AUTOMATED VERIFICATION OF EXTRACTED TIMING MODELS

M.Tech. Thesis

By

UTKARSH SHARMA

DISCIPLINE OF VLSI DESIGN & NANOELECTRONICS

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JUNE 2020

AUTOMATED VERIFICATION OF EXTRACTED TIMING MODELS

A THESIS

Submitted in partial fulfillment of the requirements for the award of the degree
of
Master of Technology

by
UTKARSH SHARMA

DISCIPLINE OF VLSI DESIGN & NANOELECTRONICS
INDIAN INSTITUTE OF TECHNOLOGY INDORE
JUNE 2020

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **AUTOMATED VERIFICATION OF EXTRACTED TIMING MODELS** in the partial fulfillment of the requirements for the award of the degree of **MASTER OF TECHNOLOGY** and submitted in the **DISCIPLINE OF ELECTRICAL ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from July 2018 to June 2020 under the supervision of Dr. Santosh Kumar Vishvakarma, Associate Professor, Electrical Engineering, IIT Indore.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

UTKARSH SHARMA

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

Dr. SANTOSH KUMAR VISHVAKARMA
(Supervisor)

UTKARSH SHARMA has successfully given his M.Tech. Oral Examination held on **June 22, 2020**.

Signature(s) of Supervisor(s) of M.Tech. Thesis
Date: 22-06-2020

Convener, DPGC
Date: 22-06-2020

Signature of PSPC Member #1
Dr. Amod Umarikar
Date:22-06-2020

Signature of PSPC Member #2
Dr. Abhishek Srivastava
Date:22-06-2020

ACKNOWLEDGEMENTS

It is a matter of immense pleasure to present my M.Tech. Thesis on "AUTOMATED VERIFICATION OF EXTRACTED TIMING MODELS" for my work done during one year internship of M.Tech. (2018-2020), Indian Institute of Technology Indore at Seagate Technologies, Pune.

I wish to express my gratitude to the respected Supervisor, Dr. Santosh Kumar Vishvakarma, Associate Professor, Electrical Engineering Department, Indian Institute of Technology Indore for guidance and patient advice. I also wish to express my gratitude to respected Dr. Vipul Singh, Head of Department and respected Professor Neelesh Kumar Jain, Director of Indian Institute of Technology Indore for providing me possible facilities and support.

I am highly thankful to respected Dr. Abhishek Srivastava, Associate Professor, Computer Science Engineering Department, IIT Indore and respected Dr. Amod C. Umarikar, Associate Professor, Electrical Engineering Department, IIT Indore for encouragement and patient advice. I also wish to express our sincere thanks to respected Mr. Sachin Bastimane, Sr. Engineering Manager, Seagate Technologies and respected Mr. Vikrant Joshi, Sr. Engineering Manager, Seagate Technologies, for their support and motivation during my internship. Many thanks to Mr. Mayank Barsaiyan and Mr. Prashant Meshram for their support at Seagate. I also thank all my teachers at I.I.T. Indore for motivation and help.

I, Utkarsh Sharma want to express my hearties thank to my parents and all my family members whose blessings and affection have remained constant sources of inspiration. Without the help of all these people, it was not possible for me to complete this work.

Utkarsh Sharma
Roll No. 1802102021
M.Tech VLSI Design & Nanotechnology,
Indian Institute of Technology Indore

Abstract

IC design process consist of many steps other than designing such as logic synthesis, functional verification, placement, routing, etc before the design can be sent for fabrication. **Static Timing Analysis (STA)** plays a vital role in today's designing of commercial circuits. It needs to be performed repeatedly after multiple steps of IC design process. STA ensures that the chip being designed meets timing requirements. Static timing analysis is a method of figuring the expected timing of a digital circuit without demanding simulation of the full circuit.

Most commercial chip designing and implementation of IP models use **Extracted Timing Models (ETMs)** for faster overall process. Extracted Timing Models were created after performing Static Timing Analysis in this work. These models also need to be verified after their creation. These timing models are usually verified manually. This manual verification is very slow. To speed this up, Automated Verification of Extracted Timing Models is done in this work. This has found too significantly reduce the time of Verification of Extracted Timing Models.

The circuits designed for testing the ETM Verification method are Approximate MAC unit and a Memory Controller. MAC Units find vast applications in computing from Arithmetic Logic Units (ALUs) to Artificial Neural Networks. **Approximate MAC Unit** designed here is a MAC Unit with some approximations for faster working using less number of gates.

Another circuit designed for ETM Verification is a memory controller. A **Memory Controller** is a digital circuit that manages the movement of data to and from a memory. Designing semiconductor memories focuses on correct functionality of the circuit, make the most of circuit density, and engaging circuits such that clock and data signals are efficiently routed. These commercial semiconductor memories used today need many essential circuits to function properly like Memory Controllers. Memory Controller was designed using Verilog HDL during this work.

TABLE OF CONTENTS

Chapter 5: Synthesis of Gate-Level Netlist

5.1. HDL of Design

5.2. Design Constraints

5.3. Technology Library

Chapter 6: Extracted Timing Models

6.1. Steps for Creation of Extracting Timing Model

6.2. Specifying Parasitics Information

6.3. Introduction to Static Timing Analysis

6.4. Modes of Static Timing Analysis

6.5. Performing Static Timing Analysis

6.6. Path Reports

6.7. Fixing Timing Violations

6.8. Verification of Extracted Timing Model

Chapter 7: Results and Discussion

7.1. Simulation and Functional Verification of Approximate Multiplier

LIST OF FIGURES

Chapter 1

Introduction

Measuring the ability of a circuit is to meet timing constraints and work fast needs a dexterity to quantify delays and check timing violations during the design process, at a number of steps. Also, the delay computation should be included into the interior loop of timing paths at different phases of design, like logic synthesis, placement and routing, etc. Such timing quantifications can be made theoretically by means of a laborious circuit simulation, but approaches like this are not practical because commercial circuits consist of a large number of cells. Static Timing Analysis (STA) and Extracted Timing Models play a very important role for facilitating the accurate and fast measurement of timing of circuit. The pacing comes due to the use of timing models and by discounting logical functionality in design.

Often a large circuit is designed using integration of internal blocks. These blocks have a large number of cells that they are designed separately. **Static timing analysis (STA)** is a technique of checking timing constraints and fixing timing violations of a digital circuit without simulating the complete circuit, at different stages of IC design. Static Timing Analysis need to be done repeatedly and also after design integration. Thus complete Static Timing Analysis of huge designs may be too slow. To make this process fast, the block which have undergone STA and met timing specifications are replaced by their **Extracted Timing Models (ETMs)** after Static Timing Analysis of these smaller designs. Another major use of ETMs is in implementation of IP models. This requires verification of ETMs before they can be used for faster chip design process. This has become a pillar of circuit designing over the past few decades.

As the method of Verification of Extracted Timing Models must work for different circuits to be effective, two digital circuits were designed, a combinational circuit and a sequential circuit in this work.

In today's world of Artificial Intelligence, Artificial Neural Networks (ANN) play a vital role. Artificial Neural Networks (ANNs) use Multiplier-Accumulator Units for learning as well as in

prediction or classification. They are analogous to neural networks in brain**.** The basic unit of an ANN is a Multiplier–Accumulator (MAC) Unit which is like neurons in human brain. These units are basis of modelling complex patterns for prediction as well as classification problems.

Thus, the combinational circuit designed for use in ETM Verification is an Approximate Multiplier–Accumulator (MAC) Unit. Its main advantage is that it is faster than the conventional MAC Unit. This will lead to faster modelling if used in Artificial Neural Networks.

Another circuit used to test ETM Verification method is a Memory Controller. A memory controller is a digital circuit which bring about the read and write operations of data from and to a memory respectively. Most widely used Random Access Memories are Synchronous Dynamic Random Access Memories **(**SDRAMs), High Bandwidth Memories (HBMs) and Flash Memories. These memories need many essential circuits to function properly like **Memory Controllers**. Precisely, the sequential circuit designed is a Synchronous dynamic random-access memory **(**SDRAM) Controller, one the most commonly used memory controller in commercial applications.

Chapter 2

Review of Past Work and Problem Formulation

The basic functionality of widely used digital circuits like ALUs, Memories, ASICs and FPGAs is based on calculation of binary data. This calculation comprises of and multiplication additions of (signed/unsigned) data in 1s and 0s. Thus, MAC Units are building blocks of such circuits. They are used almost everywhere where there are digital circuits for computing. MAC Units are used for binary calculations in Arithmetic Logic Units. Neurons in any Artificial Neural Network also comprise of multiple MAC Units. The Approximate MAC Unit designed here aims to work faster, using less number of logic gates.

Most widely used memories used are Synchronous dynamic random-access memory **(**SDRAM), High Bandwidth Memory (HBM) and Flash Memory. Currently SDRAM is manufactured at technology lengths of 12nm, 10nm in different semiconductor industries. The ICs of these memories are enormously complicated. They also need Memory Controllers, Precharge circuits, etc. A normal desktop processor chip today has over 1 billion transistors. In short, the design of an IC spending EDA Softwares is the design, test, and verification of the instructions that the IC is to execute.

A typical IC design cycle involves following major steps:

- **System Specification:** The procedure of circuit design starts with the specifying the functionality that the complete design must provide, but does not designate by what means it shall be attained. The early specification is essentially a detailed technically description of what the customer needs in the completed circuit to accomplish. It can include a range of electrical necessities, such as what signals the circuit will input, what signals it must output, what power supplies are accessible.

- **Micro-Architecture of the Design:** This defines the fundamental arrangement, goals and principles of the product. It defines high level concepts and the inherent value plan of the product. Architecture groups take in account many variables and interface with groups.

- **RTL (Register Transfer Level) Design:** The circuit of which architecture has been built desires to be designed using Hardware Description Language. A hardware description language is a dedicated computer language used to elaborate the construction and behavior of electronic circuits. Commercial tools generally used for this purpose are Xilinx Vivado, Altera Modelsim, etc. Most common languages used are Verilog HDL, System Verilog and VHDL.

- **Functional Verification:** The working of the RTL Design has to be verified afore synthesizing its Gate-Level Netlist.

- **Logic Synthesis:** The RTL is charted into a gate-level netlist in the aimed technology of the integrated circuit.

- **FloorPlanning:** It is the organization of logical blocks (eg. Flip Flops) on silicon chip.

- **Placement and Routing:** Placement of the cells is usually done grounded on the timing/area requirements.

- **Static Timing Analysis:** Static timing analysis is a manner of verifying the Signal-integrity of a design by examination timing violations in all paths possible considered to be operating in worst conditions.

- Once the chip design is ready to go to the Fabrication unit, the design files are sent to the fabrication lab. This file is mostly a GDS II file.

  GDSII stream format is a database file format used in productions for data exchange of IC or artwork of IC layout. It is a binary file format expressing planar geometric forms, text labels, and additional information about the layout of design in hierarchical form. GDSII stands for Geometrical Database Standard for Information Interchange. It is commonly known as **GDSII** in short form.

Most common designs contains numerous logical cells thus have huge number of timing paths. Before sending the design file for fabrication, it has to be ensured that the circuit designed meets timing constraints and has no timing violations. This measurement is done using **Static Timing Analysis (STA)**. Please note that STA is mostly performed after multiple stages like Logic Synthesis, Placement, Routing, etc. Large designs many times consist of multiple blocks, each of which are smaller designs. Performing Static Timing Analysis of complete each internal block when the design is integrated is not a good practice. Thus Extracted Timing Models of blocks (to be used in design) can be used while performing Static Timing Analysis of larger designs.

An **Extracted Timing Model (ETM)** is a representation of a block through means of several timing arcs. In a design after a block satisfies its timing constraints at the gate level, the comprehensive internal timing of the block is not needed in timing analysis of the chip. A model containing timing arcs for block interfaces is used instead.

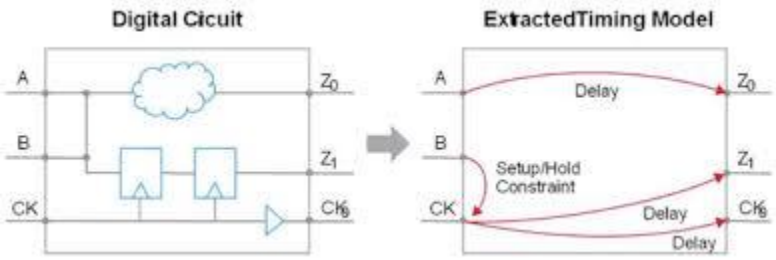


Fig.2.1. A Circuit and its Extracted Timing Model Representation

ETMs also provide IP salvage with the content protected since the model contains timing info in a hidden manner, deprived of any Netlist information.

The Extracted Timing Model needs to be verified before it can be used. This verification is done manually, which takes a lot of time for commercial designs. The ETM Verification method in this work also aims to reduce time in IC design process. The Automated Verification should also be independent of the circuit design.

Advantages of Using Extracted Timing models:

i. Reduced Time to Sign-Off:

Often a large design constitutes of a small block. With help of ETM of small design, Timing Analysis of small design is not required to be performed again when performing Static Timing Analysis of the large design. Thus, it significantly reduces the time required to analyze a large designs. Hence also reducing time to Timing Sign-Off the large design for fabrication.

ii. Abstraction:

A timing model in place of a netlist is used to hide the contents of the block from users.

Chapter 3

Approximate MAC Unit

3.1. Need for Approximate MAC Unit

In computing, the multiply–accumulate operation is the multiplication of two quantities and supplements that product to third quantity which may be the quantity stored in accumulator itself. The hardware unit that accomplishes the operation is recognized as a multiplier–accumulator (MAC, or MAC unit). The operation of multiplication followed by addition is called a MAC or a MAC operation. Thus a MAC Unit consists of a binary multiplier and a binary adder.

A binary multiplier is a microelectronic circuit used in applications, like an Arithmetic Logic Unit. It comprises of binary adders. Conservative Multiplier works on heart of computing a group of partial products, shifting them to the left and then adding them together. It is the conventional multiplication method. This method is used by CPU such that it performs the multiplication with the shift and add operations of its arithmetic logic unit instead of a dedicated circuit in multiplication part of multiplier-adder operation.

A major drawback of conventional MAC unit is that it is slow, because it involves a large number of intermediary additions. These additions consume a lot of time.

The Approximate MAC designed here performs much faster using Shifting and Approximation in multiplication and Approximation in addition, giving approximately same results. After designing it will be used in Automated Extracted Timing Model Verification.

3.2. Design of Approximate MAC Unit:

The Approximate Multplier-Adder designed in this work is of 16 bits. The Approximate MAC designed here performs much faster using Shifting and Approximation in multiplication and Approximation in addition, giving approximately same results.

The multiplier in MAC unit works based on logical equivalence that multiplication of binary numbers is equivalent to their shifting and addition. The additions in MAC unit are based on approximate adders. The algorithm of the 16 bit Approximate MAC Unit is shown below. It is used in designing the 16x16 Approximate MAC Unit in Verilog HDL.

Pins and Ports in Approximate MAC:

The 16 bit approximate MAC Unit consists of three input ports:

- port a: A16 bit input port to which one of the multiplicands is fed.
- port b: Another16 bit input port to which represents the multiplier.
- port k: it is 16 bit input port to which the input that is to be added to resultant of multiplication of port a and port b is fed.
- port f: : it represents the 32 bit output of the multiplication - addition operation.
- port f16: it represents the 16 bit approximated output of the multiplication - addition operation.

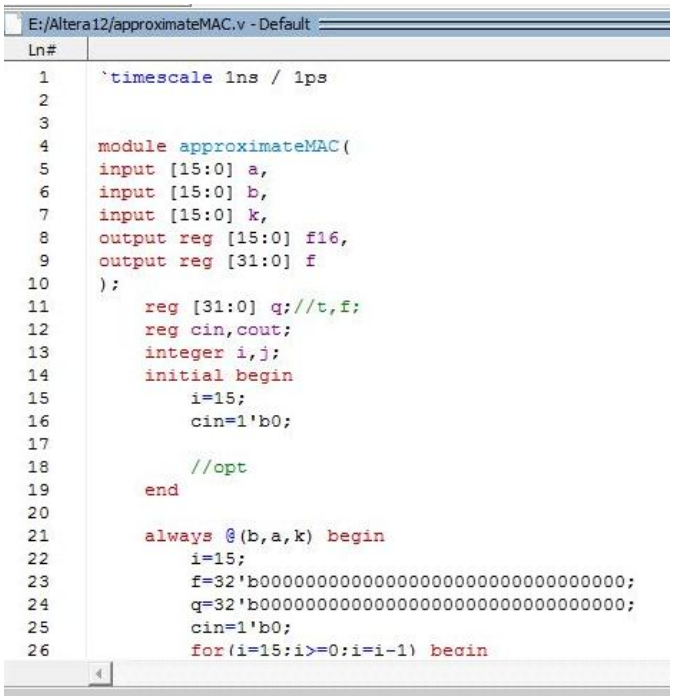


Fig. 3.1. Verilog HDL code screenshot of Multiplier-Accumulator Unit designed

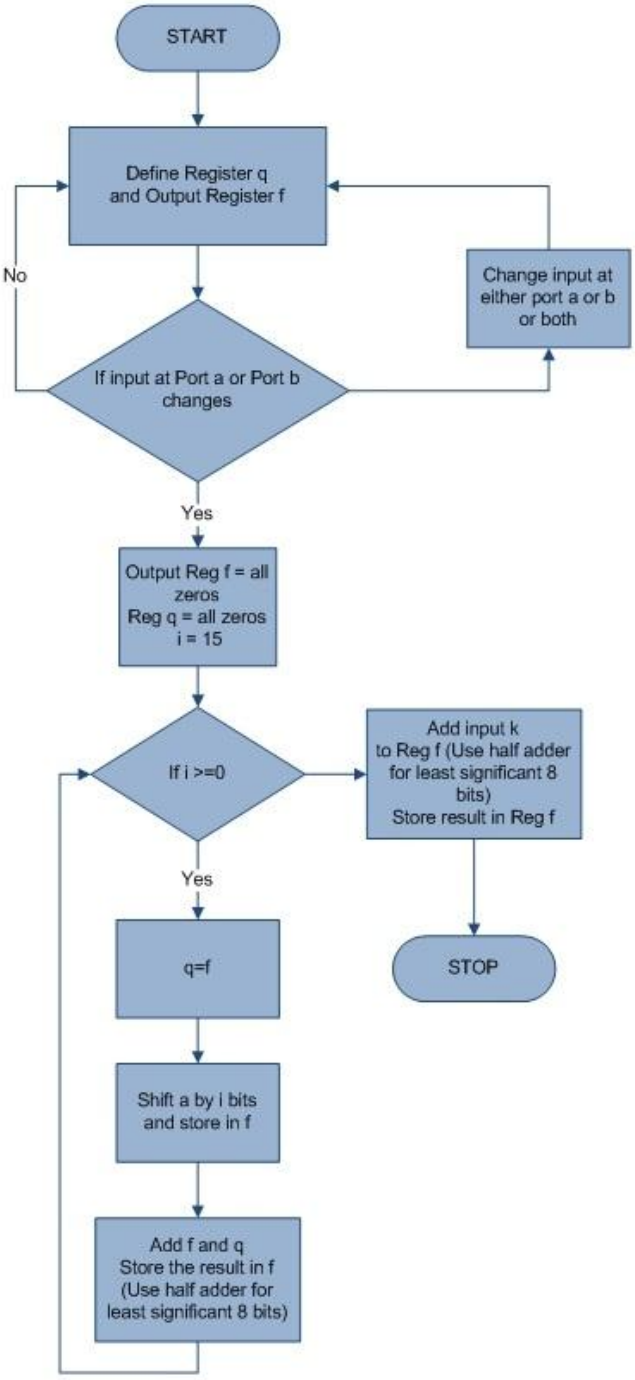


Fig.3.2 Flow Chart of the designed Approximate MAC Unit

The flowchart of the Approximate MAC Unit designed is in Fig.3.1. Its simulation and verification is in Chapter Results. Earlier this design was intended to function only as a less resource using multiplier based on shifting operation. Later the multiplier operation was used in designing a MAC Unit along with approximate computations. The RTL netlist of the design was made using Verilog HDL language on Altera Modelsim. Its simulation and verification was also performed on Altera Modelsim.

3.3. Applications of Approximate MAC Unit:

i. Artificial Neural Networks: MAC units are basic building blocks in artificial neural networks. The operation below in a neuron of a neural network is a multiplier-addition operation. This working of the Artificial Neural Network (implemented on FPGA, etc.) can be much faster if Approximate MAC units are used. With ref to Fig.2 working of a neuron in an ANN is:

   Let p = all zeros in binary

   $t = w_i * x_i + b$

   $t = w_i * x_i + a$

   $t = w_i * x_i + a$

   Output of neuron $y = f(t)$,

   Where f is activation function, x_i is input, b is the bias and w_i is weight of dendrite

   Thus, an ANN neuron consists of multiple MAC Units if implemented in terms of digital design.

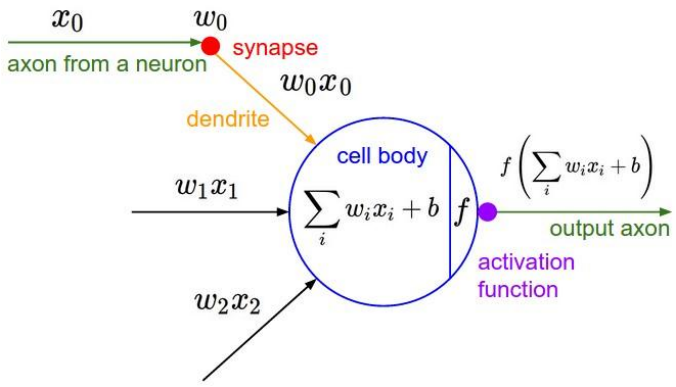


Fig.3.3 Multiplication-Accumulation MAC Unit as a Neuron

ii. Computing in Arithmetic Logic unit: MAC units are extensively used in ALU. Using Approximate MAC Units designed here can significantly increase speed of ALU operations.

Chapter 4

Memory Controller

4.1. Synchronous DRAM

A memory in which data at can be written or read in same time regardless of physical location of data is Random Access Memory (RAM). Dynamic RAM (DRAM) is a type of random access semiconductor memory that provisions each data bit in a memory cell. The memory cell consists of a capacitor and a FET, both typically based on metal-oxide-semiconductor technology. A capacitor has tendency to decay charge, so the data will not remain on the chip for long. To avoid this, DRAM necessitates an external memory refresh circuit that charges these capacitors to their original state by rewriting the data periodically. Due to its necessity of a system to do refreshing, DRAM has more complicated circuit and timing necessities than SRAM.. The foremost advantage of DRAM is its great packing density. This is because its memory cell comprises of a single FET and a capacitor compared to six transistors (6T SRAM cell) or four transistors (4T SRAM cell) SRAM. Due to this, DRAM has very high densities, leading to economic value of each bit. This is main reason that DRAMs are more widely used than SRAMs.

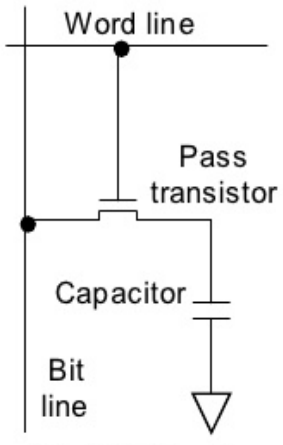


Fig.4.1 A DRAM Cell

Synchronous Dynamic Random Access Memory (SDRAM) is a DRAM where the operation of its external pin interface is harmonized by an externally provided clock signal. SDRAM has a synchronous interface, whereby alterations on control inputs are recognized post a rising edge of its clock input. The main advantage of SDRAM is that the memory is separated into several equally sized but sovereign sections called banks, so that the device can operate on multiple banks simultaneously and speed up access due to alternating pattern of access. Due to this, SDRAMs realize better concurrency and larger data transfer rates than asynchronous DRAMs.

4.2. Memory Controller Design

The ICs of these memories are enormously complicated. They also need Memory Controllers, Precharge circuits, etc. Memory controllers encompass the logic obligatory to read and write to SDRAM and to refresh the SDRAM. They also decide when the memory banks should be precharged.

There are six states in this SDRAM controller as follows:

i. Initial State: It is the state when SDRAM Controller is turned on and also after precharge state.

ii. Active State: The memory controller state will change to Active State on any R/W request. It is to open a row for read and write operations.

iii. Read: Read state is for read operation. The read operation dominates over write operation.

iv. Write: Write state is for write operation.

v. Refresh State: This represents periodic refresh of one row of each bank.

vi. Precharge State: In this state rows of all banks will be closed (or deactivated).

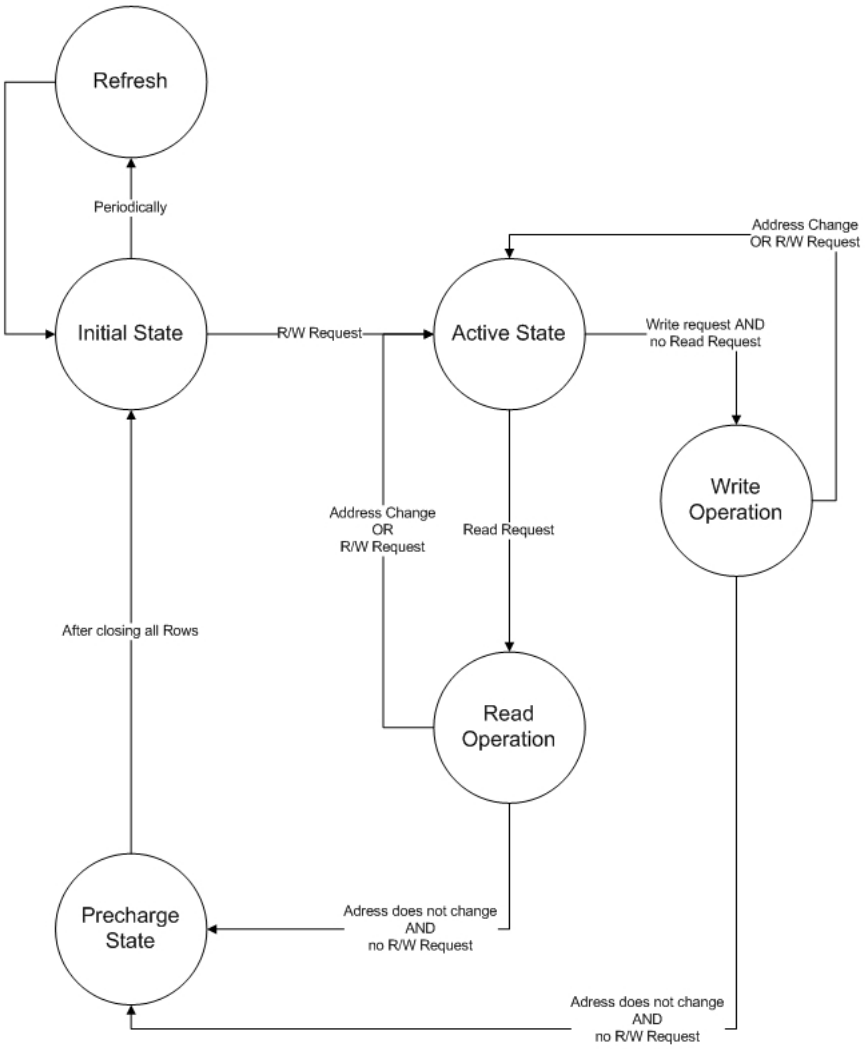


Fig.4.2 State Diagram of designed SDRAM Controller

The Synchronous DRAM Controller has 59 input pins, 37 output pins and 16 inout pins. It has data transfer bus of 16 bits. The circuit can control Synchronous DRAM divided in two banks.

4.3. Memory Controller Peripheral Pins:

- sdramCKE: This is a 1 bit output signal from memory controller to SDRAM. It specifies clock enable and is active high signal.

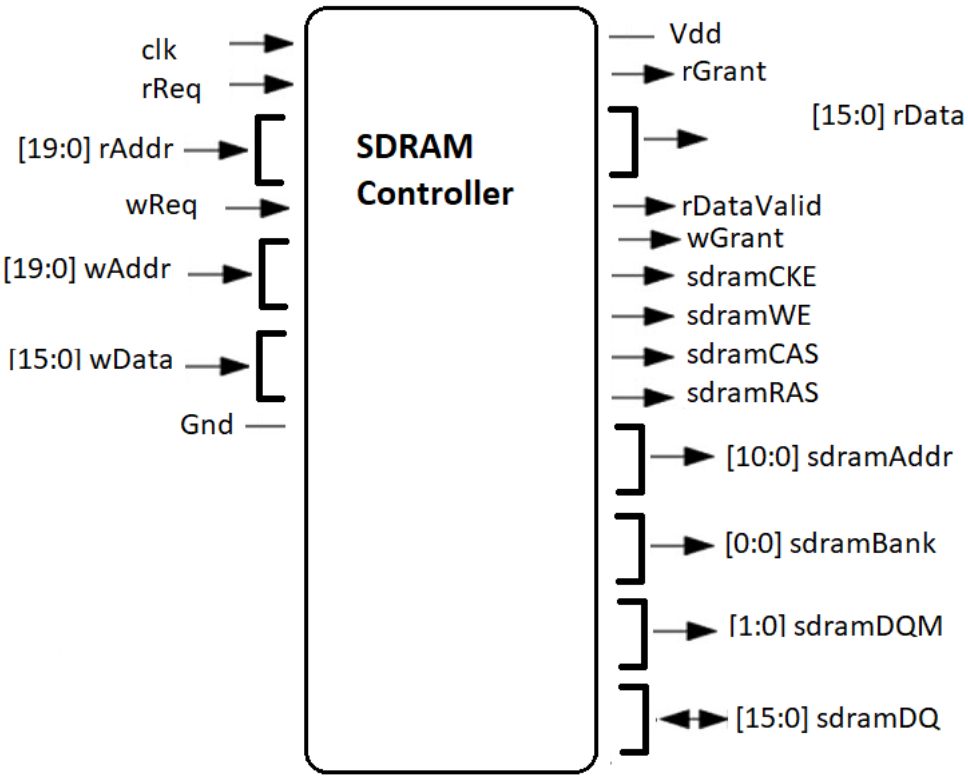


Fig.4.3. Pin diagram of SDRAM Controller

- clk: Clock input to memory controller.
- sdramCAS: This is Column Address Strobe (CAS) signal for SDRAM and an output command signal in SDRAM controller. sdramCAS signal when high, tells the memory that associated address is a column address.
- sdramRAS: This is Row Address Strobe (RAS) signal for SDRAM and an output command signal in SDRAM controller. sdramRAS signal when high, tells the memory that associated address is a row address.
- sdramWE: It is write enable output from controller, sent to SDRAM.
- sdramBank: It is a 1 bit output signal from memory controller to select a bank in SDRAM.

- sdramDQ: It is data bus for both reading and writing from/to the SDRAM. It is a 16 bit bidirectional port.
- sdramDQM: This is a 2 bit output port for data masking. When high, sdramDQM signals suppress data inout/output transfer through data bus. Mostly data bus has one DQM line per 8 bits. Thus here sdramDQM is of 2 bits.
- rReq: It is a single bit input signal sent to memory controller to request read operation.
- rGrant: : It is a single bit output signal sent from controller to the memory (SDRAM) to grant read operation.
- rAddr: It is an input port of 20pins. It tells SDRAM controller the location from which data has to be read by controller from memory.
- rData: The data read by SDRAM controller from memory is output through this port of 16bits.
- rDataValid: This signal is for validating the data read from memory by SDRAM controller on read request.
- wReq: A one bit input signal sent to memory controller to request write operation.
- wGrant: : It is a 1 bit output signal sent from controller to the memory (SDRAM) to grant write operation.
- wAddr: It is an input port of 20pins. It tells SDRAM controller the location to which data has to be written by controller to memory.
- wData: The data read by SDRAM controller from memory is output through this port of 16bits.
- sdramAddr: This 11 bit output port is used by SDRAM controller to tell the memory the address of location in selected memory bank to/from which data has to written/read.

4.4. Memory Controller Working

The input clock memory controller is active at positive edge. The state of controller changes from Initial to Active on R/W request(s). The Read request will dominate over Write request (if both requests are made at same time). If sdramDQM sent with write data is high, the data sent through sdramDQ will not actually get written to the DRAM. When averred high two cycles afore a read cycle, then data from memory is not output to sdramDQ from the chip. Precharge State is meant for closing opened rows of both banks. If SDRAM COntroller is in Initial state then it changes periodically to Refresh state, meant for keeping required capacitors charged for holding data. After closing all rows in Precharge state, the controller state will change to Initial State. Signal sdramDQM is for Data Maskin. If sdramDQM is high, the data transfer through data bus sdramDQ is suppressed. This suppress means the data is not truly written to the DRAM in write operation and will not be the read from the SDRAM. The flow chart for HDL design of this SDRAM Controller is presented in Fig.4.4.

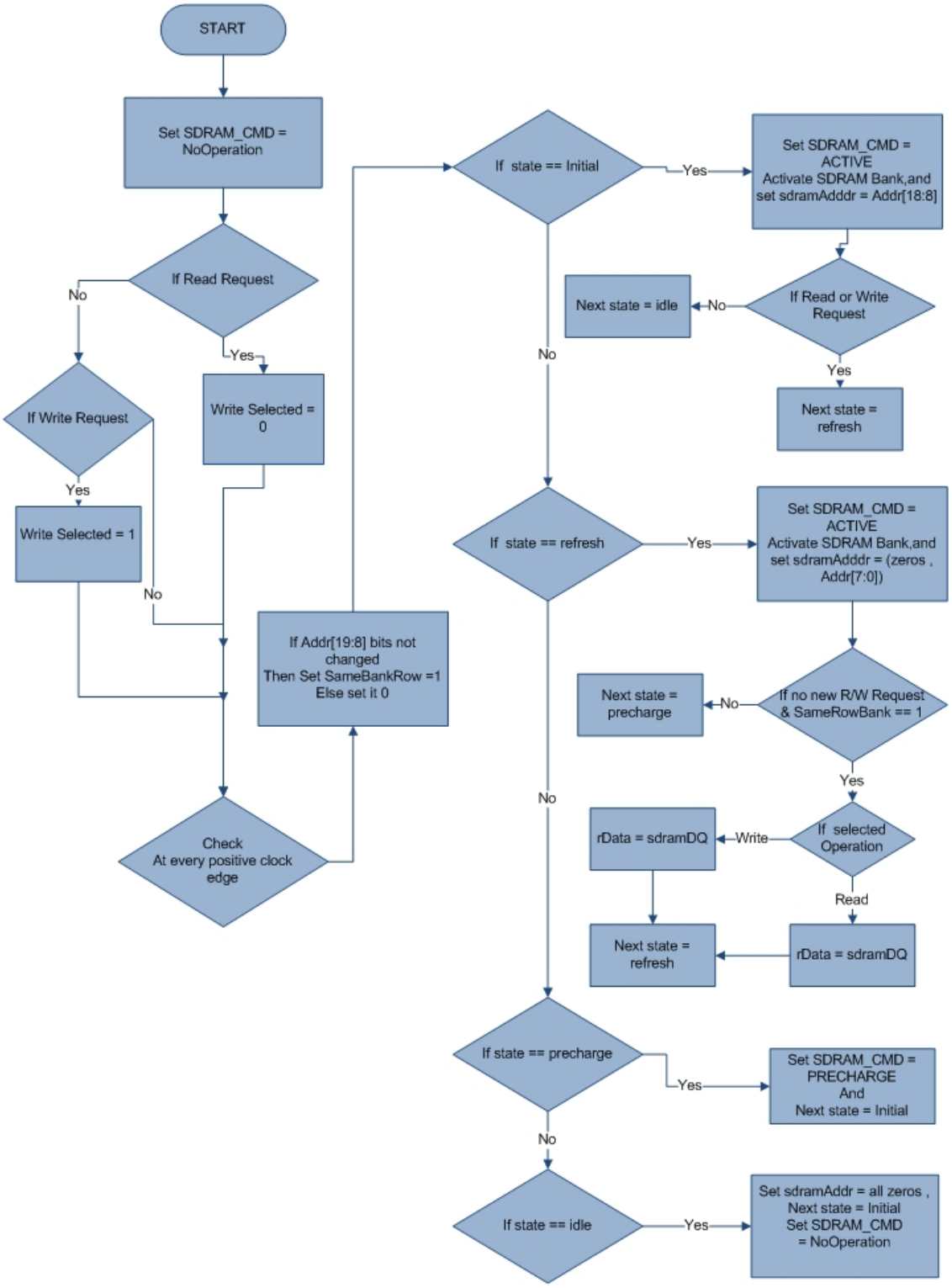


Fig.4.4. Flow Chart of SDRAM Controller

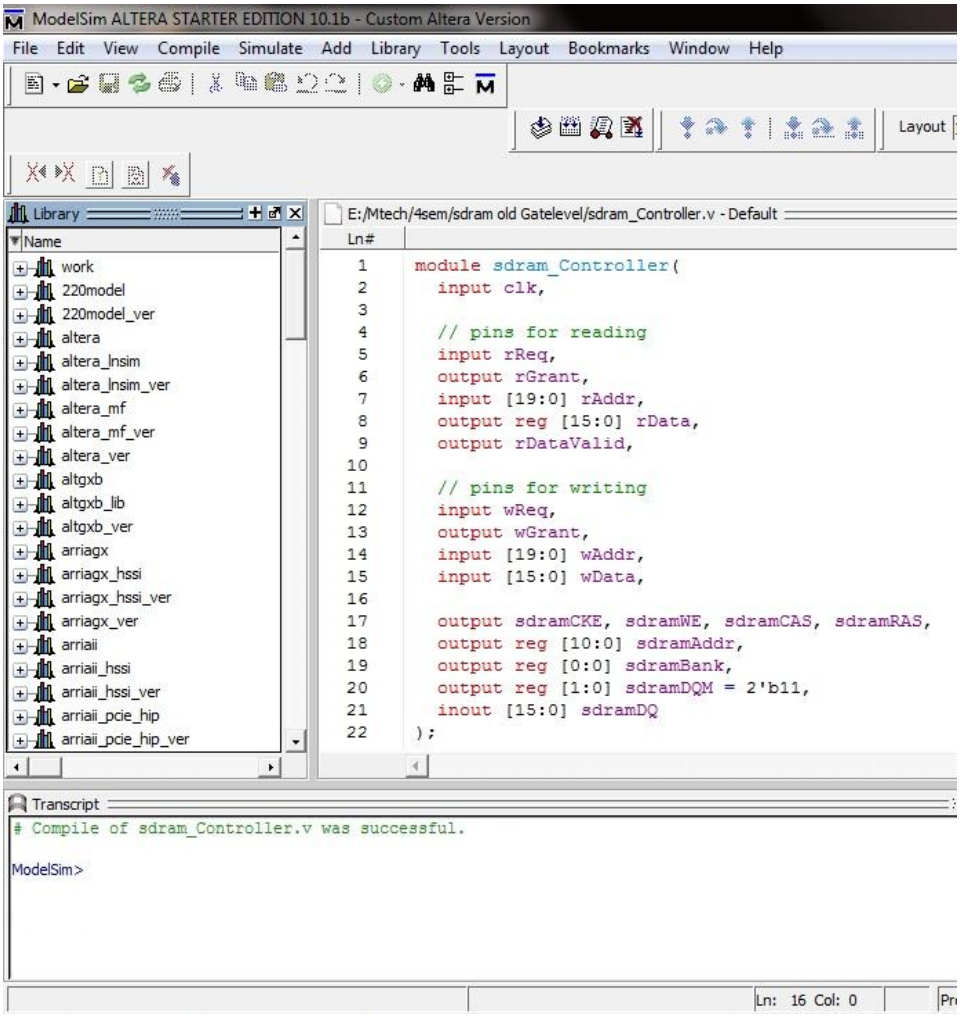


Fig.4.5. Screenshot of Verilog HDL code of SDRAM Controller designed

Chapter 5

Synthesis of Gate-level Netlist

The circuit was designed using Verilog HDL (High-level Description language). A **Gate level Netlist** is representation of design in form of logic gates. Synthesis is development of gate level netlist (embraces nets, sequential and combinational cells and their connectivity) from Register Transfer Level explanation of the design to gate level netlist (contains nets, sequential and combinational cells and their connectivity). The Gate Level Netlist is specific to technology used.

Goals of Synthesizing Gate Level Netlist:

- To get a gate level netlist
- The RTL and the netlist should be logically equivalent
- Insertion of clock gating
- Optimizing the logic
- Design for Test Logic can also be inserted

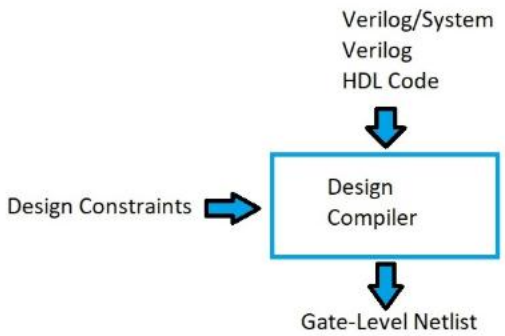


Fig.5.1.Inputs Required for Synthesis of Gate Level Netlist

To synthesize Gate Level Netlist using Synopsys Design Compiler, the following is required:

1. Design HDL code

2. Design Constraints

3. Library files

5.1. HDL of Design

As discussed earlier, the design was written at behavioral level in a .v file using Verilog HDL language on Altera Modelsim.

5.2. Design Constraints

These are to ensure design goals in expressions of Area, Timing and Power to acquire the finest probable implementation of a circuit. Incorrect or incomplete design constraints can lead to wasted design iterations. Constraints defined by the designer are design specific. Synopsys Design Constraints (SDC) is a means to postulate the envisioned design, as well as the timing, power and area constraints for a design. In this work, SDC was written in Tool Command Language. SDC comprises chiefly subsequent constraints that are very necessary for design

i. Clock definition

ii. Case analysis

iii. False path

iv. Multi cycle path

v. Min/Max delay

vi. Input/output delay

vii. Virtual Clock

i. Clock definition:

It is used to set constraints for a clock, for example its frequency, pulse width, rise time, fall time, etc. If the circuit has no clock, i.e., it is a combinational circuit, then a virtual clock must be defined in the design constraints file.

eg. create_clock u13/Z -name CLK -period 25 -waveform { 5 10 15 25}

ii. Case Analysis

Case analysis specifies a list of ports or pins that have a constant logic value (0, 1, or static) or only one type of transition (rising or falling). Case analysis places any particular set of pins in given operating mode without altering the netlist.

eg. set_case_analysis 0 U1

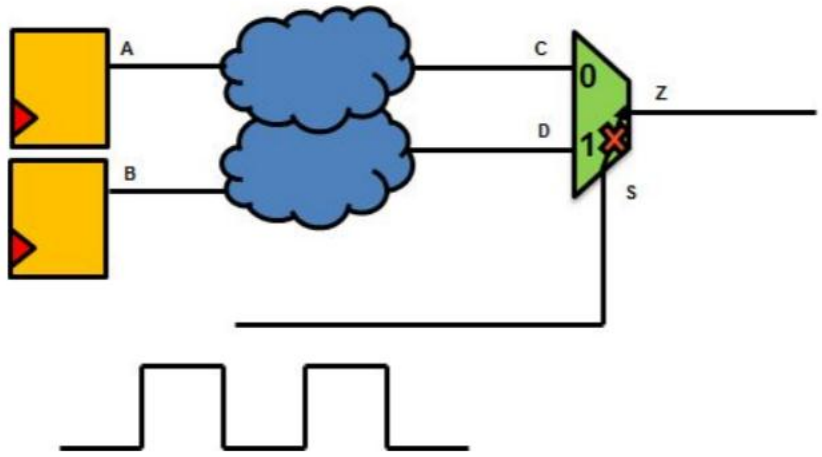


Fig.5.2. Example of Case Analysis application

iii. False Path:

False paths are timing paths that will never really be exercised in the final design. This is because they are physically present but logically not present. The constraint is used to mark paths in a design as false, to overlook them while performing timing analysis. For example in fig below:

There is physical connection from pin in0 through in1 to output of MUX1 but this path is logically absent. This is because the select input sel cannot have two different values at same time.

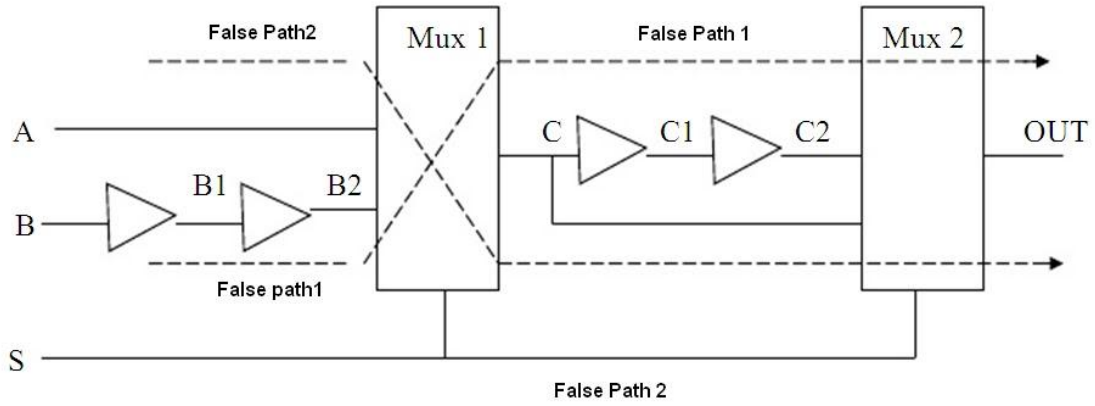


Fig.5.3. False Paths in a Circuit

Here Fig.5.3. shows that logic conditions for each false path 1 and false Path 2 will never occur. But while enhancing the circuit to be most effective, such paths can affect the meeting of timing constraints.

iv. Multicycle Path: Any path that consumes more than 1 clock cycle for transmission of data from its startpoint to its endpoint is called a Multicycle Path. This occurs generally in cases when data launched from one flop is permitted to consume time larger than a clock cycle to arrive at the destination flop. This is guaranteed by gating the data or gating the clock to the destination flip-flops. The set_multicycle_path command is helpful to alter the default launch edge used for either the setup or hold analysis. By default, Timing Analyzer shoulders that all data signals consume one clock cycle to navigate to their destinations.

A typical System on Chip (SoC) consists of many interconnected components. Each of these component works on dissimilar frequencies dependent on performance and other necessities.

Multi-cycle paths can be sited in the design by the designer in any of the following scenarios:

- **Very large data-paths:** In various cases there are data paths that take multiple clock cycles for data to move between start and end points. Such paths bound the frequency of whole component. Let there be a situation such that one of the components, which is intended to function at 500 MHz. But a data path this case is too hefty to labour at this frequency and consumes 5ns, thus can work at maximum of 200 MHz. Thus if all paths are assumed to work at single cycle, the design will not work at more than 200 MHz. A solution can be that if this path is unheeded then the rest os the components on the chip can reach 500 MHz. Thus, this path is sacrificed solitary to make the other components work at 500 MHz.

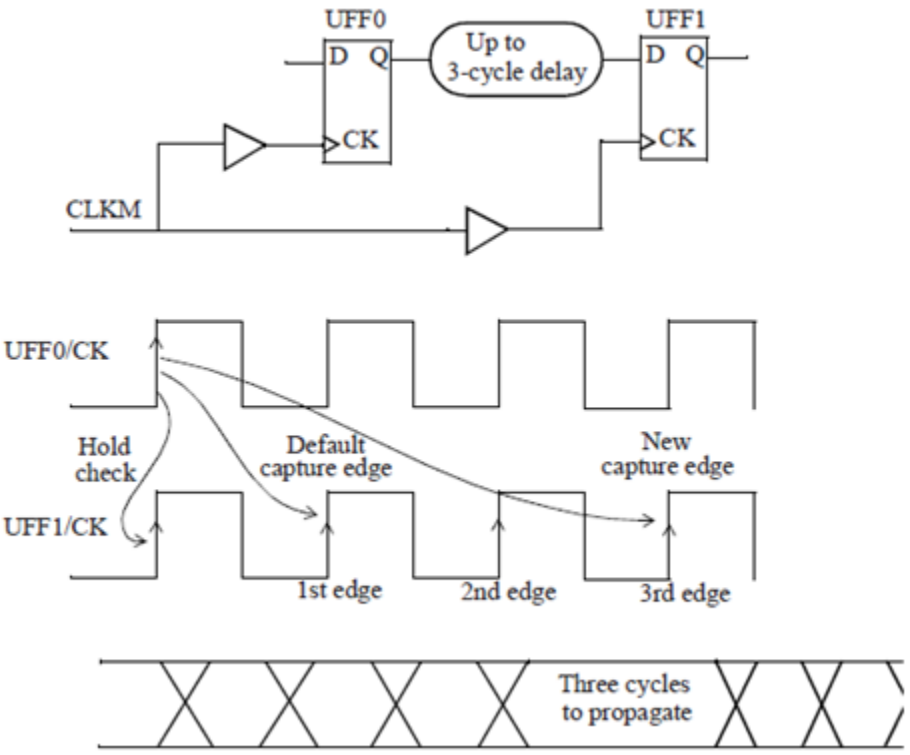


Fig.5.4. Multicycle Path

- **Paths between slow clock and fast clock:** In various cases, data paths use clock to start-point flip-flop that is half in frequency compared to clock to end point. This means, the start-point can direct the data at only half the frequency than the end point can accept. In such case, there is no gain of supplying clock to the end-point at twice the start-point

clock frequency. The data path can be altered such that endpoint collects data after a gap of one cycle. In other words, in place of single cycle data-path, we can afford a two cycle data-path in such a case. This will really save power as smaller drive strength cells with less area and power can be used.

v. Min/Max delay:

Constraint for signal to have minimum delay while propagating between two pins is required mostly to avoid Hold Time Violations. Maximum time delay of signal needs to be constrained for keeping check on Setup Time. These constraints are set using commands set_max_delay and set_min_delay. The set_max_delay and set_min_delay constraints both have options to allow to specify the paths where the delay should be applied. -from and -to options can be used to target any pins in the netlist as the source or destination. Specific delay contraints for rising or falling edge of the source or destination clock can also be applied.

vi. Virtual Clock:

A virtual clock is a generated clock that does not correspond to a pin or port in the design. Virtual clocks thus do not create timing paths within the design. Virtual clocks are primarily used to specify timing constraints for clocked devices outside the design. The command create_generated_clock can be used to define clocks with a fixed relationship (frequency) to another clock. Propagation delays from the source are included in clock path delay calculations. Virtual clock was used in synthesis of Gate-Level Netlist of Apporoximate MAC Unit.

5.3. Technology Library

Standard cell

Standard cell are used to reuse logic gates and other basic elements in a design. Standard Cells provide abstraction in design. A standard cell is a collection of FETs and interconnect

assemblies that provide a combinational logic equation (e.g., AND, OR, XOR, XNOR, inverters) or a sequential logic function (flip-flops or latches). The simplest cells are straight depictions of the digital NAND, NOR, and XOR gates. Cells of a lot larger complexity are also frequently used (such as a MUXed D-input flip-flop). The digital logic function of a cell is termed its Logical view.

Technology Library is a collection of standard cells of combinational elements like AND, OR, NOR, NAND, NOT and buffers and likewise memory holding elements like flip-flops and latches. These cells are comprehended as FETs per fixed-height, variable-width. This means cells of same logical functionality based on FETs of different width are different. The crucial feature with these archives is their permanent height, which consents them to be employed in rows, helping the course of automatic digital layout. The cells are improved layouts with smallest area and minimum delay. The standard-cell library used had following main modules:

- **Library Database:** It contains of several of views counting layout, schematic, symbol, abstract, and other logical or simulation views. The data in the library files used was in Synopsys Milkyway format. Further popular format is Cadence LEF format. Synopsys Milkyway format contains condensed information regarding the cell layouts, adequate for programmed tools for placement and routing.

- **Timing Abstract:** The timing summary is mostly in Liberty format. Its purpose to provide functional descriptions of each cell such as, timing, power, and noise information

- **SPICE Models:** SPICE stands for Simulation Program with Integrated Circuit Emphasis. It is open-source simulator for electronic circuits. It is a package used in integrated circuits and printed circuit board design to test if circuits can work when all its components are combined together and to foresee circuit behavior. The standard models broadly used consist of BSIM3, BSIM4, BSIMSOI, PSP, HICUM, and MEXTRAM.

- **DRC Checks:** DRC stands for Design Rule Check. Design rules are a successions of parameters delivered by semiconductor manufacturing industries that help the designer to test the rightness of a design. These checks refer to checking that if the modules of the

integrated circuit when placed will meet geometric restrictions or not. IC designers have to check that chip to be mass-produced from the design will function in the approved manner and reliably. Also it has to be ensured that the dies can be produced with acceptable yield. Design rules for production are developed by process engineers predominantly based on the technology length they work on and the tools they practice.

Logic Synthesis: It is procedure by which an abstract representation of anticipated circuit behavior, usually at register transfer level (RTL), is curved into a design enactment in expressions of logic gates. The tool used to perform this is a synthesis tool. The synthesis tool utilized in this work is Design Compiler by Synopsys.

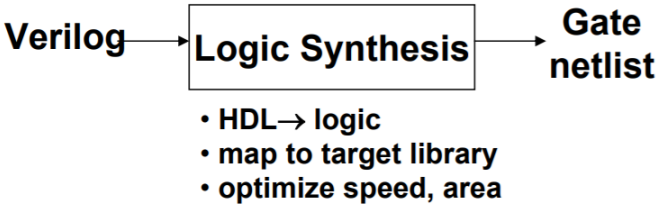


Fig.5.5. Logic Synthesis

The manner of precisely transforming the ASIC's register-transfer level (RTL) portrayal into a netlist independent of technology is done by using the technology library cells. The tool used for Gate Level Synthesis is **Design Compiler**. Appropriate synthesis techniques protect scientific equivalency among the early RTL code and the netlist synthesized. The netlist covers all pins, registers and wires present in RTL. Also netlist has no unmapped RTL statements. This was safeguarded by formal verification. The netlist is the standard-cell illustration of the ASIC design, at the logical view level. It entails of objects of the standard cell library gates, in addition to port connectivity among the gates.

Chapter 6

Extracted Timing Models

In a design after a block meets its timing constraints (at block level), interior timing analysis is not required to be done again for timing analysis of the chip if a model comprehending timing arcs for block interfaces is used instead. This model is known as **Extracted Timing Model (ETM**). Features of Extracted Timing Model are as follows:

- Completely models the full input and output timing relations of the design.
- Contents of the design are hidden
- The timing arcs have delay depending on input transition and output load. Thus, ETM is practical with dissimilar transition times at input and diverse output loads.
- Single PVT for each model. This means a timing model is specific to Process, Voltage and Temperature (PVT) conditions. Different PVT corners will have different timing model of the same design.

6.1. Creation of Extracting Timing Model

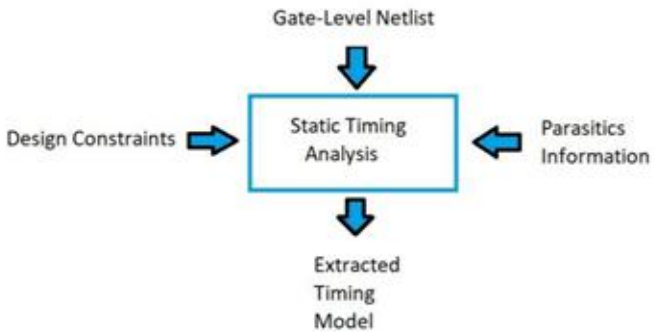


Fig.6.1. Inputs Required for Creation of Extracted Timing Model

Steps for Verification of Extracting Timing Model of a design are as follows:

1. Synthesis of Gate level Netlist
2. Specifying Design Constraints and Parasitics Information
3. Specifying Design Constraints
4. Static Timing Analysis of the Design
5. Verification of Extracted Timing Model

Synthesis of **Gate level Netlist** from Verilog HDL Code of each design was done using Synopsys Design Compiler. **Design Constraints** to be specified are same as used to synthesize Gate-Level Netlist. They are specified using tcl (Tool Command Language). The Synthesis of Gate-Level Netlist and Design Constraints Specification is discussed in Chapter 5.

6.2. Specifying Parasitics Information

There are three popular specifications for parasitic information of standard cells that are used in industry:

i. **Detailed Standard Parasitic Format:**
   Detailed Standard Parasitic Format (DSPF) is a format that can be read by almost all EDA tools. It consists of most detailed description of parasitic information.

ii. **Standard Parasitic Exchange Format:**
   Parasitic data (Resistance and Capacitance values) of interconnects in a chip are required to calculate delays. Standard Parasitic Exchange Format (SPEF) is a representation of parasitic data of interconnections in a chip in ASCII format. It is standardized by IEEE standard. SPEF captures parasitic resistance and capacitance of non-ideal wires. These interconnects have inductance too which is not incorporated in SPEF. The main functions of SPEF is delay computation and safeguarding signal integrity of a chip. The

specification for SPEF is a portion of standard 1481-1999 IEEE Standard for Integrated Circuit Delay as well as Power Calculation System.

iii. **Reduced Parasitic Exchange Format:**

This format of parasitic information is more detailed than SPEF but not as verbose as RSPF. RSPF characterizes each net as a Resistance-Capacitance pi model that entails an equivalent near capacitance at the driver of the net, an equivalent far capacitance for the net. It also calls for equivalent resistance linking these capacitances. Every net partakes a single pi network, nevertheless of how many pins are on the net.

6.3. Introduction to Static Timing Analysis

Static Timing Analysis (STA) is a technique of authenticating the timing performance of a design. This is examination of all paths in design for timing restrictions in worst case situations. It assumes the worst plausible delay for every logic element. The logical working of design is not tested. It only tests that signals incoming at appropriate time and are stable for desired amount of time, etc. Any violations found need to be fixed. Accordingly to fix timing violations in a design, its Static Timing Analysis is indispensable. The term “Static” in Static timing analysis refers to that the timing analysis done is autonomous of input vector. A hip of complex design comprises of huge numbers of logic paths in it. Thus several path reports are generated to perform static timing analysis. STA is advantageous as timing analysis is performed on all possible paths. In contrast to simulation of design, static timing analysis is

- Faster – Static Timing Analysis is faster than Dynamic Timing Analysis because simulation of numerous test vectors is not required in STA.
- More Thorough – STA is further detailed since it tests the worst-case timing for all combinations of logic that are plausible, instead of only those alerted through a specific test vectors.

Importance of Static Timing Analysis also lies in the point that the signal-integrity effect can result in cross-talk noise effects. Cross talk can distress the design timing.

Static Timing Analysis is needed in combinational circuits also because of following reasons:

- To reduce the delay of critical paths.
- Checking if false paths are present and removing them
- Choosing one of multiple logically equivalent designs. For this the design with least delay in its most critical path is chosen.

6.4. Modes of Static Timing Analysis:

i. Single Mode:

In single mode, late and early paths from a single corner are used for both setup and hold analysis. The timing model if generated in single mode has late and early paths from a single corner. Therefore, the timing model generated for setup is the same as that generated for hold analysis. Thus the same model can be used for setup as well as hold analysis.

ii. Scan Mode:

The objective of Static Timing Analysis in this mode is to have testing relaxed by providing a humble method to set and notice every flip-flop in an Integrated Circuit. Scan_in and scan_out outline the input and output of a scan chain. A scan enable pin is a distinct signal that is auxiliary to a design. When this signal is applied, each flip-flop in the circuit is linked into a long shift register.

The function of clock signal is to govern all the FFs in the chain through shift phase and the capture phase. A random arrangement of binary data is made input to the chain of flip-flops, and the state of each flip-flop is noted.

iii. Boundary Scan Mode:

Also called as b-scan mode, Boundary scan mode of STA is a method for examining wires on sub parts inside an IC. Boundary Scan mode was created by the Joint Test Action Group (JTAG) as a specification for boundary scan testing. It was regularized in 1990 as an IEEE Standard. It comprises the introduction of at least one test cell that is associated to each pin of the design which can selectively dominate the working of that pin.

iv. On Chip Variations (OCV) mode:

In OCV mode, late paths from max corner and early paths from min corner are used for setup and hold analysis respectively. Here, OCV mode has been used to generate the extracted timing models. Therefore, the timing model for setup analysis is the same as that generated for hold analysis. It has advantage of considering on-chip variations to generate a timing model that can be used for both Setup and Hold Analysis. On Chip Variations are discussed below in detail.

On Chip Variations (OCV):

On Chip Variations are variations that depend on factors like Process, Supply Voltage and Temperature. These are also called as inter-chip variations. Several Integrated Circuits (IC) are fabricated collectively as a chip. Each die in this fabricated chip is an IC. Thus fabrication processes exhibit differences from die to die. For example, etching may be more on one IC and less on other. These parameters affect the delay of the cell. Mostly variations are in following parameters:

Process:

Process variation refers to differences in dies of same fabricated chip due to diference in fabrication process across the chip. In the course of manufacturing a die, the area at the centre and that at the boundary will have dissimilar process variation. Thus masking, etching and several other processes may vary over chip. The reason for this variation is that fabricated layers will not be uniform all over the chip. Moving towards the edge of the chip, layers can contrast in their sizes. Please note that process variation is steady, not abrupt.

Below are few imperative factors which can lead to process variation;

- Wavelength of the UV light
- Manufacturing flaws

Process Variation moreover results in RC variation, where R and C are resistances and capacitances in MOSFET.

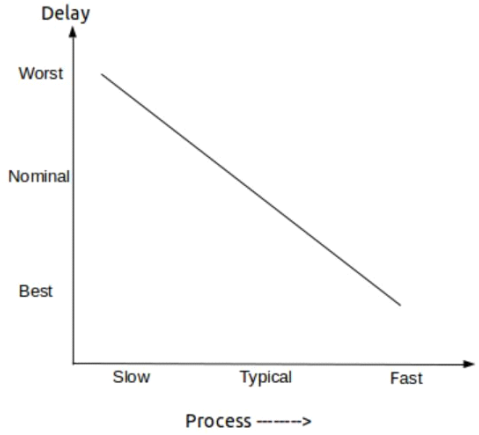


Fig.6.2. Propagation Delay vs. Process Corners

Voltage Variation:

Supply voltage for a chips manufactured in today's world is very low. A chip operating at 1V, can have voltage variation from 1.1V to 0.9V. This is a considerable percentage of supply voltage in ideal case. Thus voltage variation is considered. The most significant reason for instabilities in supply voltage is IR (Current-Resistance) drop. It is produced by the flowing of current via parasitic resistance of the power grid. IR drop lessens the supply voltage from the requisite value.

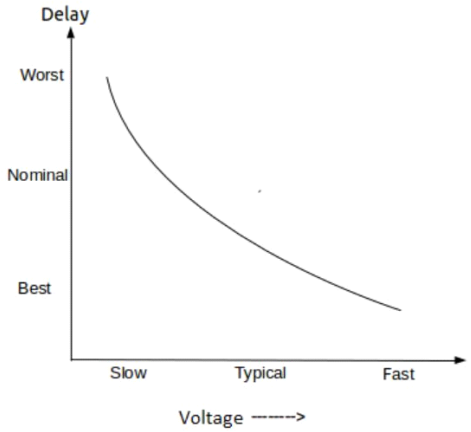


Fig.6.3. Propagation Delay vs. Supply Voltage

Temperature Variation:

The temperature variation refers to temperature at junction in chip and surroundings temperature. The temperature dissimilarity must to be considered as the temperature at the junction inside the

chip can contrast significantly. This alteration in temperature distresses parasitic resistance values. Change in parasitic resistance deviates the delay of a cell surges with increase in temperature. As temperature rises, mobility and threshold voltage start falling. The delay is contrariwise proportional to the mobility and directly related to the threshold voltage. Therefore both mobility and threshold voltage selects the value of delay.

From current equation of a MOSFET, i.e.,

$$I_d = [\mu n C_{ox} (W/L)(V_{gs} - V_{th})^2]/2$$

In the higher technology lengths, the supply voltage is larger, the dominance of Threshold voltage (V_{th}) is very less as ($V_{gs} - V_{th}$) value is large. Hence current is principally influenced by the mobility. So at higher technology lengths, when the temperature surges mobility decreases and as a result the delay will be more.

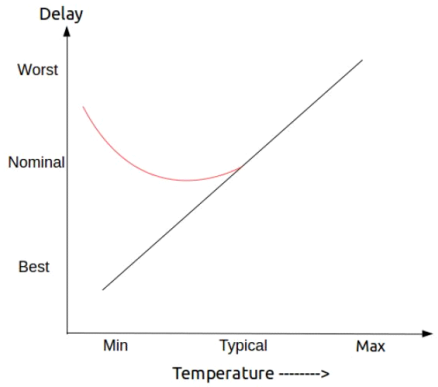


Fig.6.4. Propagation Delay vs. Temperature

At the lower technology lengths, used in today's world, the supply voltage is very low, so the difference ($V_{gs} - V_{th}$) is minor and the square of this value is insignificant. This leads to lessening in drain current, which upsurges delay at lower temperature.

6.5. Performing Static Timing Analysis

Before Extracted Timing Model can be generated, it must be ensured that a design is free from timing violations and meets all required timing constraints. To analyze a circuit for timing restrictions or to perform STA main steps are as follows:

i. Breaking the design into sets of timing paths.

ii. Computation of the propagation delay over every path.

iii. Examining timing constraints violated in sets of timing paths.

Timing path: Timing path is the path from a start point to an end point. Start point and end point are as follows:

- Start Point: It is usually input pin of the design or a block in design or clock pin of the flip-flop, latch or memory (any sequential cell). This is because data flow starts from here with respect to that block or design. It can also be a clock-gating element's input pin.
- End Point: It is mostly output pin of the design or block where data is captured. or clock pin of the any sequential cell. As an output pin of a block can be input pin of another block, endpoint can also be an input pin. Set, Reset or Clear pin of a sequential cell can also be an endpoint.

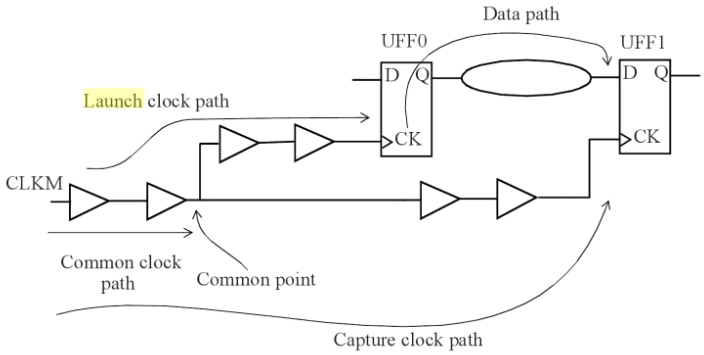


Fig.6.5. Type of Timing Paths

In Static Timing Analysis (STA), static delays like gate delay and net delays are measured in every path and these delays are matched across their requisite extreme values. The design to be evaluated is divided into several timing paths comprising of gates, flip flops plus interconnecting wires between them.

Classification of timing paths according to the type of signals is as follows:

i. Data Path

ii. Clock Path

iii. Clock Gating Path

iv. Asynchronous Path

i. Data Path:

Based on Start and End point, there are four types of data paths as follows:

i. Input to Output: It starts at an input pin and ends at an output pin. It is void of any sequential elements. It is rarely found in a synchronous design.

ii. Input to Register: It starts from an input pin of design or block and ends at input of a register. The register is controlled by clock so it is semi-synchronous.

iii. Register to Register: This data path is purely sequential because both start and end points are input/output pins of some registers. The registers may be controlled by different clocks.

iv. Register to Output: It is from output pin of a register to output pin of design or block.

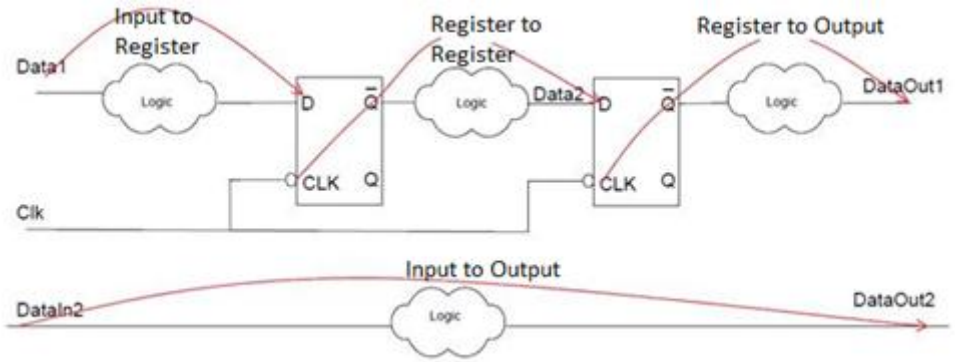


Fig.6.6. Types of Data Paths

ii. Clock Path:

It is path from clock input pin of a block to clock input pin of a launch or capture flip-flop.

iii. Clock gating Path:

A clock gating component is a standard cell output of which regulates the clock input of a sequential circuit. If the clock path passes via a gate used for controlling clock to a sequential cell to accomplish additional gains, then it is called a Clock Gating Path or Gated Clock Path.

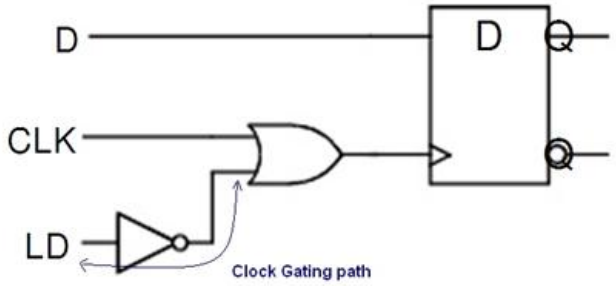


Fig.6.7. Clock Gating Path

iv. Asynchronous Path:

It is a path starting at an input pin that is not a clock pin to an asynchronous set or clear pin of a sequential cell. The working of set/reset pin is free from the clock edge. These are level triggered pins and can start working at any time of data signal. Consequently it can be understood that this path is not concurrent with any of other paths. Hence this path is termed an Asynchronous path.

Propagation Delay Calculation:

The propagation delay is the difference in time when output switches, after corresponding switch in input. There are two kinds of Propagation Delay:

Cell Delay:

It is time delay in signal while flowing from input pin to output pin of a cell. Cell delay information is derived from the library of the cells. i.e. - .lef file.

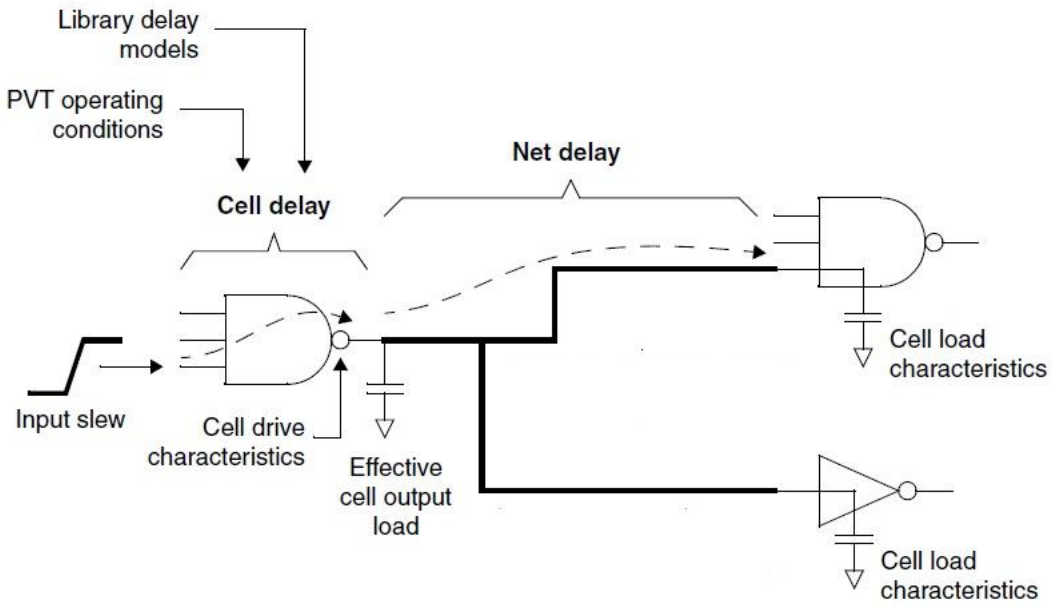


Fig.6.8. Types of Delay

Net Delay:

It is delay in signal while travelling through interconnections between a driver and a load pin. Following Information is required to calculate net delay:

- Features of the Driver cell driving that specific net.
- Load characteristic of the cell being driven by that net.
- Resistance-Capacitance value of that net.

Wire Load Models (WLM):

The function of **Wire Load Models (WLM)** is to compute Net delays. Wire load models are developed by ASIC vendors by means of statistical evidence taken from a range of example designs. Technology File contains the Wire Load Models. ASIC vendors use collect wire load information using statistical data based on physical layout parasitic. Information from the

statistics is used in tables. Wire load models are influenced by the technology used. They describe outcome of wire length and fanout on resistance, capacitance as well as area of the nets. All characteristics (resistance, capacitance and area) are specified per unit length of the wire. Thus using wire length between two pins, the parameters R, C and area are calculated for that wire. The information Net fanout vs load, Net fanout vs resistance, Net fanout vs area values is basically stored in a set of tables. Thus Net delay is calculated using Resistance, Capacitance & Area values and these tables.

Launch Path and Capture Path:

Launch path is the path through which data signal travels from launch flip-flop to capture flip-flop. This includes path from clock input of launch flip-flop to output of launch flip-flop and path from launch flip-flop output to capture flip-flop input. **Capture path** is path consisting of clock path to capture flip-flop. Please refer figure 6.9 for launch and capture paths.

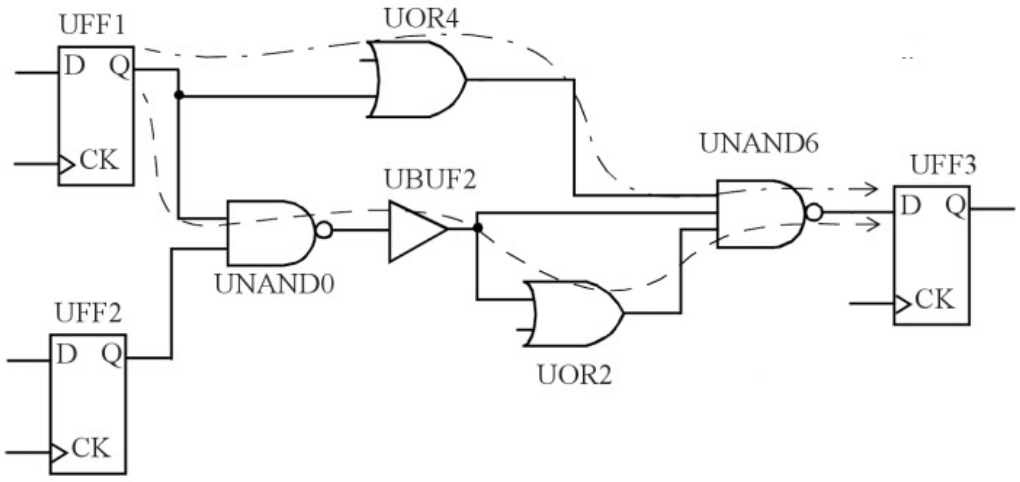


Fig.6.9. Capture and Launch Flip Flops

Critical path:

The critical path is any particular timing path that has maximum delay. The "maximum" is in reference to the timing report options specified in the tool for timing analysis. Any path that does not meet timing requirements is a critical path. Thus a timing report of a design can have more than one critical path. By default if no options are specified during generating timing report, the number of critical path report for a design is one. This is the path that violates the timing requirement the most. Please note that adding further gates in critical path is not tolerable because this can increase delay of that critical path.

Single-cycle path:

A timing path intended to take solitary one clock cycle for the data to travel from the startpoint to the endpoint is called Single-cycle path.

Setting False Paths:

These are the paths that are physically present in the circuit but are logically not conceivable. Data is never transferred through false paths. So in such scenario, such path must be defined as false path before fixing timing violations. A number of times few false paths need to be clearly defined. E.g. for forming a rapport between two Asynchronous Clocks.

In STA, timing analysis of only “true” timing paths is performed. False paths are omitted from timing analysis for false paths don’t need to adhere to timing constraints.

False paths can be specified during generating timing reports in Static timing Analysis. This is done therefore that timing analysis tool discount these paths and maximum operating frequency is not limited. Example is a path among two multiplexed blocks which are controlled by same select signal.

Mostly, false paths exists in following cases:

- From configurations of MUXes like on discussed on previous page.
- Due to test inputs that are used for purpose of analyzing the chip, and are tied off in regular mode.
- Asynchronous inputs to the chip many times also contribute to paths

Setting Multicycle Paths:

A timing path aimed to use addition to one clock cycle to propagate data through it is known as a Multicycle path. While preforming STA, a multi-cycle path should be specified while generating timing report so that the normal restriction on this path is suppressed to allow multiple clock cycles in data propagation. The amount of expected clock cycles for a multicycle path shall also be specified. By default the number of clock cycles if not specified are considered as 2 clock cycles.

Longest and Shortest Path:

Between any two start and end points, there can be many paths. Each path having different amount of delay time.

Longest path is the path in a design, signal propagation through which that takes maximum time. Also commonly called as max path or worst path or late path.

The **shortest path** is the one, where the data signal takes minimum time. It is referred as min path or best path or early path too.

Arrival Time and Required time:

The **arrival time** is time consumed by signal to traverse the launch path. It represents the time by which the data reaches the input of the receiving flip-flop. Arrival time is different for Setup and

Hold analysis for same timing path. This is discussed later in Slack Calculation in this chapter. **Required Time** is the time consumed by signal to travel via capture path from clock input of block or design to clock input of capture flip-flop r other signal capturing sequential cell.

Setup Time:

The period of time for which data must be held stable before the active edge of clock of capture flip-flop is termed Setup time. This means data captured at capture flip-flop must arrive setup time before the active edge of clock at capture flip-flop.

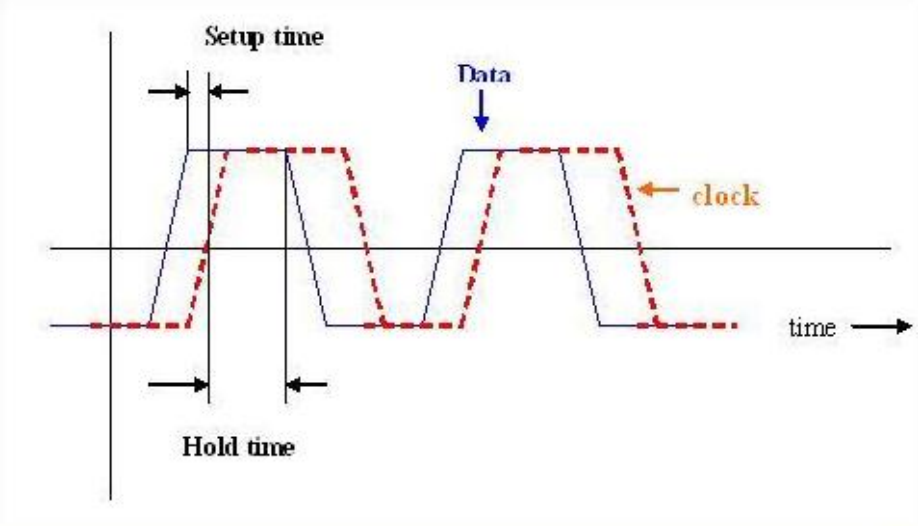


Fig.6.10. Signal flow graph showing Setup and Hold Times

Higher the clock time period less are the chances of Setup Time Violation. Thus, in worst case fixing setup time violations can limit the operating frequency of the circuit. The worst case corner (low Voltage, high Temperature) has more chances of Setup Violation. Setup Check is done at following clock edge.

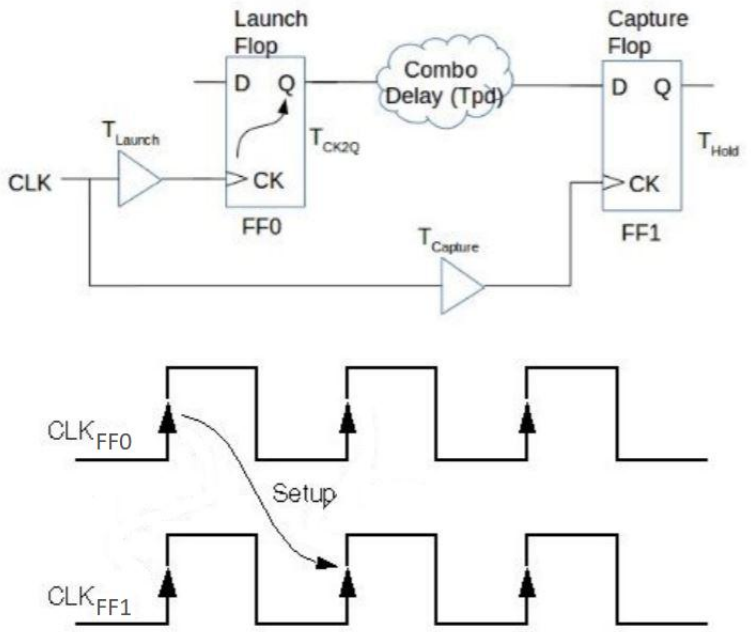


Fig.6.11. Setup Time Calculation

To avoid Setup Violation:

Arrival Time (of Data) < Required Time (of clock)

Here, $Arrival\ Time\ (of\ Data) = T_{launch(max)} + T_{ck2q(max)} + T_{pd(max)}$

$Required\ Time\ (of\ clock) = T_{clk} + T_{capture(min)} - T_{setup}$

$T_{launch(max)} + T_{ck2q(max)} + T_{pd(max)} < T_{clk} + T_{capture(min)} - T_{setup}$

Where T_{clk} is time period of clock used i.e. CLK

Higher the clock time period less are the chances of Setup Time Violation. Thus, in worst case fixing setup time violations can limit the operating frequency of the circuit. The worst case corner (low Voltage, high Temperature) has more chances of Setup Violation. Setup Check is done at succeeding clock edge.

Hold Time:

It is the period of time for which data should remain unchanged post the active edge of clock to capture flipflop. This means that data launched at launch flipflop in next clock edge must not interfere with data captureed at capture flipflop in current edge. To avoid Hold Violation,

Arrival Time (of Data) > Required Time (of clock)

Here, *Arrival Time (of Data)* = $T_{launch(min)} + T_{clk2q(min)} + T_{pd\ (min)}$

Required Time (of clock) = $T_{capture(max)} + T_{hold}$

$T_{launch(min)} + T_{clk2q(min)} + T_{pd\ (min)} > T_{capture(max)} + T_{hold}$

Where Tclk is time period of clock used i.e. CLK

Thus, hold time condition is independent of time period of the clock. Note that Hold check is done at same clock edge of capture flipflop. The best case corner (high Voltage, low Temperature) has more chances of Hold Violation.

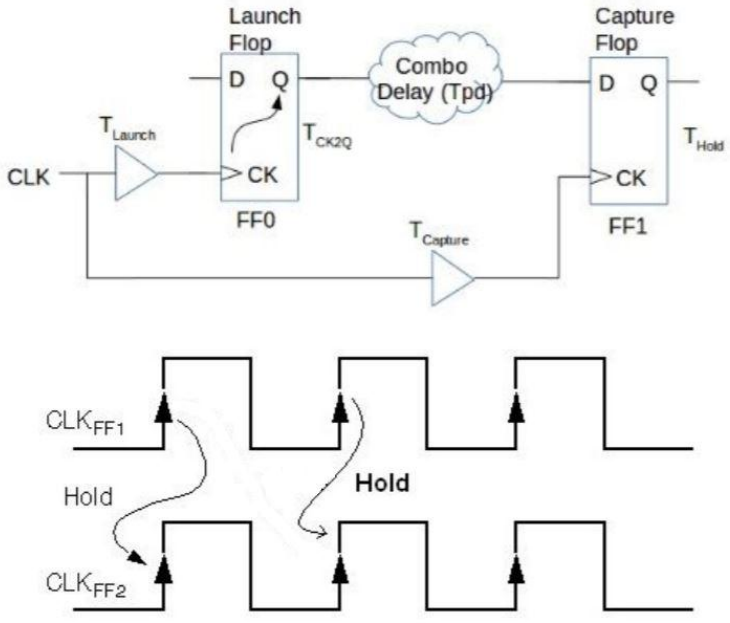


Fig.6.12. Hold Time Calculation

Slack:

The slack related with every datapath is the contrast between the required time and the arrival time. Setup Slack means that how much Arrival Time is exceeds the Required Time while Hold Slack refers to how much Required Time is greater compared to Arrival Time. Arrival Time is different for Setup Analysis and Hold Analysis. This is because in Setup Analysis the worst case will be when launch path elements provide maximum delay. On the other hand, the worst case in Hold Analysis is when launch path elements provide minimum delay. Vice-versa for Required Time.

For Setup time calculation, slack is called Setup Slack and is defined by following equation:

Setup Slack = Arrival Time - Required Time

Setup slack must be positive to avoid both Setup Time violation.

For Hold time calculation, slack is called Hold Slack and is defined by equation:

Hold Slack = Required Time - Arrival Time

Positive Hold Slack confirms no Hold Time violation.

Timing Arcs:

A timing arc expresses the propagation of signals over logic gates/nets and describes a timing relationship between two related pins. Timing Arc represents the timing relationship between 2 Pins of any element or Block or any boundaries. It represents the timing characteristic of the element or block or Boundaries.

Types of Timing Arcs:

i. Net Arc

ii. Cell Arc

- Combinational Cells
- Sequential Cells

A timing arc provides following information:

- An arc between two pins tells the relationship between a set of signals at the two pins .
- Whether the arc is a combinational arc or not.
- Maximum and minimum times it can consume from the source pin to the terminus pin of the arc to traverse in the path
- If the pin is a clock gating pin and the removal of that pin can affect other non-combinational arcs in the circuit.
- Timing sense or Unateness of the arc as elucidated below

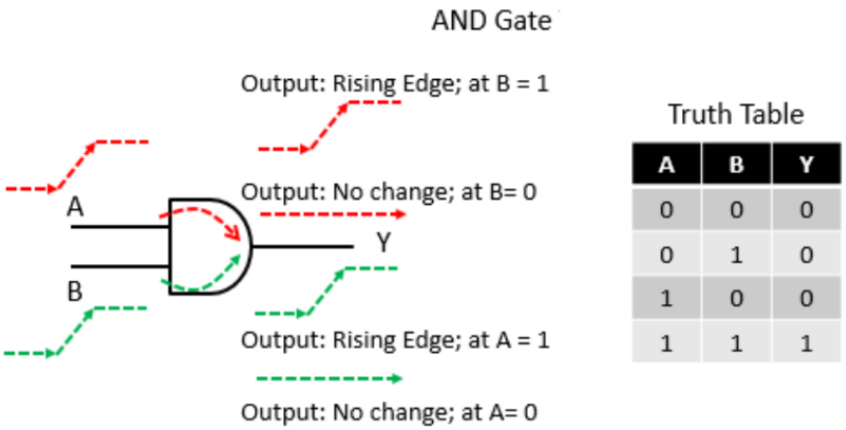
Unateness:

Unate of an arc is described as the sense of traversal from source pin of the timing arc to the destination pin of the timing arc. Timing sense is also entitled as "unateness" of timing arc. Types of Unate are as follows:

i. Positive Unate:

Positive Unate refers to when output signal has rising transition or no change for rising transition in input. Also falling change in input results in falling output or no change in output. As depicted in Fig. 6.13 (a) and Fig. 6.13 (b), i.e., there is a positive unate arc for falling edge at input pin A and another positive unate arc for rising edge at input pin A. Hence there are two positive unate arcs for each of the two input output pin relationships. Thus, there are four positive unate arcs in an AND gate. Similarly for other gates.

E.g: BUFFER, AND gate, OR gate



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Fig.6.13 (a)

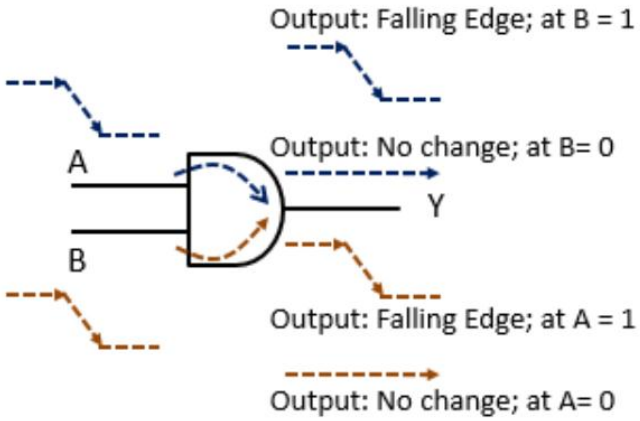


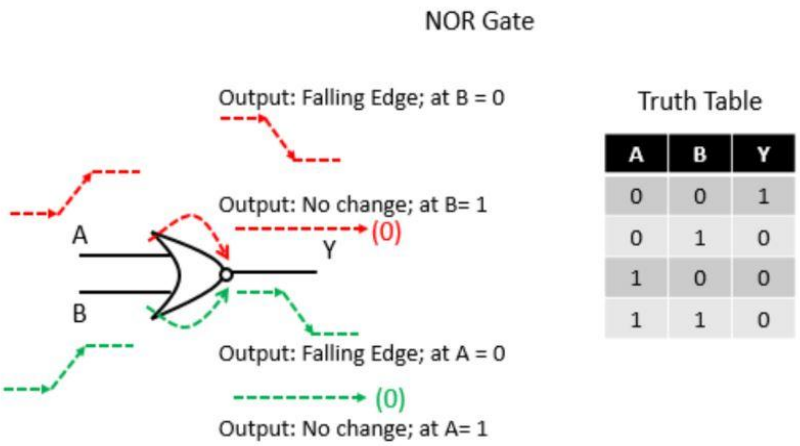
Fig.6.13 (b).

Fig.6.13 Positive Unateness in AND gate

ii. Negative Unate:

Negative Unate refers to when output signal has rising transition or no change for falling transition in input. Also rising change in input results in falling output or no change in output. E.g:

Inverter (NOT gate), NAND gate, NOR gate



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Fig.6.14 (a)

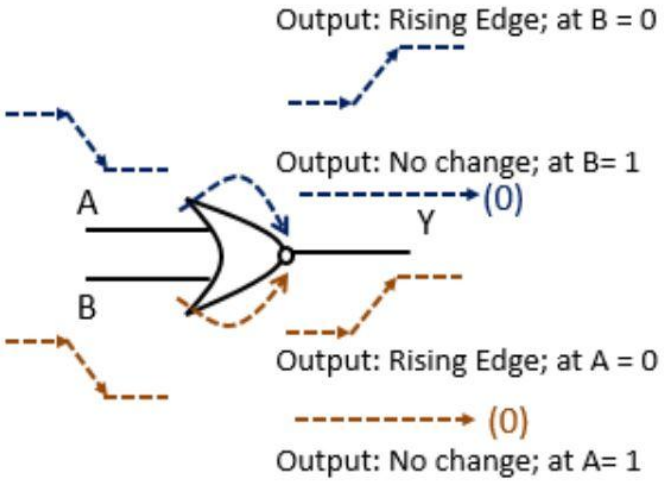
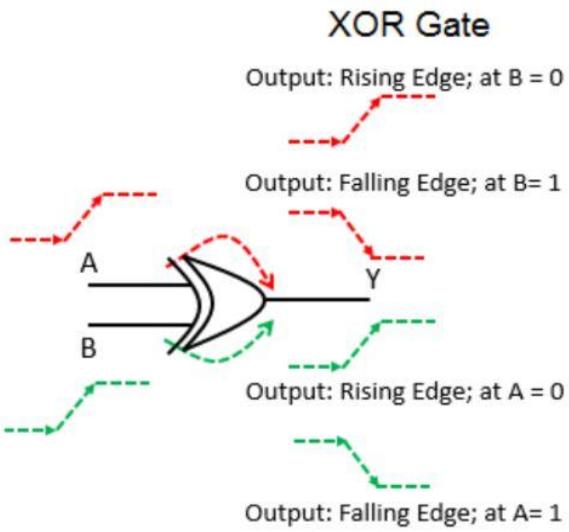


Fig.6.14 (b)

Fig.6.14 Negative Unateness in AND gate

iii. Non-Unate:

The non-unate represents a function where change in output value cannot be told only from the direction of the input change. Change in signal at output pin value does not depends on change in signal at single input pin, but also depends on other input pin. Thus it's difficult to identify this relationship directly. E.g: XOR gate



Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Fig.6.15 (a)

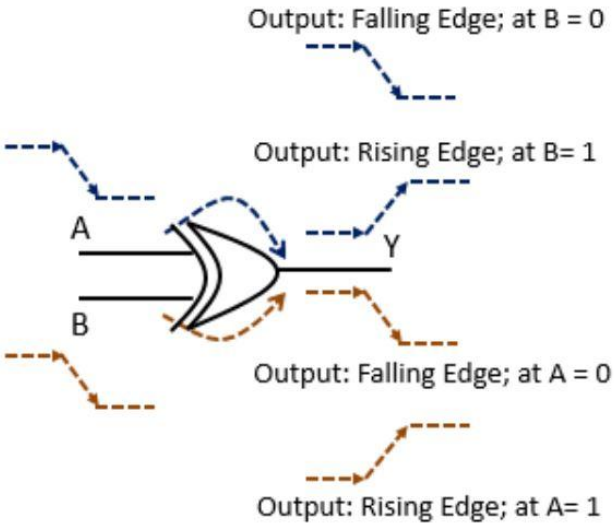


Fig.6.15 (b)

Fig.6.15 Non-Unateness in XOR gate

Derates:

As a result of On Chip Variations, certain cells may be fast or slow than expected. If these deviations are not considered, results may be distrustful and can end in violating setup or hold conditions. In order to model these, derates are used. Net delays and Cell delays are multiplied

with the derates for the capture and launch clock paths.

- **Effect of Derates on Setup analysis:**

  Setup check is done in worst case. The chances of violation of setup condition is more when the clock to launch flip-flop reaches later than the clock to capture flip-flop. For example, if derating factor is set as .07 or 7%, then all the delays in the launch path are multiplied with fator of 1.07 and delays in the capture path are multiplied with factor of 0.93.

- **Effect of Derates on Hold analysis:**

  Hold check is done in best case. Hold Violation has more chances if the clock to launch flip-flop reaches earlier than the clock to capture flip-flop. If derating factor is set as .05 or 5%, then all the delays in the launch path are multiplied with a derating factor of 0.95 and delays in the capture path are multiplied with factor of 1.05.

6.6. Path Reports

After setting the constraints and synthesizing the gate-level netlist, Synopsys Prime Time Tool is employed to perform STA. A report consisting of paths that did not met constraints is known as timing reports. Thus, a timing report usually consists of multiple path reports. A path report is timing report between one startpoint and one endpoint.

```
Point                                                    Incr       Path
-----------------------------------------------------------------------------
input external delay                                    0.000      0.000 f
pmon_trigger_sel[1] (in)                                0.036 &    0.036 f
place_optpopt_d_inst_777/X (EDBSVT16_INV_SKR_2)         0.077 &    0.113 r
U27/X (EDBSVT20_ND2_SKR_2)                              0.044 &    0.158 f
U33/X (EDBSVT24_AOI21_MM_2)                             0.047 &    0.204 r
U40/X (EDBSVT24_NR2_SKR_2)                              0.040 &    0.244 f
U26/X (EDBLVT24_ND2_SKF_2)                              0.051 &    0.295 r
pmon_ref_counter_done (out)                             0.013 &    0.308 r
data arrival time                                                  0.308

max_delay                                               1.000      1.000
clock reconvergence pessimism                           0.000      1.000
clock uncertainty                                      -0.180      0.820
output external delay                                 -22.400    -21.580
data required time                                               -21.580
-----------------------------------------------------------------------------
data required time                                               -21.580
data arrival time                                                 -0.308
-----------------------------------------------------------------------------
statistical adjustment                                  0.000    -21.888
slack (VIOLATED)                                                 -21.888
```

Fig.6.16. A Path Report showing negative Slack

Path reports are used for Static Timing Analysis and fixing timing violations. The command used to report the worse-case timing path(s) in the design is **report_timing**. A large number of options can be used with this command to customize the timing report. This includes specifying start point, end point, cross talk information, and number of worst paths, min/max violations (min refers to Hold Violation and max refers to Setup Violation). By default, the command is used to fetch the path taking the maximum negative slack in every clock group.

Please note that the paths reports are different for Setup and Hold validations of the same path. Path reports show by how much time the slack for a particular path is met or violated. The “Incr” column in path report consists of contribution of corresponding cells in arrival time or required time. Several commands are used during Static Timing Analysis for checking and fixing violations.

```
Point                                                    Incr       Path
---------------------------------------------------------------------------
clock CLK_REFCLK (rise edge)                           0.000      0.000
clock network delay (propagated)                       0.277      0.277
refclk_counter/inst_stx_sync_resetn_rstn/demet_vndr_sync_level_KEEPM/genblk1_creq_demet_vndr_sync_level_ar/CK (EDBLVT24_SYNC2RBMSFQ_2)
                                                       0.000      0.277 r
refclk_counter/inst_stx_sync_resetn_rstn/demet_vndr_sync_level_KEEPM/genblk1_creq_demet_vndr_sync_level_ar/Q (EDBLVT24_SYNC2RBMSFQ_2)
                                                       0.090 &    0.368 r
refclk_counter/inst_stx_sync_resetn_rstn/EDBLVT16_INV_2_meta_out_dont_touch_KEEPM/X (EDBLVT16_INV_2)
                                                       0.018 &    0.385 f
refclk_counter/inst_stx_sync_resetn_rstn/EDBLVT16_INV_8_meta_out_dont_touch_KEEPM/X (EDBLVT16_INV_8)
                                                       0.007 &    0.392 r
refclk_counter/U39/X (EDBSVT20_MUX2_2)                 0.166 &    0.558 r
refclk_counter/refclk_count_reg_2/RD (EDBSVT20_FSDPRBQO_V2_2)
                                                       0.001 &    0.559 r
data arrival time                                                 0.559

clock CLK_REFCLK (rise edge)                          32.000     32.000
clock network delay (propagated)                       0.293     32.293
clock reconvergence pessimism                          0.016     32.309
clock uncertainty                                     -0.180     32.129
refclk_counter/refclk_count_reg_2/CK (EDBSVT20_FSDPRBQO_V2_2)    32.129 r
library recovery time                                 -0.019     32.110
data required time                                               32.110
---------------------------------------------------------------------------
data required time                                               32.110
data arrival time                                                -0.559
---------------------------------------------------------------------------
statistical adjustment                                 0.010     31.560
slack (MET)                                                      31.560
```

Fig. 6.17. A Path Report showing positive Slack

6.7. Fixing Timing Violations

Fixing Setup and Hold Violations have different ways because different equations of Setup and Hold Time calculations.

Techniques to Fix Setup Time Violations are described below:

i. **Removing buffers in the launch path:**

Removing buffers will not affect the logic. Removing buffers in launch path will decrease overall path delay of launch path.

ii. **Replacing Buffer with two Inverters:**

Replacing Buffer with two Inverters may reduce the overall stage delay. Adding inverter decreases the transition time 2 times then the existing buffer gate. Due to that, the RC delay of the wire (interconnect delay) decreases. This is because charging of parasitic capacitor leads to more charging time.

iii. **HVT/LVT swap:**

HVT, LVT, etc. refer to different cells in standard library with different threshold voltages FETs. In HVT, SVT, LVT, ULVT. Here VT stands for Threshold Voltage (Vth). Thus HVT, SVT, LVT, ULVT refer to High, Low, Standard and Ultra –Low Threshold Voltage respectively.

HVT/LVT swap method includes replacing HVT cells by SVT, LVT or ULVT. The propagation delay drops with lessening threshold voltage due to fall in the transition time. HVT/SVT/LVT cells have same pin location and equal area. Small Vth leads to reduction in transition time. Consequently, propagation delay decreases. Subsequently replacing HVT with LVT will make the timing faster without distressing the layout, but drawback is upsurge of leakage current

- HVT (High Threshold Voltage) cells are used in the paths where timing constraints are not just met (by negligible margin). So by using HVT cells power is saved
- SVT (Standard Threshold Voltage) cells have medium delay and medium power constraint. In case timing constraint fails by small margin while using HVT, they are replaced by SVT cells and timing is checked again.
- LVT (Low Threshold Voltage) are mainly applied in paths that meet timing constraints by negligible margin. These cells are fast but, ingest more power due to comparatively higher leakage current. These cells are used only when timing is critical.

iv.**Inserting buffers for clock skew:**

Inserting buffers will not affect the logic. Inserting buffers in capture path will increase overall path delay of capture path. This delay will add to clock skew, thus relaxing the Setup Time Condition.

v. Inserting two inverters for clock skew:

Inserting even number of inverters in cascade will not affect the logic. Inserting two inverters in capture path will increase path delay of capture path. Thus clock skew will increase, thus relaxing the Setup Time Condition.

Practices to Fix Hold Time Violations are described below:

i. **Adding Delays in the launch path:**
   Introducing buffer cells or even number of NOT gates to the data path assistances to fix the Hold Violation. This is contrasting to similar method in fixing Setup Violations.

ii. **Reduction in size of elements of data path:**

   HVT/LVT swap is also used to fix Hold Violations. Violations For example, changing HVT cells into SVT or LVT or ULVT.

A path with hold violation can have some of cells in it also in paths vulnerable to setup violation. Thus fixing Hold violations may lead to Setup Time Violations and vice versa.

Swapping cells or adding/removing Buffers/even number of NOT gates in clock path is avoided as same clock pin may be clocking other sequential cells in other blocks of deisgn. Changing or removing cells in clock path may lead to Setup/hold Violations in other paths of design. Once all timing violations of a block have been fixed then the block meets its timing constraints. To use Extracted Timing Model of a block or design its Static Timing Analysis is obligatory. An Extracted Timing Model (ETM) is in .lib (Liberty library cell) format. Synopsys Prime Time Tool is used to synthesize Gate-level Netlist.

6.8. Verification of Extracted Timing Model

Extracted Timing Model needs to be verified for many attributes before it can be used in place of the corresponding block in a larger design.

Need for Automated ETM Verification:

In digital designing, designs are often used as a part of larger design, for example SDRAM controller is designed separately and used with SDRAM as a whole. Extracted Timing Model of a design can be used to model its timing relationship in timing analysis of larger design. This

removes the need of timing analysis of smaller design again while performing timing analysis of larger designs. For this, verification of Extracted Timing Model of a design is required if the design is to be used in a larger design. This verification is currently done manually.

Time taken for manual Verification of Extracted Timing Model varies form hours to days depending on the design. Most commercial digital designs have thousands of pins, which will consume much time, resulting in large time in Sign-off the design for fabrication. Thus Verification of Extracted Timing Models must be automated in way it works for all designs.

Algorithm for Verification of Extracted Timing Model:

The Extracted Timing Model may not describe logical relation between input and output of design or between internal pins. It only represents the timing relations. Following information is checked in Verification of Extracted Timing Model.

i. The input, output and inout pins of the design.

ii. Connections of input, output and inout pins to/from internal pins of the design.

iii. For every timing arc to/from input, output or inout pin it is checked if it is setup arc or hold arc.

iv. Unateness of these arcs (arcs (ii) above).

v. Clock gating pins in input/inout pins or soucred by these pins.

The flow chart for Automated Verification of Extracted Timing Models is shown in Fig.6.21. It is independent of the circuit design of which ETM is to be verified.

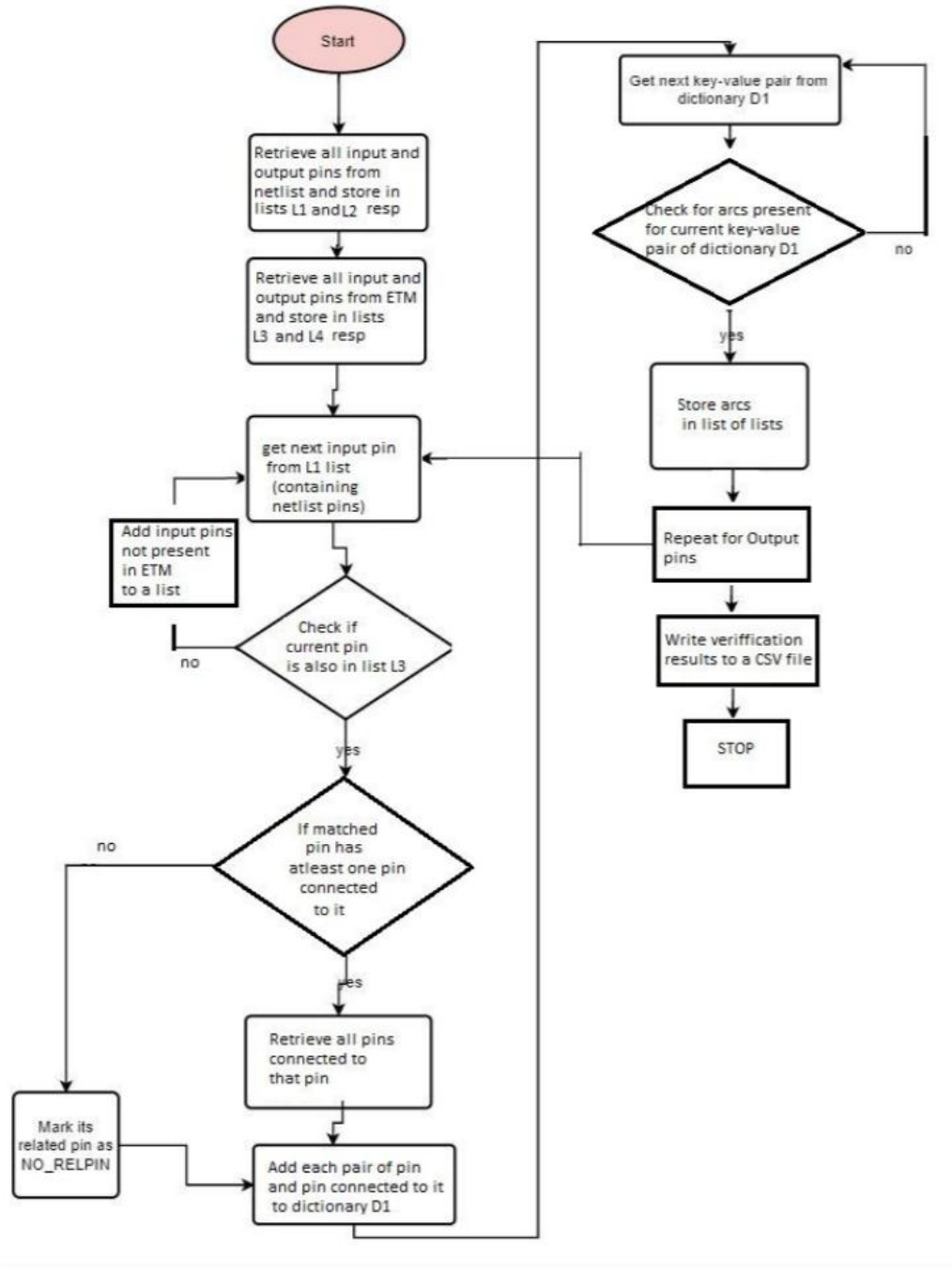


Fig.6.18. Flow Chart for Verification of Extracted Timing Models

Fig. 6.20. Screenshots of Implementation of Automated ETM Verification

Chapter 7

Results and Discussion

The ETM Verification Algorithm implementation is independent of the circuit. It takes the synthesized Gate-level netlist and corresponding Extracted Timing Model as inputs for verification. This automated verification method been tested on five different designs apart from SDRAM Controller at Seagate Technologies and has given correct results. This chapter also contains screenshots of SDRAM Controller simulation, its synthesized gate level netlist and Static Timing Analysis of the SDRAM Controller.

7.1. Simulation and Functional Verification of Approximate MAC Unit

The simulation and functional verification of Approximate MAC Unit was done using Altera Modelsim Student Edition. Verilog HDL was used to design the SDRAM Controller circuit.

Simulation result in Fig.7.1 of Approximate MAC Unit shows that multiplication of decimal equivalent 13107 and 13629 followed by addition of 3895 (decimal eq.) results in 178492160 (decimal eq.) which is approximately equal to 178639198 (decimal eq.). Several such simulation results were used to verify that the 16 bit Approximate MAC Unit designed gives results very close to 16 bit conventional multiplication-accumulation unit.

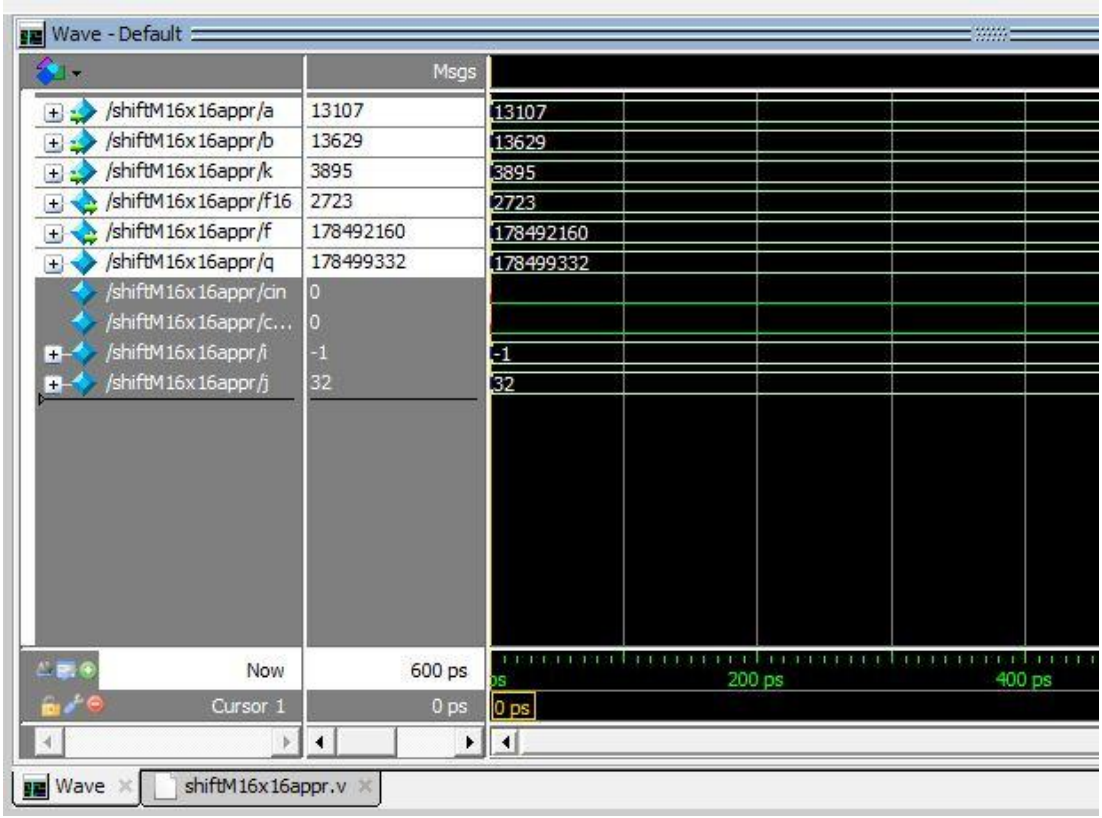


Fig.7.1. Simulation Result of Approximate MAC Unit

7.2. Simulation and Functional Verification of SDRAM Controller:

The simulation and functional verification of SDRAM Controller was done using Altera Modelsim Student Edition. Verilog HDL was used to design the SDRAM Controller circuit.

The write operation in fig 2. has data to be written to SDRAM by SDRAM Controller is send via sdramDQ port. The read operation in fig 3. has data read from SDRAM by SDRAM Controller is output at port rData.

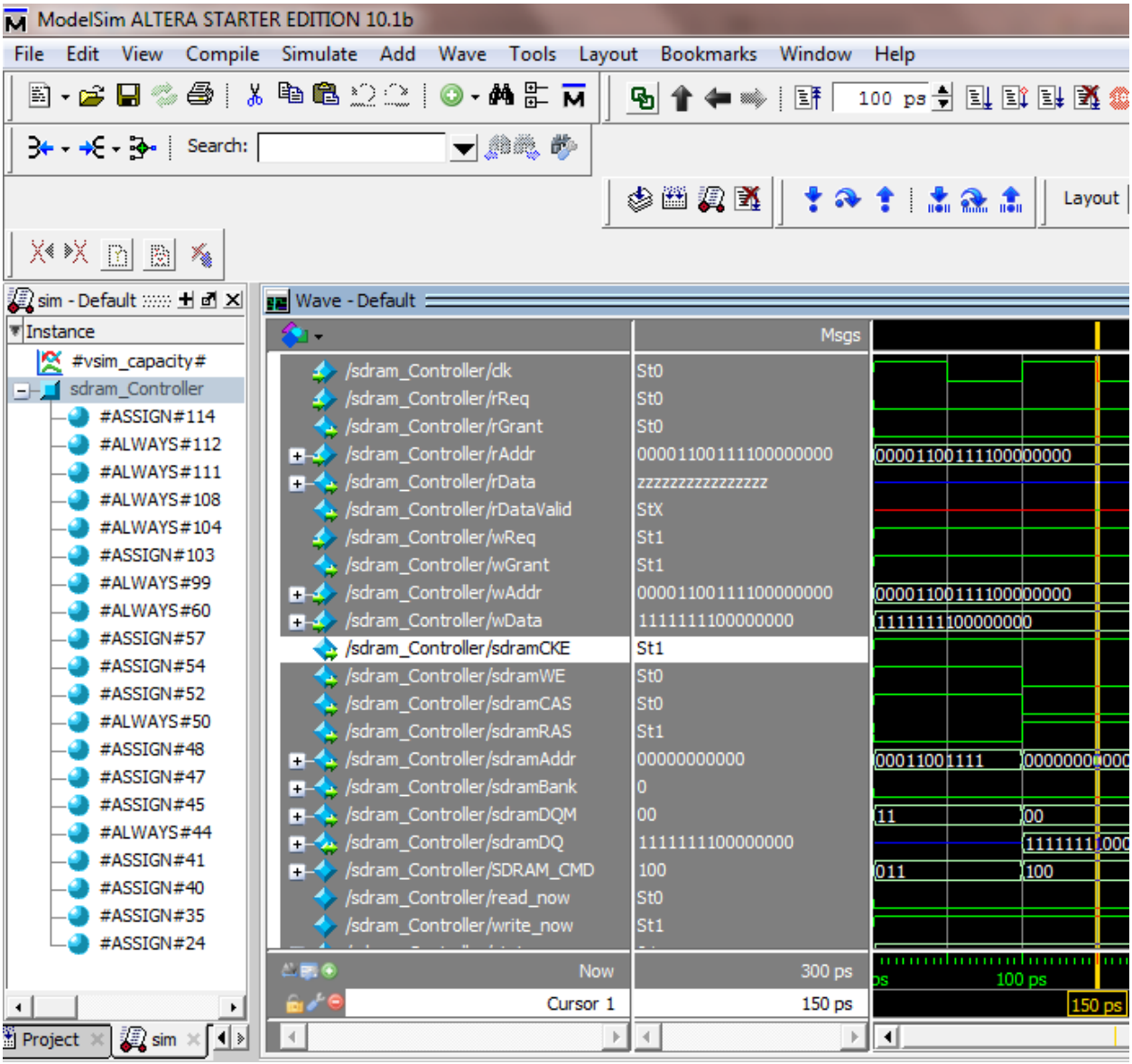


Fig 7.2. Simulation of Write operation of SDRAM Controller

The write operation shown above consists of successful writing of data input (to SDRAM Controller) via 20 bits wData port to sdramDQ. The wReq bit is one for write request. This action takes two clock cycles as shown. The data is sent from SDRAM Controller to SDRAM through sdramDQ. The status of SDRAM_CMD equal to 011 shows writing operation.

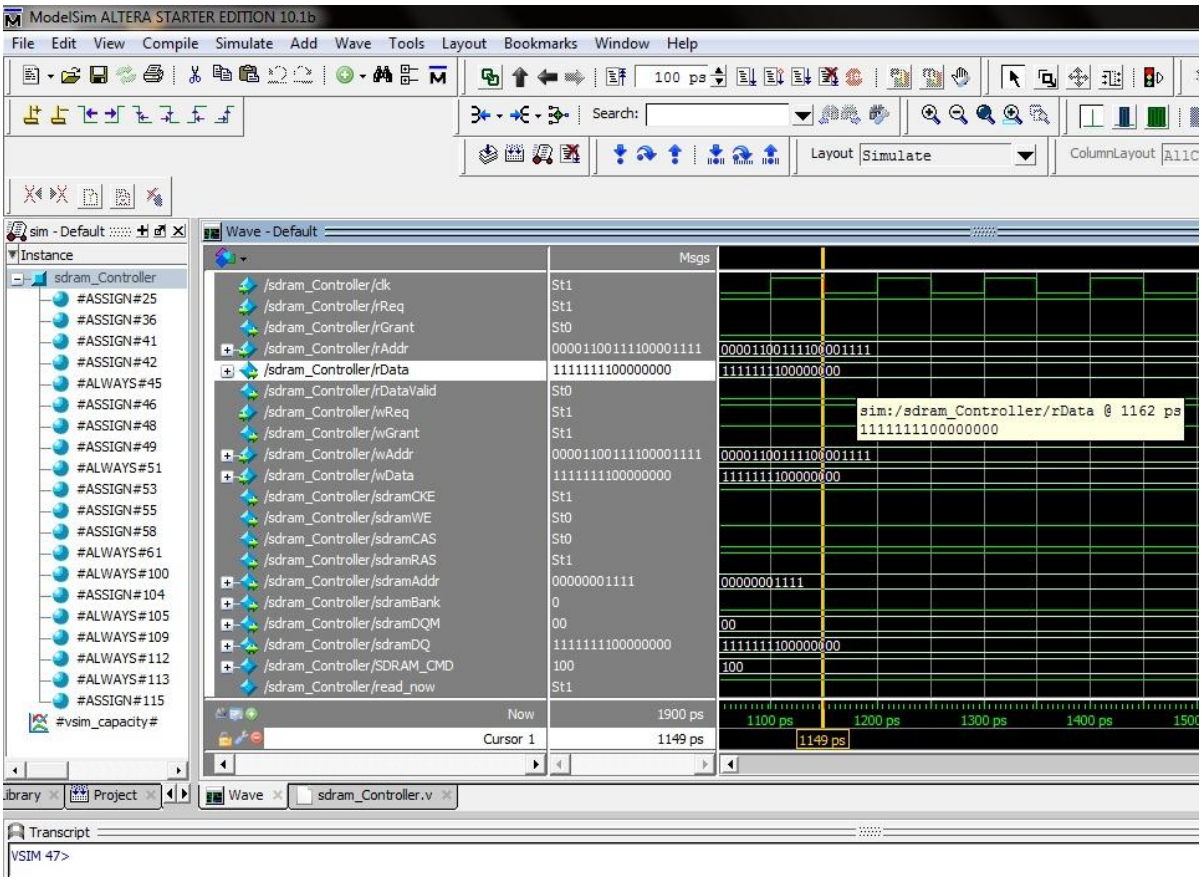


Fig 7.3. Simulation of Read operation in SDRAM Controller

The read operation during both bits rReq and wReq equal to 1 shows that read request dominates over write request. The reading operation shown above consists of successful reading of data (from SDRAM Controller) via 20 bits rData port to sdramDQ. The data is read by SDRAM Controller from SDRAM through sdramDQ. The status of SDRAM_CMD equal to 100 shows reading operation. The data is read from same location to which data was written to.

7.3. ETM Verification Results

Tool Command Language is used for implementing the algorithm of automated verification. The ETM Verification Results of any design are written in comma separated values format files.

The ETM Verification results of each of these designs are written to .csv files. They have following information:

i. Input, Inout and Output Pins

ii. Timing Arcs Information of arcs (between pins of ports and connected to internal pins):

- Unateness of Arcs
- Combinational Timing Arcs
- Setup/Hold Arcs
- Arcs to/from Clock Gating pins.

It also checks if any pins or arcs between pins are missing.

1. **Results of ETM Verification of Approximate MAC Unit:**

Design: ApproximateMAC				
Netlist_Input_Pins	Present/Absent in Liberty library cell			
k[15]	present			
k[14]	present			
k[13]	present			
k[12]	present			
k[11]	present			
k[10]	present			

Fig.7.4 (a)

Input_Pin	Connected_Pin	Arcs
a[15]	f16[15]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinationa
a[15]	f16[14]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinationa
a[15]	f16[13]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinationa
a[15]	f16[12]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinationa
a[15]	f16[11]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinationa
a[15]	f16[10]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinationa

Fig.7.4 (b)

Output_Pin	Connected_Pin	Arcs
f16[15]	a[0]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combination
f16[15]	a[10]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combination
f16[15]	a[11]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combination
f16[15]	a[12]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combination
f16[15]	a[13]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combination

Fig.7.4 (c)

Fig.7.4. Some Screenshots of ETM Verification Output of Approximate MAC Unit

2. Results of ETM Verification of SDRAM Controller:

1	Design: sdram_Controller	
2		
3		
4	Netlist_Input_Pins	Present/Absent in ETM
5	wReq	present
6	rReq	present
7	clk	present
8	wData[15]	present
9	wData[14]	present
10	wData[13]	present
11	wData[12]	present

Fig.7.5 (a)

Input_Pin	Connecte	Arcs
rReq	rGrant	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinational ;} {timi
rReq	wGrant	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinational ;} {timi
rAddr[19]	rGrant	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinational ;} {timi
rAddr[19]	wGrant	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinational ;} {timi
rAddr[18]	rGrant	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinational ;} {timi
rAddr[18]	wGrant	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinational ;} {timi
rAddr[17]	rGrant	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinational ;} {timi

Fig.7.5 (b)

Output_Pin	Connected_Pin	Arcs
rGrant	rAddr[10]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinatior
rGrant	rAddr[11]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinatior
rGrant	rAddr[12]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinatior
rGrant	rAddr[13]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinatior
rGrant	rAddr[14]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinatior
rGrant	rAddr[15]	{{timing_type : combinational ;} {timing_sense : positive_unate ;}} {{min_delay_flag : true ;} {timing_type : combinational ;} {timing_sense : positive_unate ;}} {{timing_type : combinatior

Fig.7.5 (c)

Fig.7.5. Some Screenshots of ETM Verification Output of SDRAM Controller

Results of multiple designs, have been crosschecked manually. All final results have been found to be correct.

Advantages of this method of ETM Verification:

- The main advantage of this method of ETM Verification is reduction in time of Verification from hours or days (manual ETM Verification) to few minutes. This depends on various factors of the design.
- Short time to timing signoff of large integrated designs which are made using other designs as its blocks.
- The ETM Verification method is independent of VLSI corner of the ETM.

Hence, it is proposed that using ETM Verification method is significantly advantageous can be used in verification of commercial digital design extracted timing models.

Chapter 8

Conclusions and Scope for Future Work

This Extracted Timing Model Verification method been tested on two different designs:

1. Approximate MAC Unit (a Combinational Circuit)

   Also, Computation using this Approximate MAC Unit has following advantages:

   - Significantly less number of full adders used in comparison to Conventional MAC Unit.
   - Faster computation due to less number of full adders used.
   - Faster modelling of Artificial Neural Networks using Approximate MAC Units.

2. SDRAM Controller (a Sequential Circuit)

Apart from above designs, this method of ETM Verification has been used in some other commercial designs at Seagate Technologies. IT has been found by manual crosscheck of this ETM Verification on each of these designs that it has given correct results.

Advantages of this Automated ETM Verification Method:

Results of multiple designs, have been crosschecked manually. All final results have been found to be correct.

Advantages of This ETM Verification:

- Reduction in time required in Extracted Timing Model Verification from hours (manual ETM Verification) to few minutes. This depends on various factors of the design. This leads to shorter time to timing signoff of large integrated designs which are made using other designs as its blocks.
- It is independent of the HDL language used in Register Transfer Level of the design if the Gate-Level Netlist is synthesized in Verilog HDL.
- Much less time required when used during implementation (not sign-off) of IP models. The data in IP models is protected as it is in abstracted form with timing info, void of netlist.

Hence, following can be concluded from this work:

- Using ETM Verification method is significantly advantageous can be used in verification of commercial digital design extracted timing models.
- The designed Approximate MAC Unit shall be used in place of Conventional MAC unit in time critical applications.

Although the method of ETM Verification will significantly reduce the time in of chip design process, but it has some scope of improvement in following ways:

- It may be possible to further reduce the time taken in Automated Verification of ETM to few seconds.
- It may be possible that the number of gates required in implementation of Approximate MAC Unit can be reduced further.

REFRENCES

- Science Direct (2020), www.sciencedirect.com, https://www.sciencedirect.com/topics/computer-science/timing-constraint
- Bhasker, J., Chadha, Rakesh, (2009) Static Timing Analysis for Nanometer Designs, A Practical Approach, Springer Science + Business Media, Springer Street, New York, (ISBN).
- VLSI Junction (2015), vlsijunction http://www.vlsijunction.com, http://www.vlsijunction.com/2015/08/onchip-variation-ocv.html