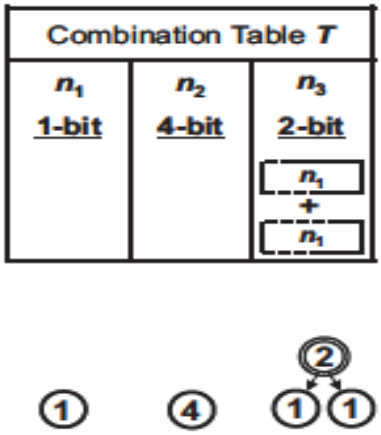


Fig. 4.4(c) New combination n3 is obtained from combining two n1s

Step 4: Create New Combination *n*4 and *n*5

Similarly, we can get a new combination *n*4 (*n*5*)* by combining *n*1 and *n*3(two *n*3's) [see Fig. 4.5 (d)]. Finally, *n*6 is obtained by combing *n*1 and *n*4.

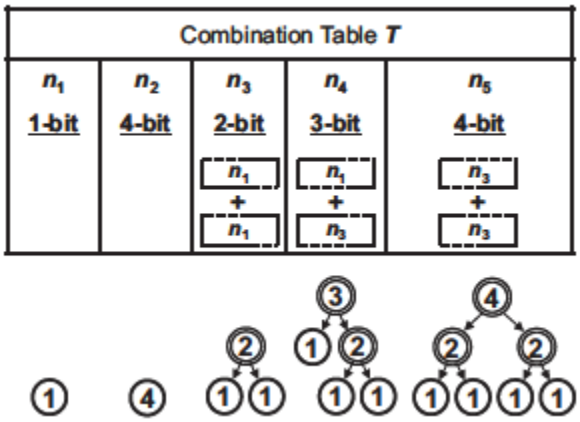


Fig. 4.5(d) New combination n4 is obtained from combining n1 and n3, and the combination n5 is obtained from combining two n3s.

Step 5: Create New Combination *n6*All possible combinations of flip-flops are shown in Fig. 4.6(e). Among these combinations, *n*5 and *n*6 are duplicated since they both represent the same condition, which replaces four

1-bit flip-flops by a 4-bit flip-flop. To speed up our program, *n*6 is deleted from T rather than *n*5 because its height is larger. After this procedure, *n*4 becomes an unused combination [see Fig. 4.6(e)] since the root of binary tree of *n*4 corresponds to the pseudo type, type3, in L and it is only included in *n*6.

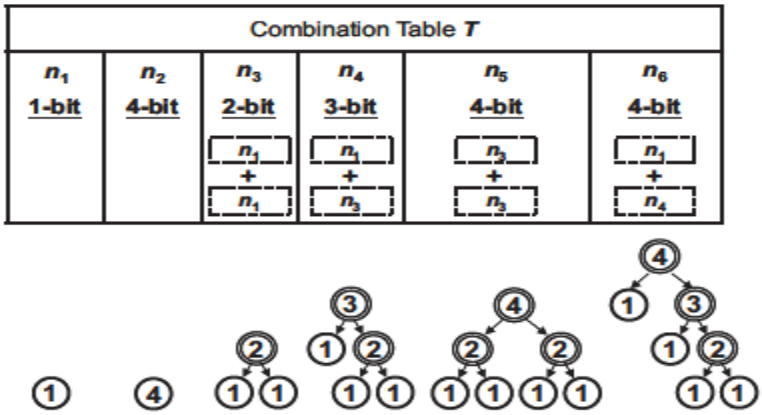


Fig. 4.6(e): New combination n6 is obtained from combining n1 and n4.

Step 6: Deleting Unused Combination

After deleting *n*6, *n*4 is also need to be deleted. The last combination table T is shown in Fig. 4.7(f).

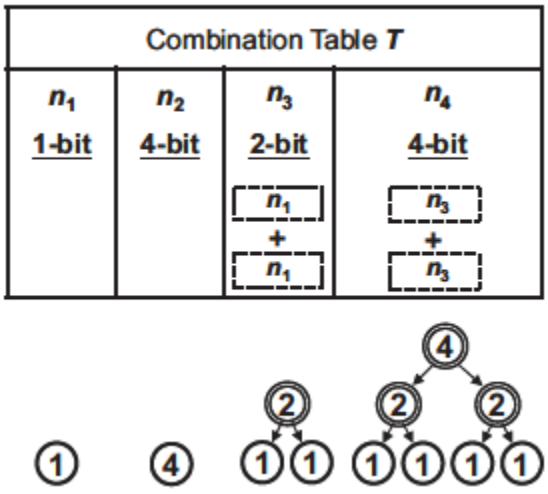


Fig. 4.7(f) Last combination table is obtained after deleting the unused combination in (e).

4.3 MERGE FLIP-FLOPS

The flip-flops can be merged with the help of the table. After the legal placement regions of flip-flops are found and the combination table is built, we can use them to merge flip-flops. To speed up our program, we will divide a chip into several bins and merge flip-flops in a local bin. However, the flip-flops in different bins may be mergeable. Thus, we have to combine several bins into a larger bin and repeat this step until no flip-flop can be merged anymore.

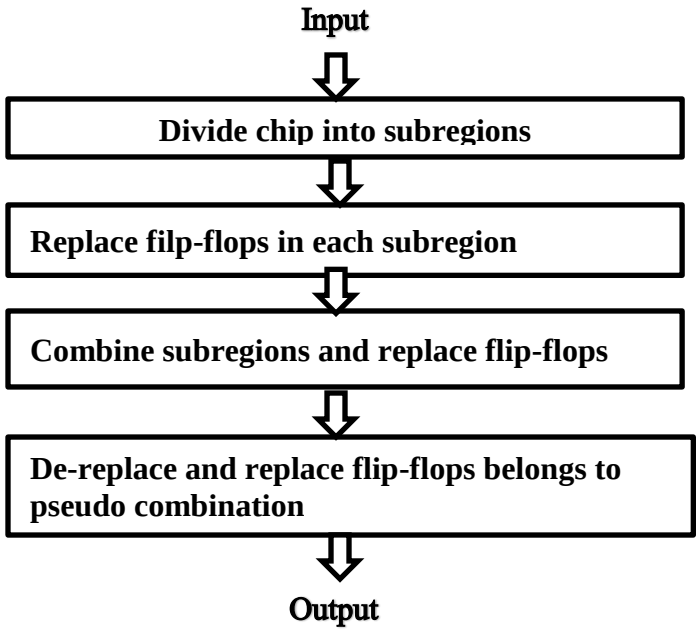


Fig 4.8 Detailed flow to merge flip-flops.

Step1: Sets of Flip-Flops Before Merging

A library containing three types of flip-flops (1-, 2-and 4-bit), we first build a combination table *T* as shown in Fig. 4.9(a).

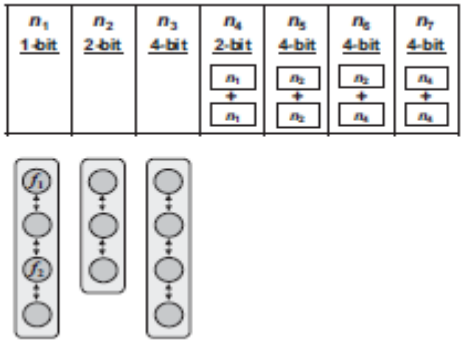


Fig 4.9(a) Sets of flip-flops before merging

Step 2: Replacing 2-Bit Flip-Flop *f*3.

In the beginning, the flip-flops with various types are, respectively, linked below *n*1, *n*2 and *n*3 in *T* according to their types. Suppose we want to form a flip-flop in *n*4, which needs two 1-bit flip-flops according to the combination table. Each pair of flip-flops in *n*1 are selected and checked to see if they can be combined If there are several possible choices, the pair with the smallest cost value is chosen to break the tie. In Fig. 4.9(a), f1 and f2 are chosen because their combination gains the smallest cost. Thus, we add a new node *f*3 in the list below *n*4 and then delete *f*1 and *f*2 from their original list [see Fig. 4.10(b)].

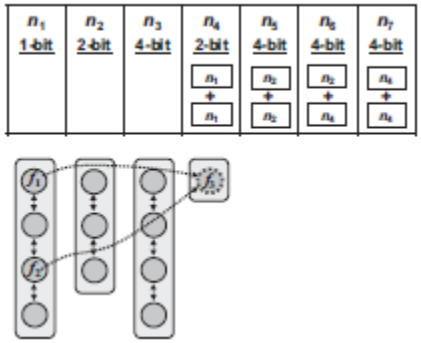


Fig 4.10(b) Two 1-bit flip-flops, f1 and f2, are replaced by the 2-bit flip-flop f3.

Step 3: Replaced 2-Bit Flip-Flop *f*6.

Similarly, *f*4 and *f*5 are combined to obtain a new flip- flop *f*6 and the result is shown in Fig. 4.11 (c).

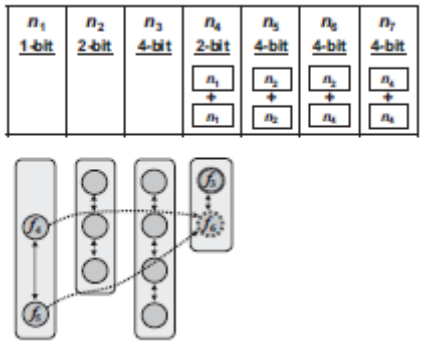


Fig 4.11(c) Two 1-bit flip-flops, f4 and f5, are replaced by the 2-bit flip-flop f6.

Step 4: Replacing 4-Bit Flip-Flop *f*9

After all flip-flops in the combinations of 1-level trees (*n*4 and *n*5*)* are obtained as shown in Fig. 4.12(d), we start to form the flip-flops in the combinations of 2-level trees (*n*6 and *n*7*)*.

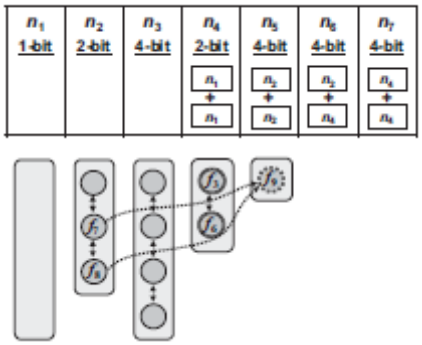


Fig 4.12(d) Two 2-bit flip-flops, f7 and f8, are replaced by the 4-bit flip-flop f9.

Step 5: Replacing 4-bit flip-flop *f*1

In Fig. 4.13 (e), there exist some flip-flops in the lists below *n*2 and *n*4 and we will merge them to get flip-flops in *n*6 and *n*7, respectively. Suppose there is no overlap region between the couple of flip-flops in *n*2 and *n*4. It fails to form a 4-bit flip-flop in *n*6. Since the 2-bit flip-flops *f*3 and *f*6 are merge-able, we can combine them to obtain a 4-bit flip-flop *f*10 in *n*7.

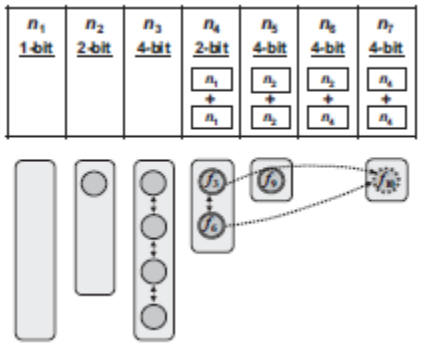


Fig 4.13(e): Two 2-bit flip-flops, f3 and f6, are replaced by the 4-bit flip-flop f10.

Step 6: Sets of Flip-Flops After Merging

Finally, there exists no couple of flip-flops that can be combined further, the procedure Finishes as shown in Fig. 4.14(f).

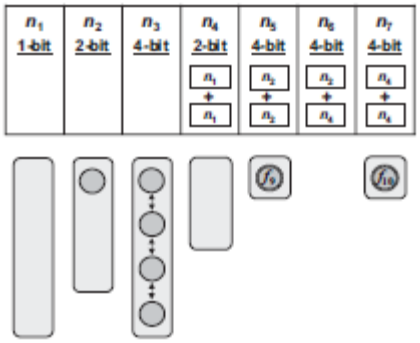


Fig 4.14(f): Sets of flip-flops after merging.

4.3.1 REGION PARTITION

To speed up our problem, we divide the whole chip into several sub regions. By suitable partition the computation complexity of merging flip-flops can be reduced significantly. As shown in Fig 4.15, we divide the region into several sub regions and each sub region contains six bins, where a bin is the smallest unit of a sub region.

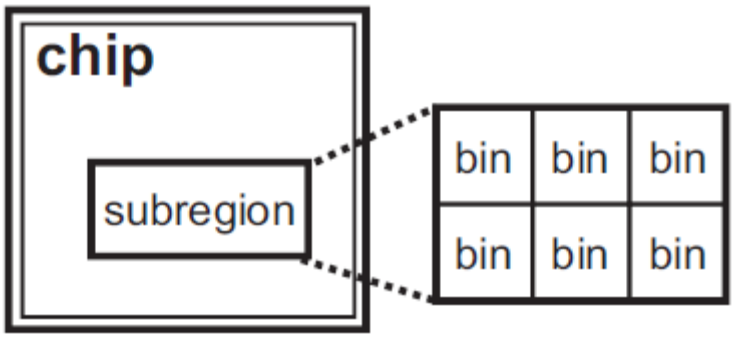


Fig. 4.15: Region partition with six bins in one sub region.

4.3.2 BOTTOM-UP FLOW OF SUBREGION COMBINATIONS

As shown in Fig. 4.16(a), there may exist some flip-flops in the boundary of each subregion that cannot be replaced by any flip-flop in its subregion. However, these flip-flops may be merged with other flip-flops in neighboring subregions as shown in Fig. 4.16(b). Hence, to reduce power consumption further more, we can combine several subregions to obtain a larger subregion and perform the replacement again in he new subregion again. The procedure repeats until we cannot achieve any replacement in the new subregion. Fig. 4.17 gives an example for this hierarchical flow. As shown in Fig 4.17(a), suppose we divide a chip into 16 subregions in the beginning. After the replacement of flip-flops is finished in each subregion, four subregions are combined to get a larger one as shown in Fig 4.17(b). Suppose some flip-flops in new subregions still can be replaced by new flip-flops in other new subregions, we would combine four subregions in Fig. 4.17(b) to get a larger one as shown in Fig 4.17(c) and perform the replacement in the new subregion again. As the procedure repeats in a higher level, the number of mergeable flip-flops gets fewer. However, it would spend much time to get little improvement for power saving. To consider this issue, there exists a trade-off between power saving and time consuming in our program.

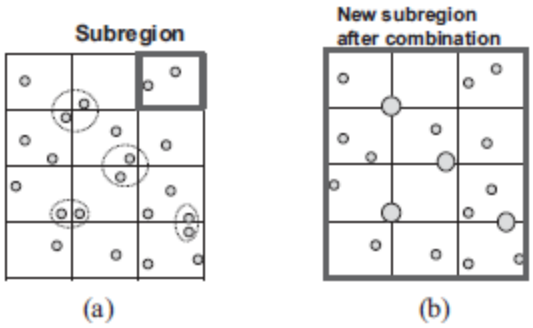


Fig. 4.16 Combination of flip-flops near subregion boundaries. (a) Result of replace flip-flops in each subregion. (b) Result of replace flip-flops in each new subregion which is obtained from combining twelve subregion in (a).

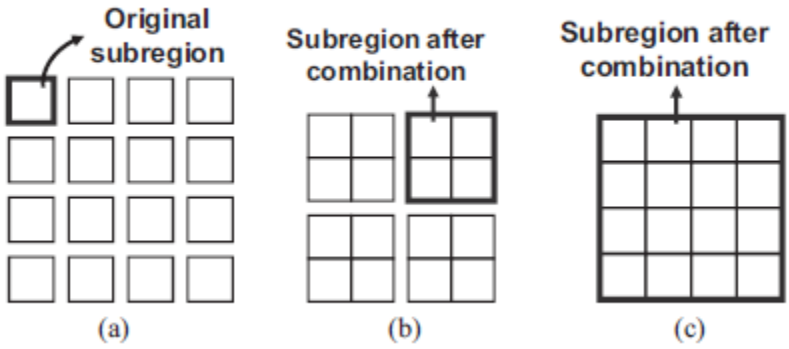


Fig. 4.17 Combination of subregions to a larger one. (a) Placement is originally partitioned into 16 subregions for replacement. (b) Subregion bounded by bold line is obtained from combining four neighboring subregions in (a). (c) Subregion bounded by bold line is obtained from combining four subregions in (b).

4.4. CLOCK GATING

Several techniques to reduce the dynamic power have been developed, of which clock gating is predominant. Ordinarily, when a logic unit is clocked, its underlying sequential elements receive the clock signal regardless of whether or not they will toggle in the next cycle.

Clock enabling signals are usually introduced by designers during the system and clock design phases, where the inter-dependencies of the various functions are well understood. In contrast, it is very difficult to define such signals in the gate level, especially in control logic, since the inter-dependencies among the states of various flip-flops depend on automatically

synthesized logic. There is a big gap between block disabling that is driven from the HDL definitions, and what can be achieved with data knowledge regarding the flip-flops activities and how they are correlated with each other.

The clock gating technique has been developed to avoid unnecessary power consumptions, like the power wasted by timing components during the time when the system is idle. Specifically for flip-flops, clock gating means disabling the clock signal when the input data does not alter the stored data. It can be applied from the system level where the entire functional unit can be selectively set into sleep mode, or from the sequential/combinational circuit level where some parts of the circuit are in sleep mode while the rest of the block are operating. But Clock gating does not come for free. Extra logic and interconnects are required to generate the clock enabling signals, and the resulting area and power overhead must be considered. In the extreme case, each clock input of a flip-flop can be disabled individually, yielding maximum clock separation. This, however, results in high overhead. Thus, the clock disabling circuit is shared by a group of several flip-flops in an attempt to reduce the overhead.

4.4.1. HOW TO IMPLEMENT CLOCK GATING?

When there is no activity at a register “data” input, there is no need to clock the register and hence the “clock” can be gated to switch it off . If the clock feeds a bank of registers, an “enable” signal can be used to gate the clock, which is called the “clock gating enable”.

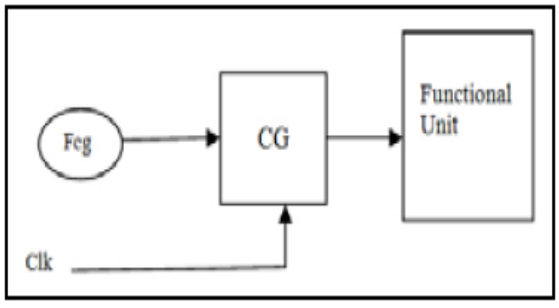


Fig. 4.16 Clock Gating Principle

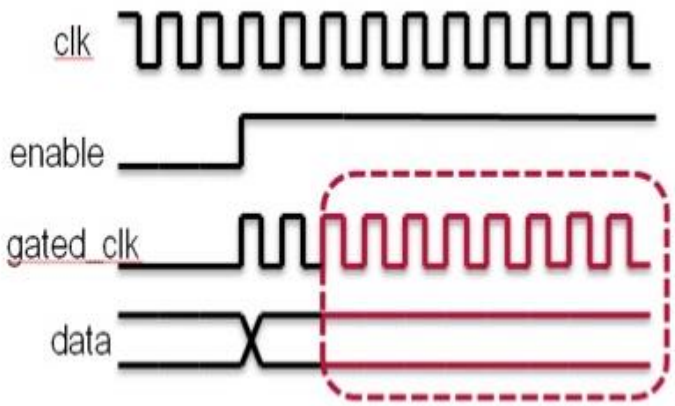


Fig. 4.17 Clock gating Waveform

The basic principle behind this power optimization technique is as shown in fig.4.16. The clock signal is computed by a function Fcg, here CLK is the systems clock signal and CLKG is the gated clock of the functional unit. So whenever the clock signal is not needed, unwanted switching activities will be suppressed and hence dynamic power reduces. Clock gating means to mask unwanted signal transitions from propagating forward. The same idea has been applied to clock signal. When the clock signal of a functional module is not required for some extended period, we use a gating function to turn off the clock feeding the module. Clock gating saves power by reducing unnecessary clock activities inside the gated module.

CHAPTER 5

WORK FLOW

The aim of project is to reduce the power and area using flip flop merging and clock gating technique.

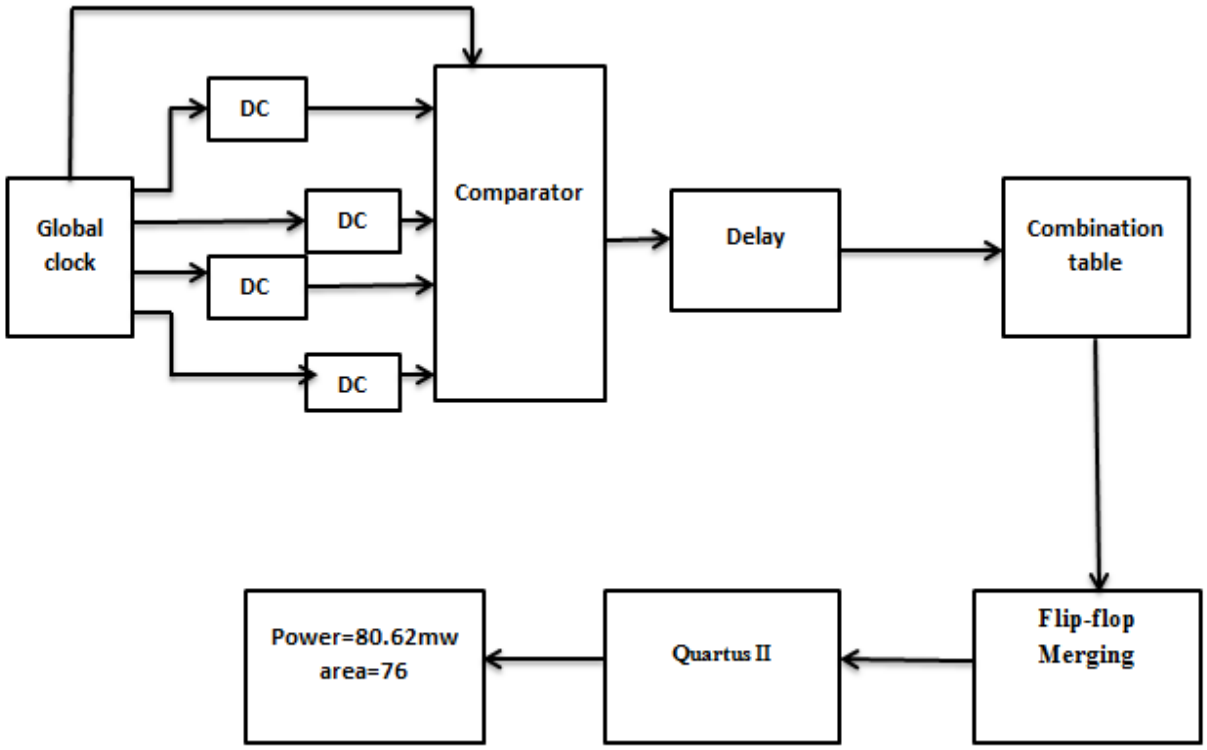


Fig 5.1 Block diagram of flip-flop after merging

Distributed clock signal are given to 50 flip-flips using a single global clock. Delay of each clock is found by comparing the distributed and global clock signal. In same way the delays are found out for the 50 flip-flops. The flip-flop having same delay are combined and a combination table is built and the flip-flops are merged. Then this merged flip-flop is given to a software named quartus II and from this power and area are found.

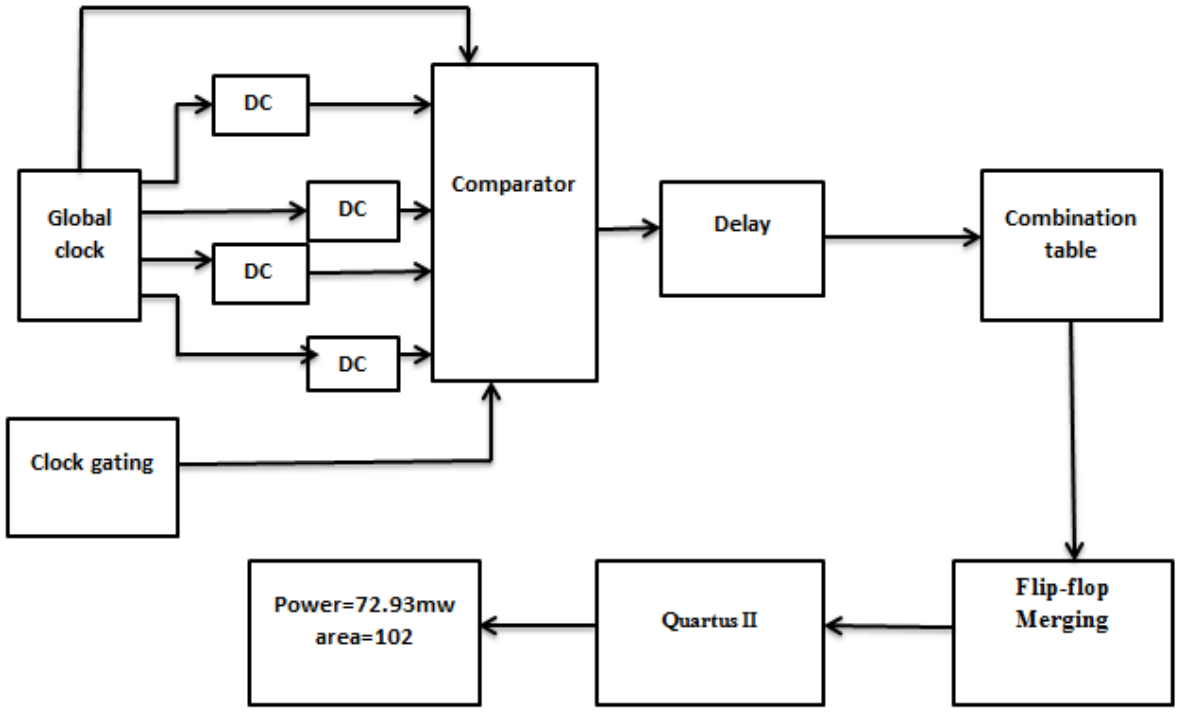


Fig 5.2 Block diagram of clock gating

If there is a change in the given input data, then the clock signal should be given. If there is no change in input data, there is no need to provide the clock signal. Clock gating means to mask the unwanted signal transitions from propagating forward. The same idea has been applied to clock signal. When the clock signal of a functional module is not required for some extended period, we use a gating function to turn off the clock feeding the module. So whenever the clock signal is not needed, unwanted switching activities will be suppressed and hence dynamic power reduces.

CHAPTER 6

SOFTWARE DESCRIPTION

6.1 SOFTWARES USED

The software used for this project is Modelsim and Quartus II 9.0 web edition and the language used is Verilog.

6.1.1 VERILOG

Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits.

Verilog was started initially as a proprietary hardware modeling language by Gateway Design Automation Inc. around 1984. Verilog simulator was first used beginning in 1985 and was extended substantially through 1987. The implementation was the Verilog simulator sold by Gateway. The first major extension was Verilog-XL, which added a few features and implemented the infamous "XL algorithm" which was a very efficient method for doing gate-level simulation.

Verilog can be used to describe designs at four levels of abstraction:

- Algorithmic level (much like c code with if, case and loop statements).
- Register transfer level (RTL uses registers connected by Boolean equations).
- Gate level (interconnected AND, NOR etc.).
- Switch level (the switches are MOS transistors inside gates).

6.1.2 MODELSIM

ModelSim is a verification and simulation tool for VHDL, Verilog, System Verilog, and mixed language designs. It is divided into four topics, which are described in brief.

- Basic simulation flow
- Project flow

- Multiple library flow
- Debugging tools

6.1.2.1 BASIC SIMULATION FLOW

The following diagram shows the basic steps for simulating a design in ModelSim.

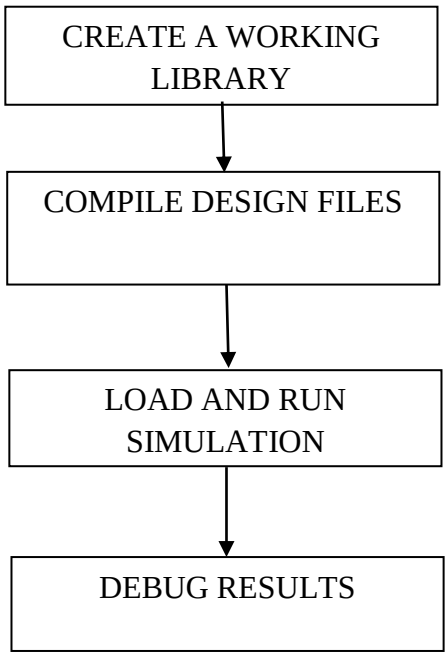


Fig 6.1 Basic Step for Simulation

- Creating the Working Library

  In ModelSim, all designs are compiled into a library. Typically to start a new Simulation in ModelSim by creating a working library called "work," which is the default library name used by the compiler as the default destination for compiled design units.

- Compiling the Design

  After creating the working library, the design is compiled. The ModelSim Library format is compatible across all supported platforms. The users can simulation the design on any platform without having to recompile the design.

- Loading the Simulator with the Design and Running the Simulation

With the design compiled, load the simulator with the design by invoking the simulator on a top-level module (Verilog) or a configuration or entity/architecture pair (VHDL).Assuming the design loads successfully, the simulation time is set to zero, and enter a run command to begin simulation.

- Debugging the Results

  If the users didn't get the expected results, the user can use ModelSim's robust debugging environment to track down the cause of the problem.

6.1.2.2 PROJECT FLOW

A project is a collection mechanism for an HDL design under specification or test. Even though the users don't have to use projects in ModelSim, they may ease interaction with the tool and are useful for organizing files and specifying simulation settings.

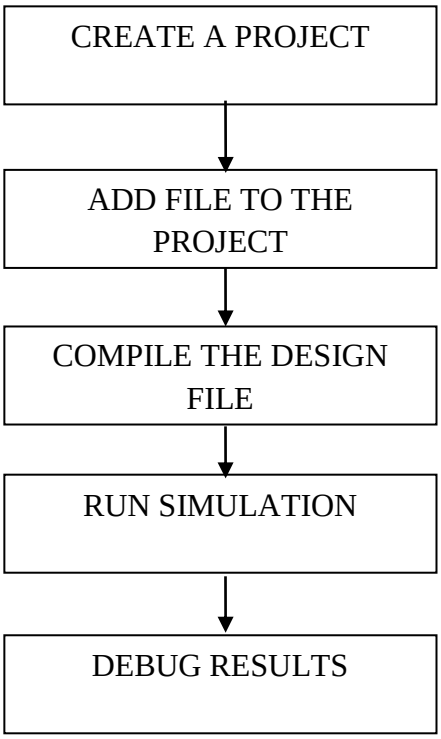


Fig 6.2 Steps for Simulation in ModelSim

The above diagram shows the basic steps for simulating a design within a ModelSim project. It is seen that the flow is similar to the basic simulation flow. However, there are two important differences:

- The user need not have to create a working library in the project flow; it is done automatically.
- Projects are persistent. In other words, they will open every time to invoke ModelSim unless it is specifically close them.

6.1.3 QUARTUS II

Quartus II is a software tool produced by Altera for analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

- Design Entry – the desired circuit is specified either by means of a schematic diagram, or by using a hardware description language, such as Verilog or VHDL
- Synthesis – the entered design is synthesized into a circuit that consists of the logic elements (LEs) provided in the FPGA chip
- Functional Simulation – the synthesized circuit is tested to verify its functional correctness; this simulation does not take into account any timing issues
- Fitting – the CAD Fitter tool determines the placement of the LEs defined in the netlist into the LEs in an actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs
- Timing Analysis – propagation delays along the various paths in the fitted circuit are analyzed to provide an indication of the expected performance of the circuit
- Timing Simulation – the fitted circuit is tested to verify both its functional correctness and timing
- Programming and Configuration – the designed circuit is implemented in a physical FPGA chip by programming the configuration switches that configure the LEs and establish the required wiring connections.

Quartus II software includes a simulator which can be used to simulate the behavior and performance of circuits designed for implementation in Altera's programmable logic devices. The simulator allows the user to apply test vectors as inputs to the designed circuit and to observe the outputs generated in response. In addition to being able to observe the simulated values on the I/O pins of the circuit, it is also possible to probe the internal nodes in

the circuit. The simulator makes use of the Waveform Editor, which makes it easy to represent the desired signals as waveforms. Doing this tutorial, the reader will learn about:

- Test vectors needed to test the designed circuit
- Using the Quartus II Waveform Editor to draw the test vectors
- Functional simulation, which is used to verify the functional correctness of a synthesized circuit
- Timing simulation, which takes into account propagation delays due to logic elements and interconnecting wiring

BOARD	**DEVICE NAME**
DE0	Cyclone III EP3C16F484C6
DE1	Cyclone II EP2C20F484C7
DE2	Cyclone II EP2C35F672C6
DE2-70	Cyclone II EP2C70F896C6
DE2-115	Cyclone IVE EP4CE115F29C7

Table 6.1 Types of Device

CHAPTER 7

SIMULATION RESULTS

The results of flip flop merging and clock gating are presented in this chapter using the software MODELSIM and QUARTUS II.

7.1 POWER REDUCTION RATIO

Power reduction ratio is calculated based on the following equations,

$$PR\_Ratio(\%) = \frac{power_{original} - power_{merged}}{power_{original}} \cdot 100\%$$

where the powermerged are the measured power after the program is applied, and the poweroriginal are the measured power of the original test case.

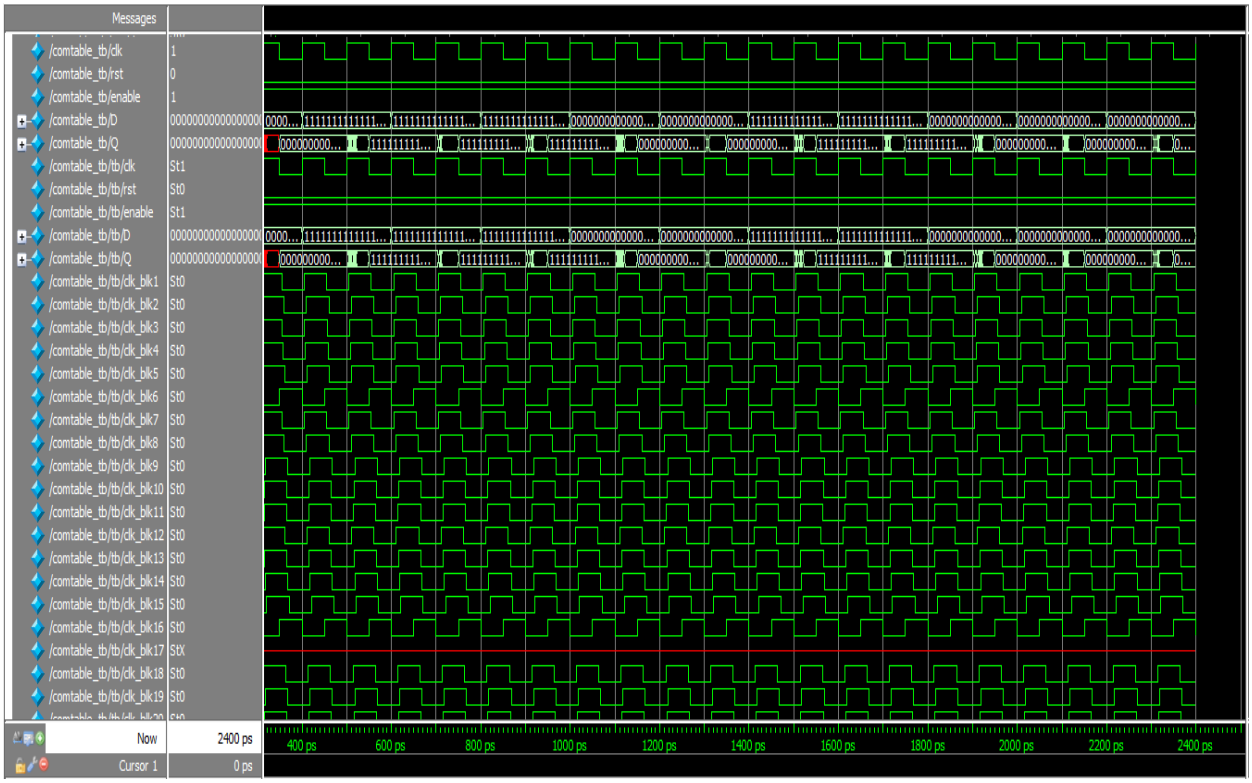
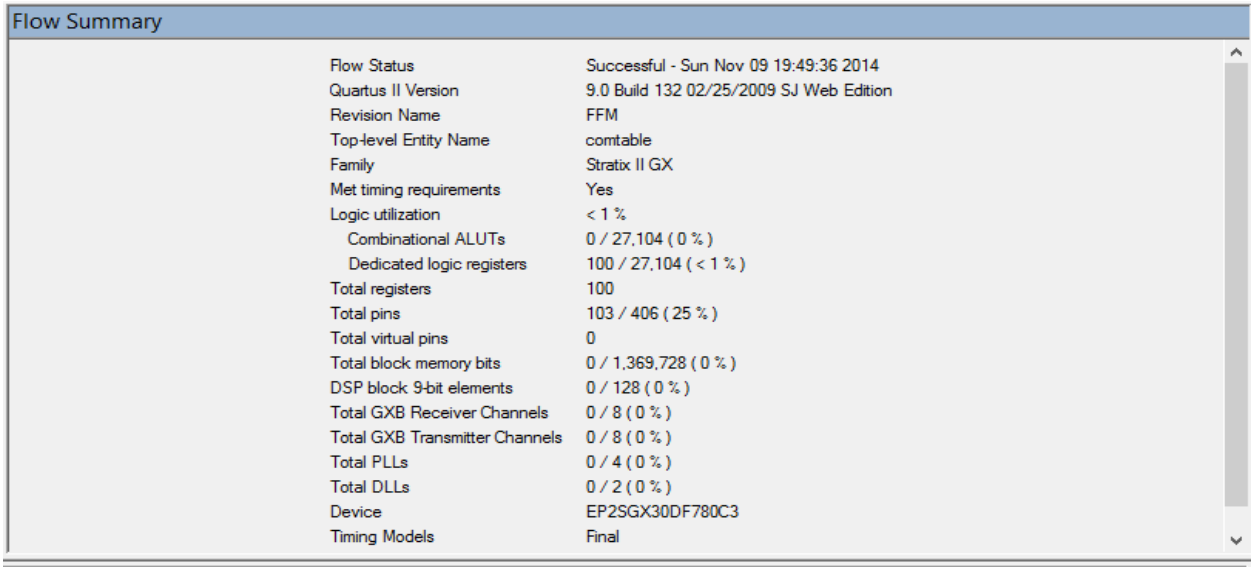


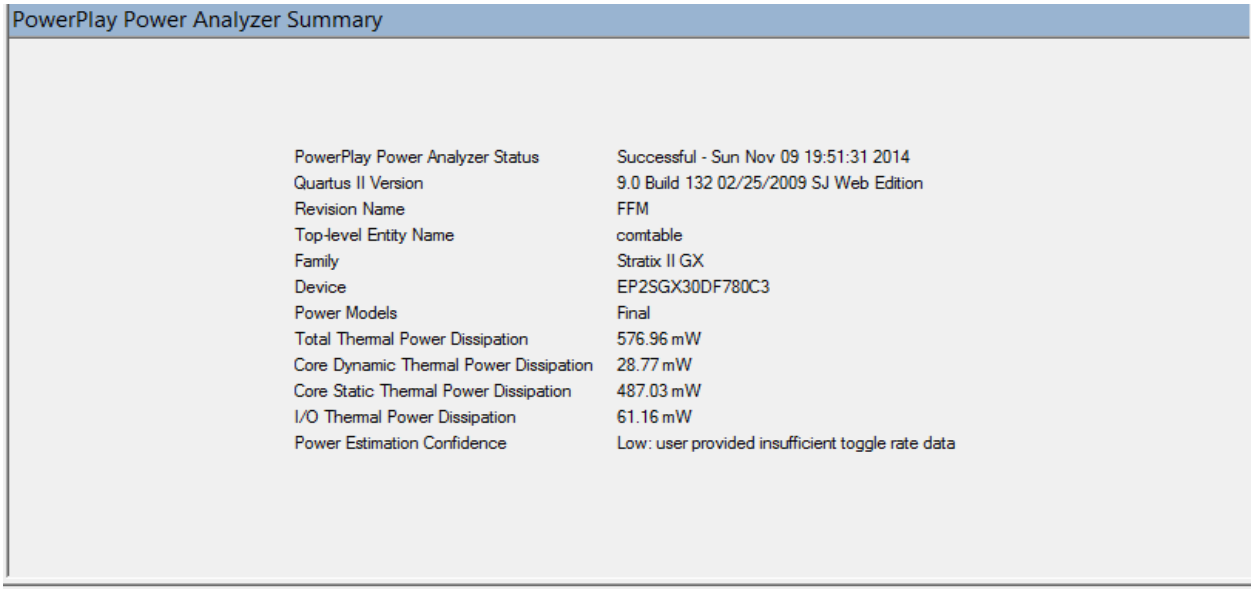
Fig 7.1 Simulation result for flip-flop before merging



Flow Summary

Flow Status	Successful - Sun Nov 09 19:49:36 2014
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	FFM
Top-level Entity Name	comtable
Family	Stratix II GX
Met timing requirements	Yes
Logic utilization	< 1 %
Combinational ALUTs	0 / 27,104 (0 %)
Dedicated logic registers	100 / 27,104 (< 1 %)
Total registers	100
Total pins	103 / 406 (25 %)
Total virtual pins	0
Total block memory bits	0 / 1,369,728 (0 %)
DSP block 9-bit elements	0 / 128 (0 %)
Total GXB Receiver Channels	0 / 8 (0 %)
Total GXB Transmitter Channels	0 / 8 (0 %)
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2SGX30DF780C3
Timing Models	Final

Fig 7.2 Total area estimation for flip-flop before merging



PowerPlay Power Analyzer Summary

PowerPlay Power Analyzer Status	Successful - Sun Nov 09 19:51:31 2014
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	FFM
Top-level Entity Name	comtable
Family	Stratix II GX
Device	EP2SGX30DF780C3
Power Models	Final
Total Thermal Power Dissipation	576.96 mW
Core Dynamic Thermal Power Dissipation	28.77 mW
Core Static Thermal Power Dissipation	487.03 mW
I/O Thermal Power Dissipation	61.16 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig 7.3 Total power estimation for flip-flop before merging

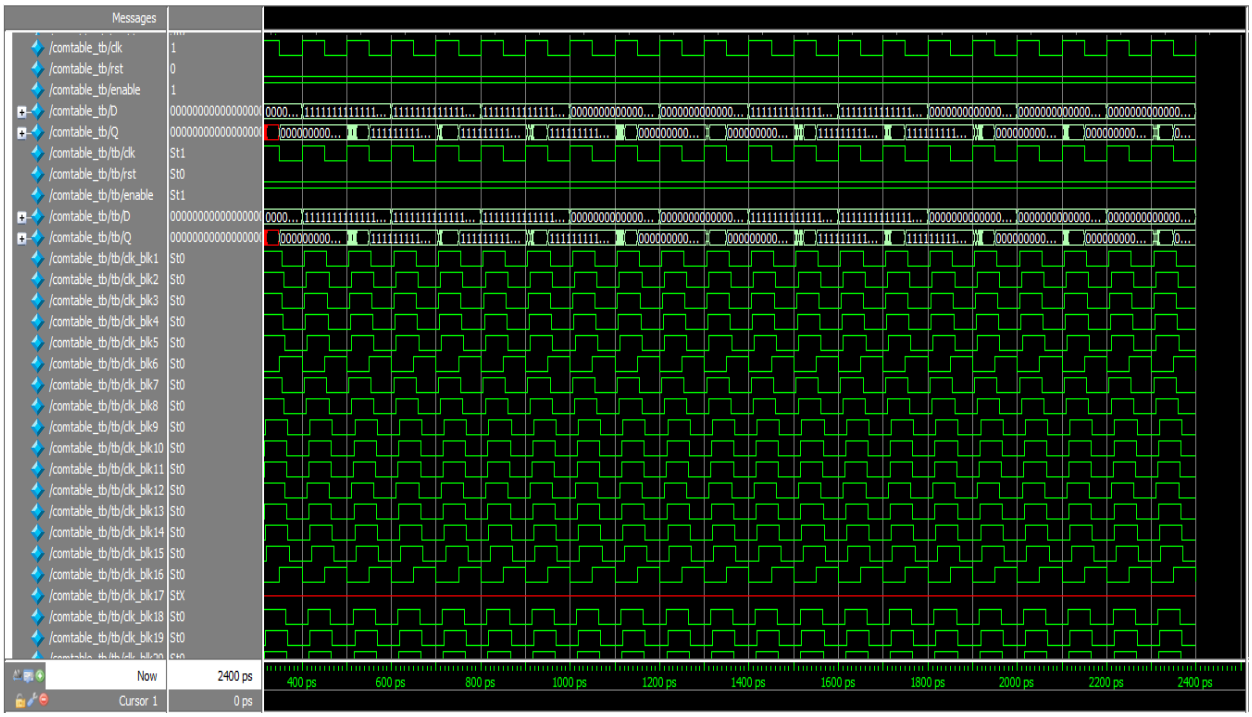


Fig 7.4 Simulation result for flip-flop after merging

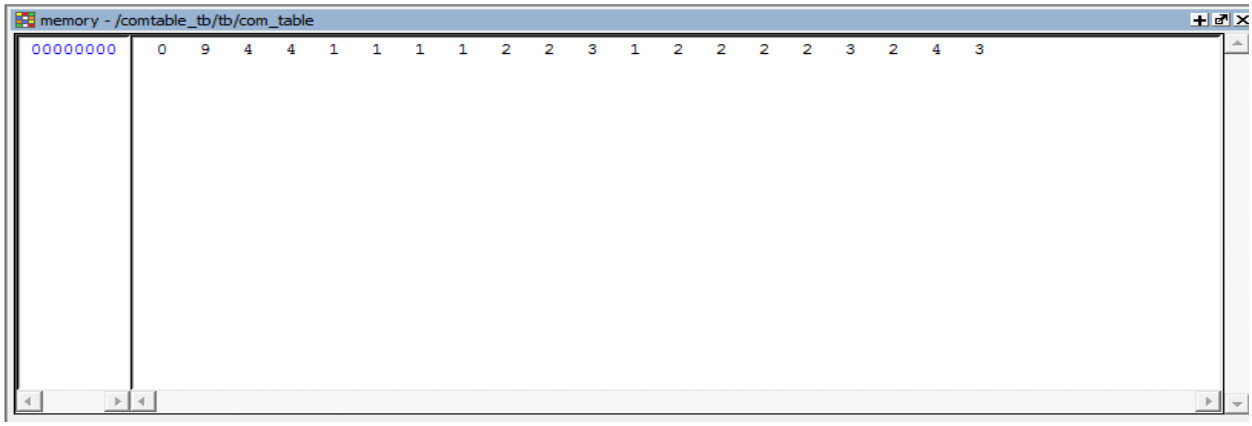
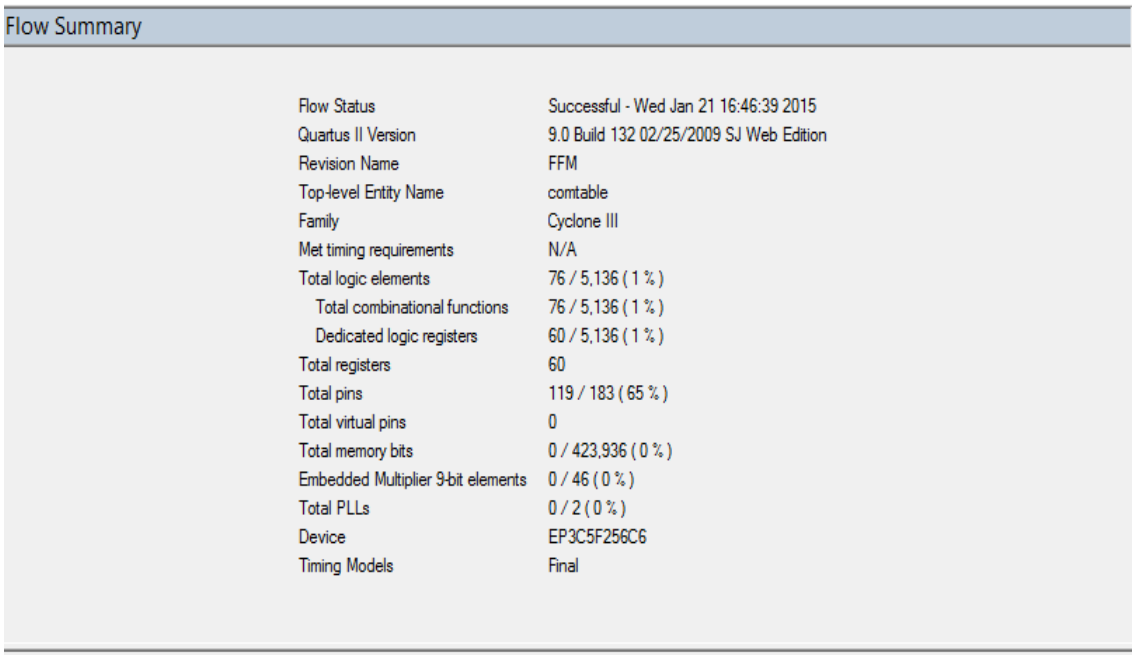


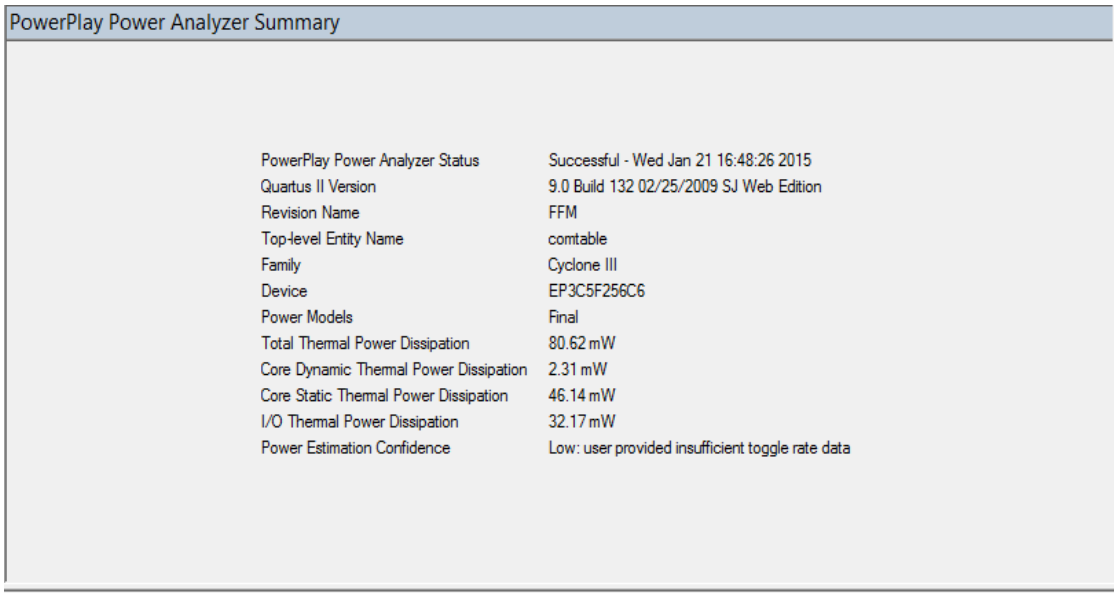
Fig 7.5 Simulation result for combination table to merging flip-flop



Flow Summary

Flow Status	Successful - Wed Jan 21 16:46:39 2015
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	FFM
Top-level Entity Name	comtable
Family	Cyclone III
Met timing requirements	N/A
Total logic elements	76 / 5,136 (1 %)
Total combinational functions	76 / 5,136 (1 %)
Dedicated logic registers	60 / 5,136 (1 %)
Total registers	60
Total pins	119 / 183 (65 %)
Total virtual pins	0
Total memory bits	0 / 423,936 (0 %)
Embedded Multiplier 9-bit elements	0 / 46 (0 %)
Total PLLs	0 / 2 (0 %)
Device	EP3C5F256C6
Timing Models	Final

Fig 7.6 Total area estimaion for flip-flop after merging



PowerPlay Power Analyzer Summary

PowerPlay Power Analyzer Status	Successful - Wed Jan 21 16:48:26 2015
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	FFM
Top-level Entity Name	comtable
Family	Cyclone III
Device	EP3C5F256C6
Power Models	Final
Total Thermal Power Dissipation	80.62 mW
Core Dynamic Thermal Power Dissipation	2.31 mW
Core Static Thermal Power Dissipation	46.14 mW
I/O Thermal Power Dissipation	32.17 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig 7.7 Total power estimaion for flip-flop after merging

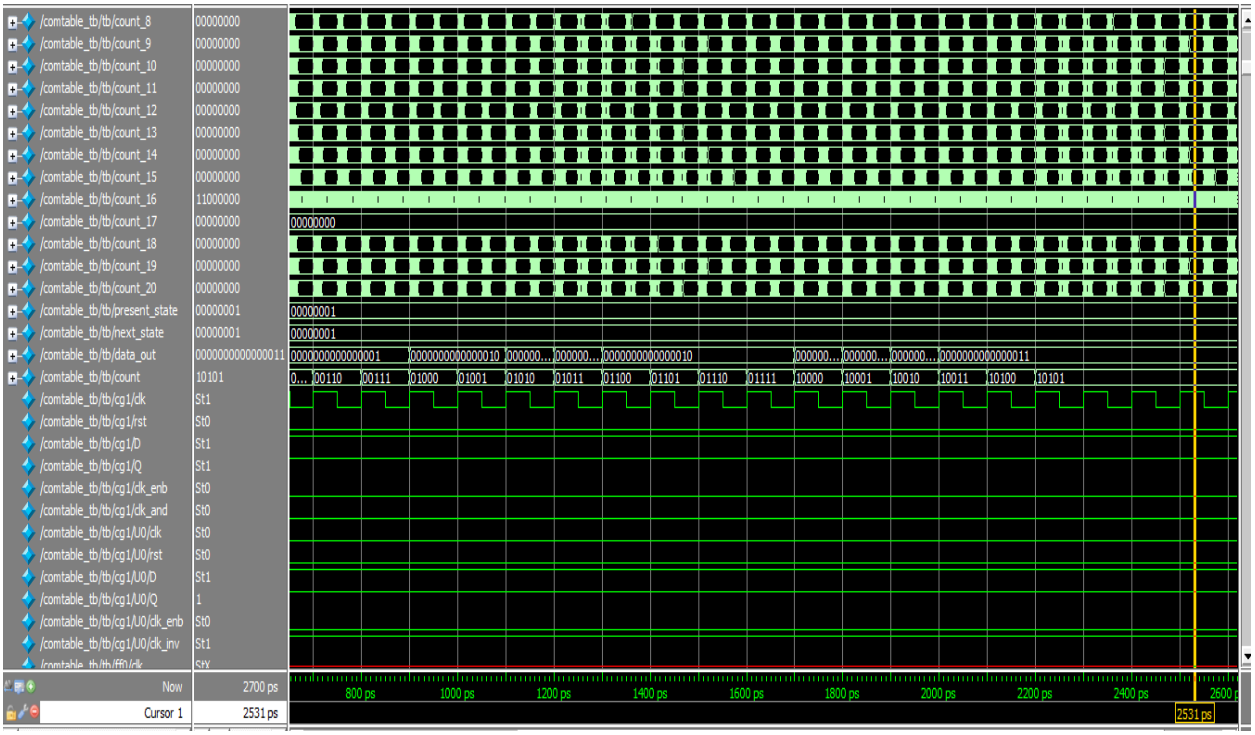


Fig 7.8 Simulation result for clock gating

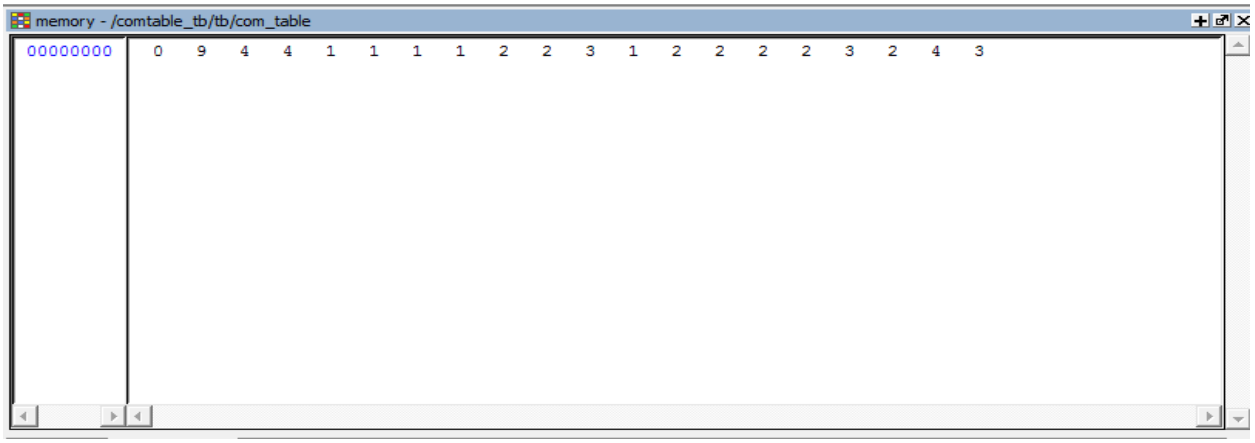
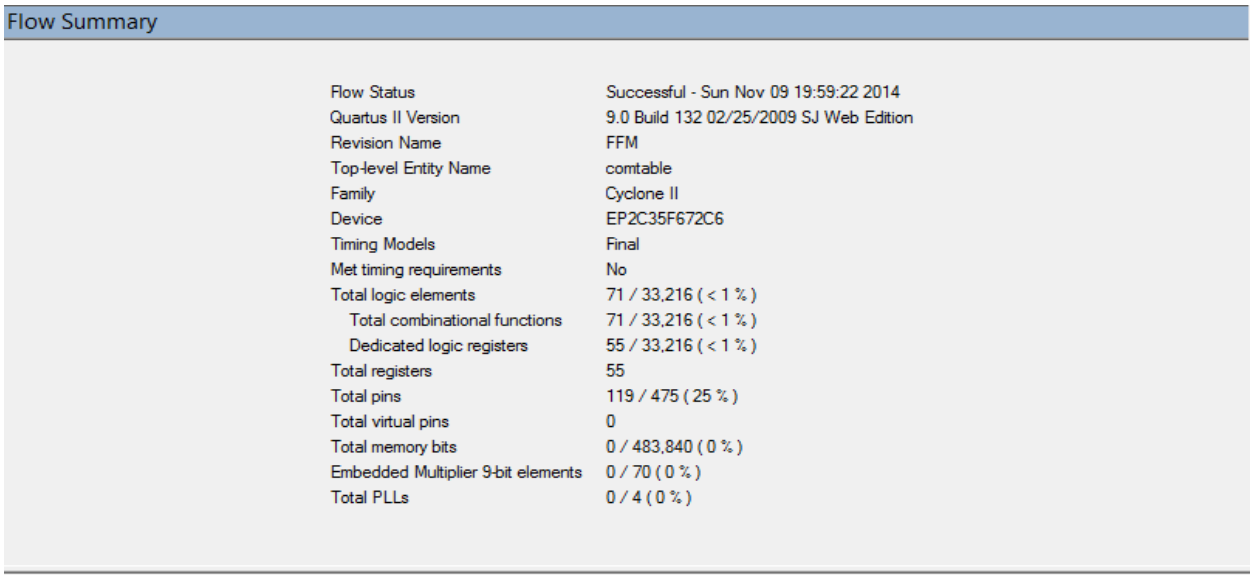


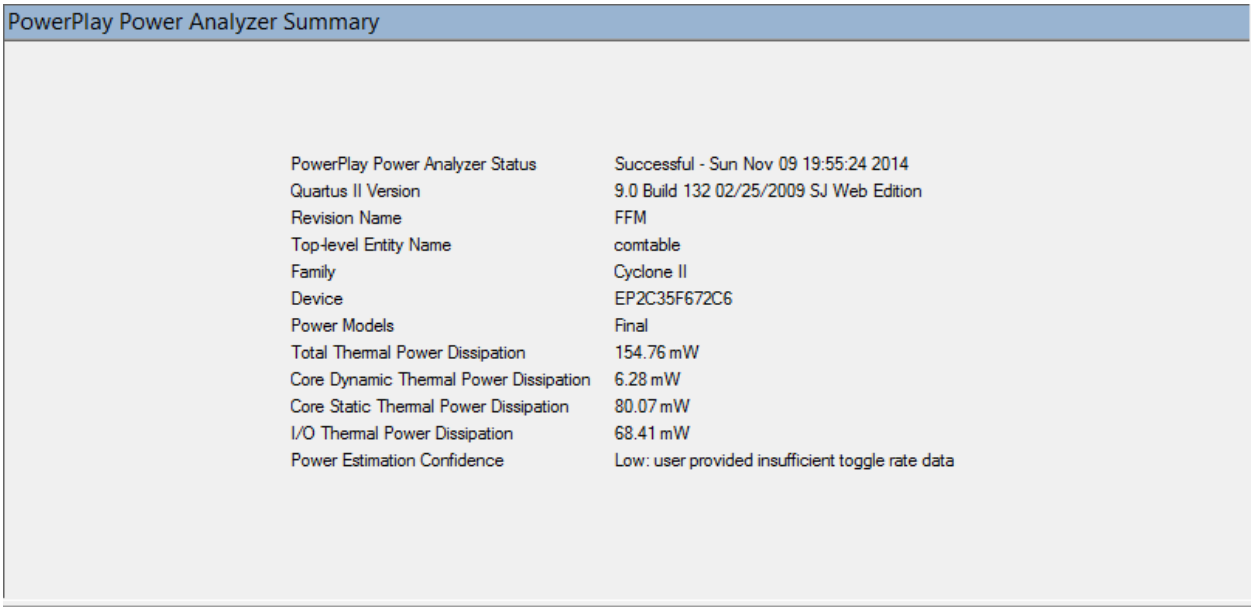
Fig 7.9 Simulation result for combination table to merging flip-flop



Flow Summary

Flow Status	Successful - Sun Nov 09 19:59:22 2014
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	FFM
Top-level Entity Name	comtable
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	No
Total logic elements	71 / 33,216 (< 1 %)
Total combinational functions	71 / 33,216 (< 1 %)
Dedicated logic registers	55 / 33,216 (< 1 %)
Total registers	55
Total pins	119 / 475 (25 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Fig 7.10 Total area estimaion for clock gating



PowerPlay Power Analyzer Summary

PowerPlay Power Analyzer Status	Successful - Sun Nov 09 19:55:24 2014
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	FFM
Top-level Entity Name	comtable
Family	Cyclone II
Device	EP2C35F672C6
Power Models	Final
Total Thermal Power Dissipation	154.76 mW
Core Dynamic Thermal Power Dissipation	6.28 mW
Core Static Thermal Power Dissipation	80.07 mW
I/O Thermal Power Dissipation	68.41 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig 7.11 Total power estimaion for clock gating

Parameter	Flip-flop number	Before merging flip-flop	After merging flip-flop	Clock gating
Power(mw)	50	116.97	80.62	72.93
Area	50	150	76	102

Table 7.1-Comparison Table for power and area estimation

CHAPTER 8

CONCLUSION

As the number of flip-flops in a chip increases dramatically, the complexity would increase exponentially, which makes the method impractical. To handle this problem more efficiently and get better results, the following approaches are used.1)To facilitate the identification of mergeable flip-flops, we transform the coordinate system of cells. In this way the memory used to record the feasible placement region can also be reduced.2) To avoid wasting time in finding impossible combinations of flip-flops, we first build a combination table before actually merging two flip-flops. 3) Partition a chip into several sub-regions and perform replacement in each sub-region to reduce the complexity. However this method may degrade the solution quality. To resolve the problem we used a hierarchical way to enhance the result and processing time can be reduced. The flip flop merging and clock gating technique is used to reduce the dynamic power and area. The simulation is done using ModelSim software and the power analysis through Quartus II. From the power analysis, 37.65% of power has been consumed before merging the flip flop and after merging the flip flop with clock gating technology.

REFERENCES

[1] P. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R.L.Allmon, "Post placement power optimization with Multi Bit Flip- Flop," IEEE J. Solid-State Circuits, vol. 33, no. 5, pp. 676–686, May 2012.

[2] D. Duarte, V. Narayanan, and M. J. Irwin, "Power aware placement," in Proc. IEEE VLSI Comput. Soc. Annu. Symp., Pittsburgh, PA, Apr. 2005, pp. 52–57.

[3] H. Kawagachi and T. Sakurai, "Impact of technology scaling in the clock power," in VLSI Circuits Dig. Tech. Papers Symp., Jun. 2003, pp. 97–98.

[4] W. Hou, D. Liu, and P.-H. Ho, "Automatic register banking for low power clock trees," in Proc. Quality Electron. Design, San Jose, Mar. 2010, pp. 647–652

[5] Y. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang, "High performance microprocessor design," in Proc. Design Autom. Conf., Jun. `1998, pp. 795–800.

[6] W. Aloisi and R. Mita, "Gated-clock design of linear-feedback shift registers," IEEE Trans. Circuits Syst., II, Brief Papers, vol. 55, no. 5, pp. 546–550, Jun. 2008.

[7] Jhen-Hong He1, Li-Wei Huang2, Jui-Hung Hung3, Yu-Cheng Lin4, Guo-syuan Liou5,Tsai-Ming Hsieh "Clock Network Power Saving Using Multi-Bit Flip-Flops in Multiple Voltage Island Design" Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013).

[8] M. Donno, E. Macii, and L. Mazzoni, "Power-aware clock tree planning," in Proc. Int. Symp. Phys. Design, 2004, pp. 138–147.

[9] S. Wimer and I. Koren, "The Optimal fan-out of clock network for power minimization by adaptive gating," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 10, pp. 1772–1780, Oct. 2012.

[10] Hosain.R, L. D. Wronshi, and albicki.A, 1994. "Low power design using double edge triggered flip-flop," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 2, no. 2, pp. 261–265.

[11] C. Chunhong, K. Changjun, and M. Sarrafzadeh, "Activity-sensitive clock tree construction for low power," in Proc. Int. Symp. Low Power Electron. Design, 2002, pp. 279–282.

[12] W. Shen, Y. Cai, X. Hong, and J. Hu, “Activity and register placement aware gated clock network design,” in Proc. Int. Symp. Phys. Design, 2008, pp. 182–189.

[13] Y.-T. Chang, C.-C. Hsu, P.-H. Lin, Y.-W. Tsai, and S.-F. Chen, “Post-placement power optimization with multi-bit flip-flops,” in Proc. IEEE/ACM Comput.-Aided Design Int. Conf., San Jose, CA, Nov. 2010, pp. 218–223.

[14]Faraday Technology Corporation [Online]. Available: http://www. Faraday tech.com/index.html

[15] C. Bron and J. Kerbosch, “Algorithm 457: Finding all cliques of an undirected graph,” ACM Commun., vol. 16, no. 9, pp. 575–577, 1973.

[16] CAD Contest of Taiwan [Online]. Available: http://cad_contest.cs.nctu.edu.tw/cad11

LIST OF PUBLICATIONS

Conferences

- Presented a paper titled “Power Reduction By Flip Flop Merging Technique Using Heuristic Algorithm”**,** National Conference on Emerging Trends In Computer Communication and Informatics (ETCCI'15), TamilNadu College of Engineering, Coimbatore, 18th Feb, 2015

- Presented a paper titled “Low Power Flip Flop Merging Technique by Critical Path Delay Analysis” 2nd International Conference on Electronics and Communication Systems (ICECS), Karpagam College of Engineering, Coimbatore, 27th Feb, 2015