IP QUALITY IMPROVEMENT USING SPYGLASS

Major Project Report

Submitted in partial fulfillment of the requirements

For the Degree of

Master of Technology

In

VLSI Design

(VLSI Engineering)

By

Lad Rachit Jayantibhai

(09MEC013)

Department of Electronics & Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2011

IP QUALITY IMPROVEMENT USING SPYGLASS

Major Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

In

VLSI Design Engineering

(VLSI Engineering)

By

Lad Rachit Jayantibhai

(09MEC013)

Under the Guidance of

Prof. Usha Mehta

Department of Electronics & Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2011

Declaration

This is to certify that

i) The thesis comprises my original work towards the degree of Master of Technology in VLSI Design at Nirma University and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

Lad Rachit Jayantibhai

Certificate

This is to certify that the Major Project entitled **"IP QUALITY IMPROVEMENT USING SPYGLASS"** submitted by **Lad Rachit Jayantibhai (09MEC013)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design of Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Prof. Usha Mehta
Associate Professor,
Department E & C Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Mr. Sheetal Kumar Jain
Sr. Design Engineer,
HED-HVD Design Team,
ST Microelectronics,
Greater Noida.

Dr. N.M. Devashryee
PG Cordinator ,VLSI Design,
Department E & C Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. A S Ranade
Head of Department,E&C Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr. K Kotecha
Director,
Institute of Technology,
Nirma University ,Ahmedabad

Abstract

Time to market & Power is becoming a critical design criterion for ASIC/SoC designers. Early Design Analysis find the issues which can pin points structural ,coding & consistency problem at RTL. Early Design Analysis can find most comprehensive design problems related to clock, reset & CDC(clock domain crossings) which cannot find in the verification of the design but this captures at the netlist level which will cause a design respin & TTM will increase. Early Design analysis finds such issues at RTL level, so it reduces the design respin time. To generate the netlist from the RTL designer needs the synthesis constraints. These constraints need to be accurate for the quality netlist. Early Design Analysis also helps to find fault in the synthesis constraints. Now Early Design Analysis also helps to generate the synthesis constraints automatically with the tool. This automation reduces the human efforts to generate constraints & generated constraints will have more accuracy. To meet power requirement of the design it is likely to have power estimation at the RTL level itself, so the designer can apply different techniques at RTL level to meet power requirement when chip will manufactured.

During this thesis, work based on the different steps of Early Design Analysis will be carried out on the different IP's with the the tool named **SPYGLASS**. Spyglass will observed problems related with the different IP's & will ensure that applied solution is feasible or not for the design at every stage of the Early Design Analysis. Also the comparision of the results will be performed for the features like constraints generation & power estimation will be carried out with some standard results.

Acknowledgements

I would like to express my gratitude and sincere thanks to Prof. A. S. Ranade Head of Electrical Engineering Department and Dr. N. M. Devashrayee Coordinator M.Tech VLSI Design Engineering program for allowing me to undertake this thesis work and for his guidelines during the review process.

I am deeply indebted to my thesis supervisor **Mr. Sheetal Jain & Mr. Vivek Mohan Sharma(Manager) at the ST Microelectronics** for their constant guidance and motivation. They have devoted significant amount of their valuable time to plan and discuss the thesis work. Without their experience and insights, it would have been very difficult to do quality work. I also thank ST Microelectronics organisation for providing me an opportunity to work in its environment and carry out my thesis work.

I wish to thank my friends of my class for their delightful company which kept me in good humor throughout the year.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

Lad Rachit Jayantibhai

09MEC013

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 What is Early Design Analysis

Early Design Analysis is a technique to analyze the design at RTL(Register Transfer Level) level to check the issues like connectivity, synthesizability & also implementation issues like CDC(clock domain crossing),power related clock gating, some timing exception ,DFT(Design for Testability) set up. So this checks ensures the designer that what changes designer has to make at RTL level so that these issues can be solved at RTL level only so that designer can reduce the implementation time of the design by avoiding the iterations. For that we are using tool called SPYGLASS which can do Early Design Analysis effectively.

1.2 Need of Early Design Analysis

RTL designers need to make decisions that impact the power, timing and physical feasibility of a chip during the downstream implementation flow. Since RTL designers have limited visibility into physical implementation, the impact of these decisions are resolved late in the process. The result can be several iterations between back-end and front-end designers, a costly problem. So Early Design Analysis provides a early estimates of area, power, timing and routability for RTL designers

without the need for physical design expertise or tools. For that I have used Spyglass tool for the early design analysis. With SpyGlass, RTL designers can make sure RTL blocks will be easier to implement and will meet SoC design goals like area requirement, timing requirements, power estimation etc of the physical chips. So for to solve the such issues Early Design Analysis is done.

1.3 Key features of Spyglass[3]

- Full language support for Verilog (IEEE 1364, Verilog 2001, and SystemVerilog) and VHDL (VHDL-87 and VHDL-93).
- A rich suite of built-in rules, including.
- File checks, such as file names, design units per file and headers.
- Naming checks on signals, ports, parameters, constants, clocks and other constructs.
- Style and related checks.
- Coding for synthesis and related checks.
- Design practice and related checks.
- Area, timing and synchronization checks.
- Clock and reset checks.
- DFT, LowPower, Constraints, ERC and similar checks (cost options)

- A variety of report format options so you can set up your own reports
- Built-in engines, including RTL synthesis and flattening, to enable detailed implementation tests including clocking, reset and synchronization of asynchronous signals.

1.4 Key Benefits of the Spyglass[3]

- Provides relative quantitative measure of implementation readiness of RTL blocks Facilitates tracking block implementation readiness progress as part of regression setup
- Very fast analysis to enable multiple iterations within a day leads to faster design closure
- RTL designers can take corrective action based on Rules and reports without having to learn physical aspects or develop tool expertise
- Integrated into SpyGlass platform Offer physical rules like other SpyGlass tools
- Rich set of visualization, what-if analysis, reports and metrics enable easy to use floorplanning for complex IP and SoC's

In Early Design Analysis there are some checks can be performed at the RTL.

1.5 Early Design Analysis Flow

Early Design Analysis has a 4 basic steps to ensure that design is good for synthesis. These steps are,

a. RTL Checks: It is a method to perform a different checks to on the RTL design to catch the issues which will not have the impact on the simulation i.e. verification but will cause the problem after the synthesis when the gate level netlist is generated. The problem like mentioned below.

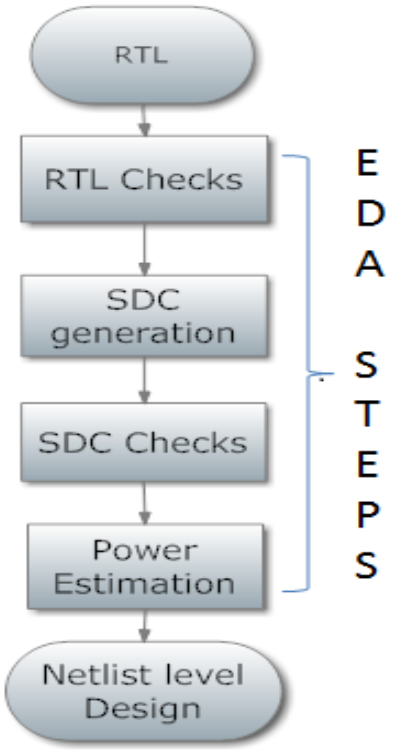

Figure 1.1: Early Design Analysis Flow

- **CDC(Clock Domain Crossing):** Such issues can not be detected at the RTL level. Verification also can not detect such a problem.This problem can be detect in the synthesis with timing violation. So to solve the problem reiterative process has to done after synthesis. These causes longer design time.To save this design time it is good to have a method that can detect such a problem at the RTL level.So designer can solve the problem at the early stage of the design. That causes reduction in the iterative time.So RTL checks is the important part of the Early Design Analysis which can reduce the design time drastically and ensures the functionality of the design.

  e.g Clocks that are asynchronous with respect to each other may reach different flops at slightly different times in each cycle during the execution of the design. This timing uncertainty may cause setup and hold-time vi-

olations randomly in the design. Setup and hold time violations can cause functional failure in the chip. This issue cannot be identified using traditional verification methods, such as simulation and static timing analysis. Static clock domain crossing analysis and verification is the only way of verifying CDC correctness. The need is to detect clock domain crossing at RTL level and make sure proper synchronization has been added in the circuit.

- Reset asynchronization problem can cause the functionality failure.
- Linting Checks: Linting checks helps the designer to find that RTL is synthesizable or not. Also due to RTL is following the standard coding guidelines there may be simulation-synthesis mismatch can occur. So may be the further implementation issues can cause failure in functionality. Linting checks also helps to solve such issues in the RTL.
- DFT Checks Manufacturing test is performed by patterns automatically generated by ATPG (Automatic Test Pattern Generation) tools. To operate effectively, these tools require that the circuits be correctly designed for testing.

b. GENSDC :

In a typical design environment, SDC files are used and modified at various stages of the design (e.g. RTL, pre-layout, post-layout). Generally these SDC files are created either manually by the Designer or with help of some automated ad-hoc scripts. It's very difficult to avoid mistakes (human error) in SDC written manually. Gensdc tool has the capability to understand the design & generate the constraints automatically. So the generated constraint is accurate & free from human errors.

c. **SDC(Synopsys Design Constraints) Checks:** IC designs go through several transformations in a typical RTL-to-layout flow, and as they do, a number of verification steps (simulation, equivalence checking, etc.) are performed to

ensure that the design intent is preserved. Timing constraints, too, are created at RTL and refined throughout the design cycle. But, traditionally, there has been no step for validating timing constraints at various stages of design. The creation and refinement of constraints has been largely a manual, error-prone and time-consuming process. Managing thousands of lines of constraints throughout the flow is nearly impossible. As a result, constraints issues are a major cause of additional iterations during implementation. According to Electronic Design News, a typical design project goes through 10 or more iterations due to refinement of timing constraints. Poor constraints impact the chip quality in terms of area, power, and timing. As a result, timing closure takes longer. Worst of all, incorrect constraints can result in silicon failing timing and hence a re-spin. There is thus a critical need for an EDA solution that ensures valid timing constraints throughout the design flow. **e.g** IC designs go through several transformations in a typical RTL-to-layout flow, and as they do, a number of verification steps (simulation, equivalence checking, etc.) are performed to ensure that the design intent is preserved.

d. Power Estimation & Reduction Power reduction is becoming a critical design criterion for ASIC/SOC designers. Reducing both dynamic and leakage power is imperative to meet power budgets for portable devices as well as to ensure that the systems that these ASICs meet their packaging and cooling costs. In addition, the power of an ASIC has a significant impact on its reliability and manufacturing yield. Traditionally, most automated power optimization tools have focused at gate-level and physical level optimizations. However, major power reductions are only possible by addressing power at the RTL and system levels. At these levels, it is possible to make the sequential modifications needed to reduce power and energy consumption via techniques like sequential clock gating, power gating, voltage/frequency scaling and other micro-architectural techniques. The focus of this tutorial will be on techniques for power reduction

at the RTL and system level. It will also focus on expressing power intent at system and RTL levels and the flows needed to use that power intent in tools for functional verification, RTL-level optimization, logic synthesis and physical design. The following sections describe the key focus areas in the tutorial. Since power of a design is a function of how it performs a computation over time, almost all the major transformations that have significant impact on the power of a design are sequential in nature - they change the sequence of values generated at key internal registers or memories in time. We will discuss the sequential optimizations like, sequential clock gating, power gating, dynamic voltage scaling and memory banking. The impact of these optimizations on verification and implementation flows will be highlighted and solutions to verification and implementation issues will be presented. In the last few years, standards have started emerging to allow designers to express power intent such as voltage islands and power modes in a design. These are allowing for the same power intent to be seen by all the tools in the RTL flow: RTL simulation, logic synthesis, place and route, logic equivalence checking and any other post-layout tools. Both CPF and UPF attempt to specify this information. We will focus on the key information that the emerging power formats need to support and how this impacts the RTL design and implementation flow. Specific power optimizations enabled by such information in implementation tools will also be discussed.

1.6 Thesis Organization

In chapter 2 describes the different clock, reset CDC & linting issues related to the RTL.

In chapter 3 different "Synthesis Design constraint" is described which will have the impact & importance of the synthesis. & also how the automated constraints can generate with the GENSDC.

Chapter 4 describes the need & importance of SDC checks with some examples.

In chapter 5 how power estimation can be done at the RTL level. It also describe that the power estimated at the RTL level the can be optimized that it has a close approximation when the design is fabricated from silicon.

Finally, in chapter 6 concluding remarks is presented.

Chapter 2

RTL Checks

2.1 Introduction[5]

Shrinking device sizes and increasingly complex designs have created multimillion-transistor systems running with multiple asynchronous clocks with frequencies as high as multiple gigahertz. SoC systems have multiple interfaces, some using standards with very different clock frequencies. Several modern serial interfaces are inherently asynchronous from the rest of the chip. There is also a trend toward designing major sub-blocks of SoCs to run on independent clocks to ease the problem of clock skew across large chips. Design methodologies have traditionally focused on partition-based implementation and verification. Often these partitions are based on clock domains. The cross-clock domain crossing (CDC) signals pose a unique and challenging issue for verification. Traditional functional simulation is inadequate to verify clock domain crossings. While static timing analysis (STA) is an integral part of the timing closure solution, little attention has been paid to addressing proper clock domain implementation and verification. Existing methods provide an ad hoc partial verification that is manual, time consuming, and error prone. If the sources of potential errors are not addressed and verified early on, designs can end up with functional errors that are only detected late in the design cycle - or even worse, during post-silicon verification.

The cost of fixing errors at this stage is enormous.

- Structural issues (sCDC): If the data input to a storage element changes too close to a clock edge, the element may go into a metastable state and the output cannot be reliably used. Asynchronous clock domain crossings are particularly prone to metastability failures. To address this issue, the circuit must be designed to "buy time" so the metastable signal can settle to a stable value, typically using synchronizers. After completing the synchronization, the structures beyond the synchronizers still matter. For example, the design must ensure that the synchronized signals do not converge. Reconvergence can create functional errors.

- Functional issues (fCDC): Designers must ensure that the stability and functionality on either side of the CDC circuit are handed over properly. Otherwise, there could be loss of signal values for signals passing between clock domains, with data instability in the receiving clock domain. Only automatic formal verification techniques can ensure that multiclock designs are correct prior to tapeout. The CDC verification solution must address this verification challenge, while maximizing overall productivity and effectiveness. The CDC solution needs to cover clock domain analysis and structural and functional verification, addressing both register-transfer-level (RTL) and gate-level verification needs.

My thesis work is mainly limited to different structural checks which can ensure correctness of the structural errors of the RTL. In RTL checks there are different issues which is bening performed by the Early Design Analysis Tool(Spyglass).

2.2 CDC Checks

The design issues and challenges of handling the signal crossing domains will be discussed later in this thesis. First, look at some CDC basics.

2.2.1 CDC Domains

A clock domain is defined as that part of the design driven by either a single clock or clocks that have constant phase relationships. A clock and its inverted clock or its derived divide-by-two clocks are considered a clock domain (synchronous). Conversely, domains that have clocks with variable phase and time relationships are considered different clock domains. In Figure 2.1, the design has a single clock domain because the gCLK is the derived divide-by-two clock of the master clock CLK. In Figure 2.2,

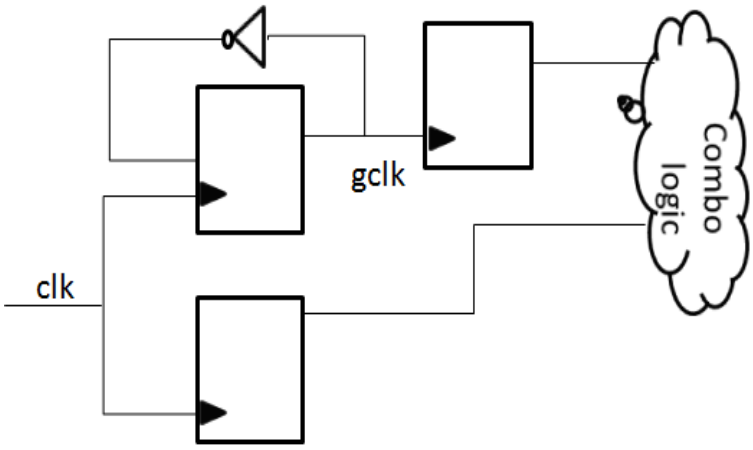

Figure 2.1: Single clock domain

multiple clocks come from different sources. The sections of logic elements driven by these clocks are called clock domains, and the signals that interface between these asynchronous clock domains are called the clock domain crossing (CDC) paths. The DA signal is considered an asynchronous signal into the clock domain - no constant phase and time relationship exists between CLK A and CLK B.

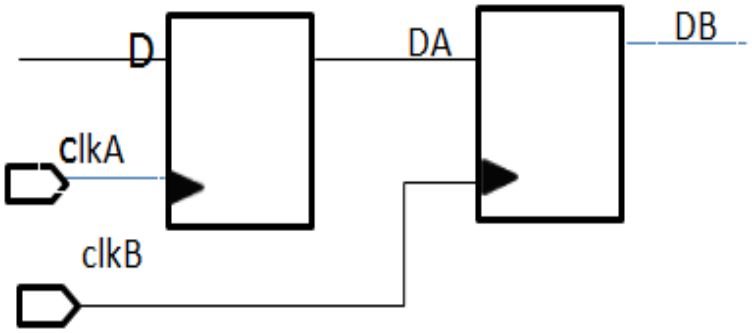

Figure 2.2: The CDC path

2.2.2 Asynchronous Signal Issues

In the multiple clock domain if the signal is not synchronized than there can be some issues that can designer face due to unsynchronized crossing.

These issues are:

a. Metastability :

The transaction of data violated the setup or hold time of the destination FF. It caused the ouput may oscillate for an indefinite amount time. The Metastability may lead to the high current or even burn out of the chip. It also caused functional issue and timing issue.

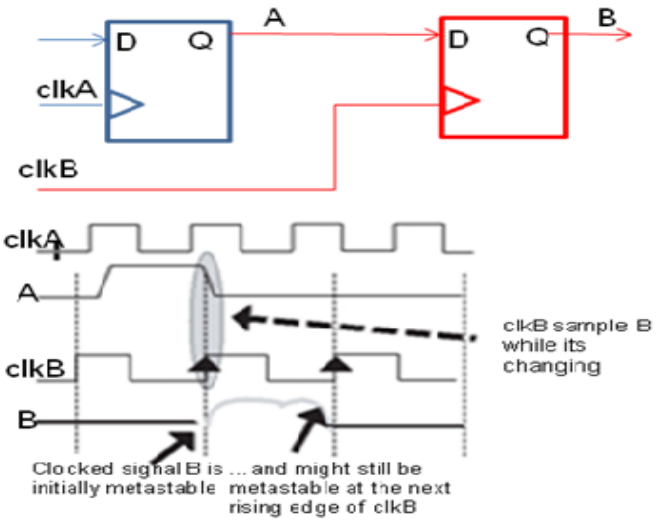

Figure 2.3: Metastability[1]

b. Data Loss:

This problem arises when a short pulse generated in a fast clock domain is fed into a slow clock domain. Under such circumstances the short signals may miss the active edge of the slow clock domain and will not be captured in the destination. The following figure illustrates the data-hold problem in fast-to-slow crossings.

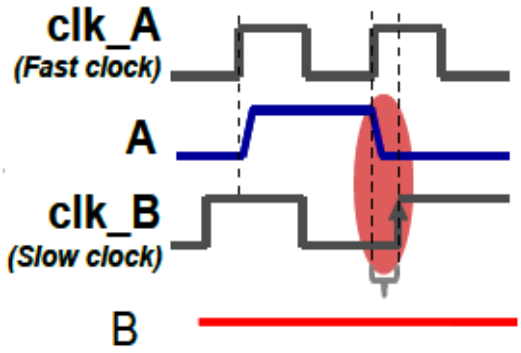

Figure 2.4: Data Loss[4]

Typically, a handshake or custom circuit is used to extend the pulse for at least one full cycle of the slow destination clock. All fast-to-slow crossings need to be functionally verified to make sure that such extenders exist and no short pulse

is generated in the destination.

c. Data Coherency:

Whenever new data is generated in the source clock domain, it may take 1 or more destination clock cycles to capture it, depending on the arrival time of active clock edges due to slow to fast clock domain crossing. This situation can be avoid using Synchronizer.

2.3 Synchronization Techniques

To avoid problems due to unsynchronized crossing on the crossing path the different synchronizer is used. There are four main techniques for the synchronizer used by the designer as mentioned below.

2.3.1 Multiflop Synchronizer

The synchronizers as shown in fig 2.5 allow sufficient time for the oscillations to settle down and ensure that a stable output is obtained in the destination domain. A commonly used synchronizer is a multi-flop synchronizer as shown in Figure 2.5.

2.3.2 Enable based Synchronizer

This scheme marks those clock crossings as synchronized where the first flip-flop in the destination clock domain is enabled by a signal synchronized to the destination clock and the clock crossing is in the data path as shown in fig 2.6. The clock domain crossing is marked as synchronized if either of the following conditions is met:

- The enable pin is driven by a signal synchronized to the destination clock domain.

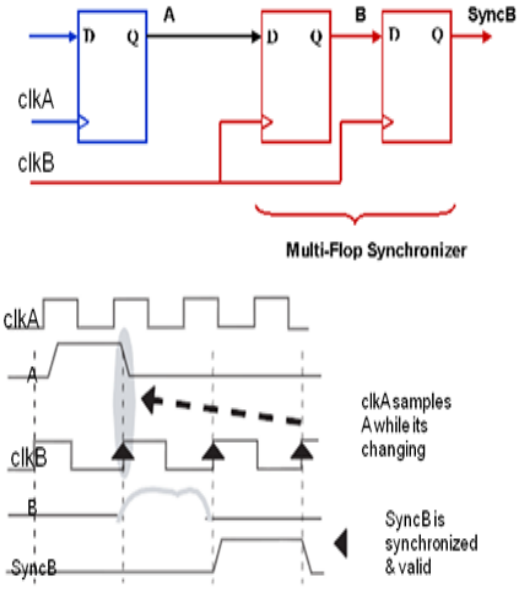

Figure 2.5: Multiflop Synchronizer[1]

- A valid synchronizer exists in any one of the paths driving the enable pin and signals in all other paths are either driven by primary ports or the destination clock domain flip-flops and there is no unsynchronized crossing in any of the paths.

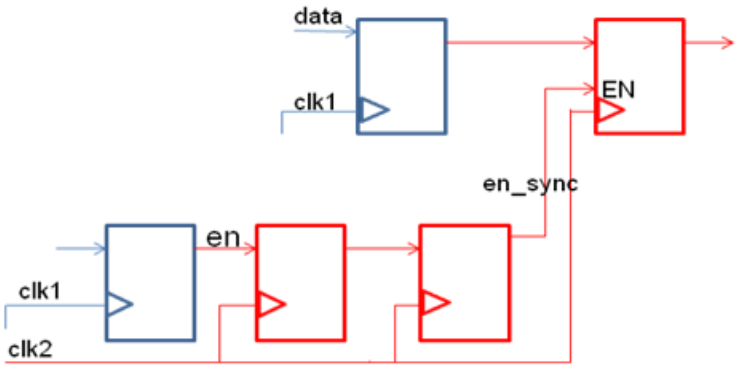

Figure 2.6: Enable based Synchronizer

2.3.3 Recirculation Synchronization Scheme

This scheme marks those clock crossings as synchronized where the first flip-flop in the destination clock domain is driven by a MUX. The clock domain crossing happens through one of the MUX input pins and the other MUX input pin is driven by the destination flip-flop output.

As shown in figure 2.7, a control signal EN, generated in the source domain is synchronized in the destination domain using a multi-flop synchronizer. The synchronized control signal EN_Sync drives the select pin of the muxes, thereby controlling the data transfer for all bits of the bus A. In this way, individual bits of the bus are not synchronized separately, and hence there is no data incoherency. However, it is important to ensure that when the control signal is active, the source domain data A[0:1] should be held constant.

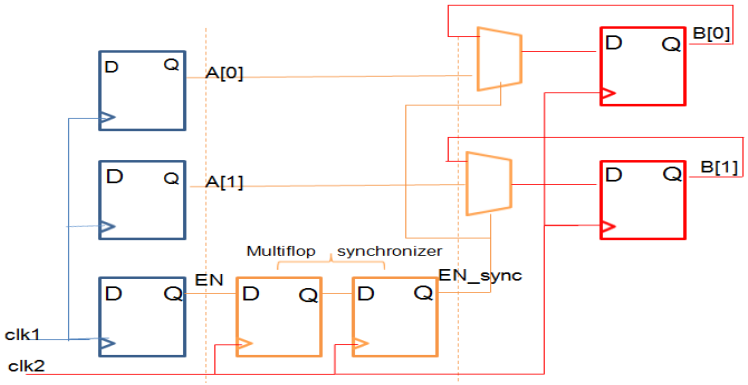

Figure 2.7: Recirculation Mux based Synchronizer

2.3.4 Handshake Synchronization

This scheme marks those clock crossings as synchronized that match the Handshake Synchronization configuration as in the following figure 2.8. Here the synchronization between **FSM1** & **FSM2** is done by control signals such as **req ,ack**.

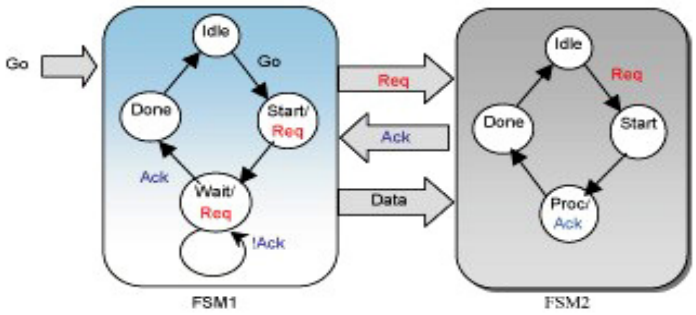

Figure 2.8: Handshake Synchronization[4]

2.4 Reset Assertion

A fully asynchronous reset is one that both asserts and de-asserts a flip-flop asynchronously. Here, asynchronous reset refers to the situation where the reset net is tied to the asynchronous reset pin of the flip-flop.

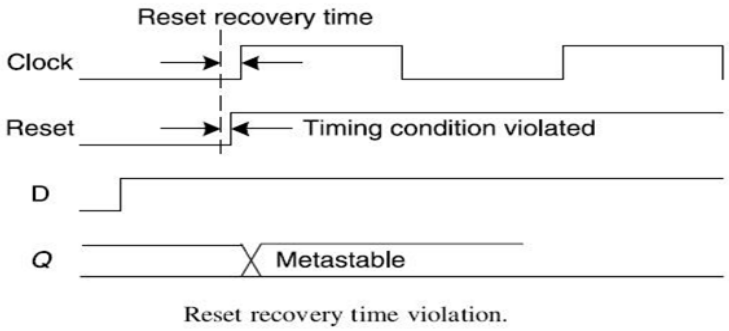

Figure 2.9: Reset Recovery time violation[8]

Additionally, the reset assertion and de-assertion is performed without any knowledge of the clock. The biggest problem of above described circuit is that it will work most of the time. However if the edge of the reset deassertion is too close to the clock edge & violate the reset recovery time than output will goes in to metastable as shown in figure 2.9.

The reset recovery time is a type of setup timing condition on a flip-flop that defines the minimum amount of time between the assertion of reset and the next rising clock

edge as shown in figure 2.10.

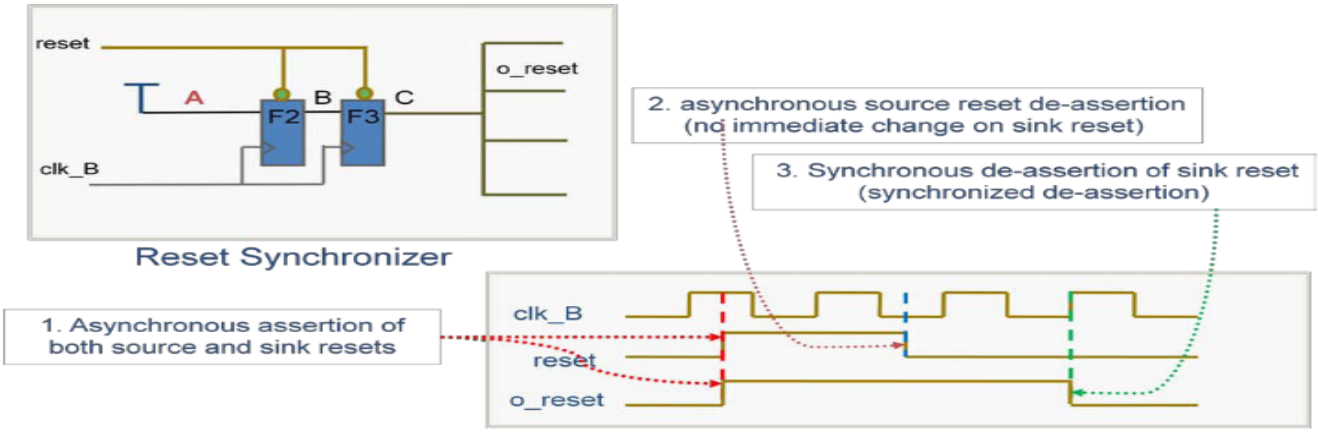

Figure 2.10: Reset Deassertion Synchronization[3]

2.5 Linting Checks

In general Lint tools that flag suspicious and non- portable usage of language construct in any programming language. It points out the code where it likely to be bugs. In chip design world Lint tools (some time referred as Design Rule Checker) check the cleanness and portability of the HDLs code for various EDA tools. Usually compiler does not show the errors and warnings which detected by lint tools. There are many advantages of it, for example when design style enforced by lint tool it avoids a situation where the synthesis tools implements something different than expected from RTL code.

e.g Consider the following Verilog code

```
input a;
input b;
output c;
assign c= a+b;
```

This example is completely ok with compiler. Complier does not generate warning.

In the above code by default the output variable 'c' is a wire. It is not necessary declare it as a wire. But some (not every) synthesis tools may need it to be declared it as a wire.

Here is the another example presents which reprsents the simulation-synthesis mismatch. If one process is defined as below,

Process(a,b,c)
e <= a and b;
f <= c and d;
g <= e and f;
end process;

Here if 'd' is changes from '1' to '0', then there may be no impact on the simulation results , but in the synthesis it will simply generate the one AND gate for "f<=c and d" . So as the input 'd' to '0' then ouput would be '0'. So it will directly affect the outputs if initial output is '1'. Linting checks ensure these kind of errors is avoided in the design.

If spyglass is applied on the code before the synthesis, it points out the above mistake. This is just one example. There are many standards and lint check rules are available. Here are some example for rules, coding style, DFT, design style, language constructs, synthesis and also possible to include custom rule checking. It is possible to select/deselect the rules/standards during lint checking. It completely depends on the need of the particular environment and design.

2.6 DFT Checks[3]

The primary DFT approach commonly used is to implement full scan on the design. Key objectives of this design method are:

- to allow any internal state necessary for testing to be achieved by forming shift registers called scan chains,

- to force latch enables to be active so that the latch may be treated as a buffer,
- to allow easy control of clocks so test results at internal nodes can be captured,

a. A typical scan flip-flop is shown in Figure 2.11. Scan flip-flops are connected (see Figure 2.12) so that shift registers are formed when the SE pin is set to 1. A critical aspect of this shifting action is that shift clocks must reach the scan flip-flops and the sets and resets must remain inactive regardless of circuit state.

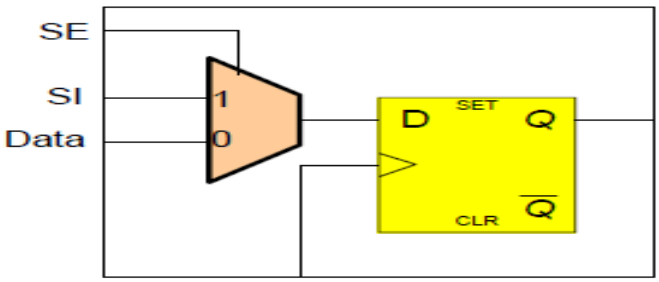

Figure 2.11: Scan Flip-flop

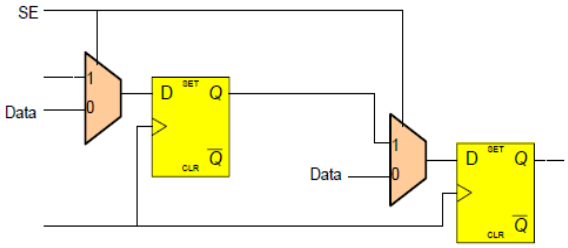

Figure 2.12: Scan Chain Segment

b. Latches are state elements that can defeat combinational ATPG tools.Tests for circuits such as shown in Figure 2.13 may require signal propagation through latches. The appropriate DFT method is to design the latch enables so that the latch is forced active in capture mode.

c. To perform the tests, the scan multiplexers must be switched back to normal or system mode so that test results can be captured for scan-out. The capture clock

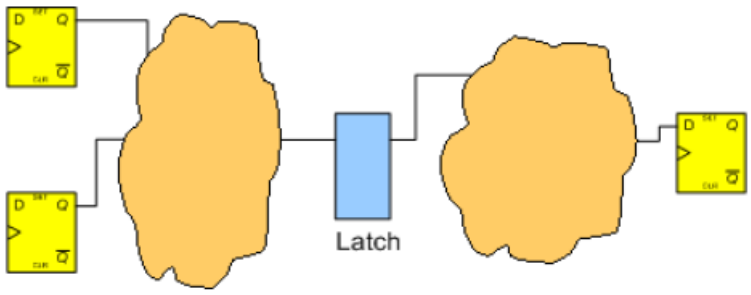

Figure 2.13: Latch Transperancy

must be operated in system mode and therefore the circuit must be designed to guarantee that the capture pulses reach the scan flip-flops regardless of the scan-in state.

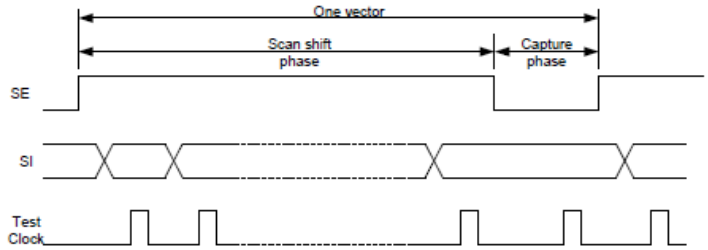

Figure 2.14: Capture Mode Diagram

So for this kind of fuctions DFT checks is applied on the RTL design. This checks ensures that every scan chain element will be checked in test mode in the Shift mode.

2.7 Typical Issues For the RTL Checks

There are many implementation issues are there which has come across my way during the RTL checks. These issues has to be solved at any level of design to get proper functionality of the IC. If such issues is solved at RTL level then it has greatly reduced the re-spin time at the later stage of the time. So solution of such cases is has much importance in VLSI Design. Here below some of these issues are mentined which has to be solved at any cost.

2.7.1 RTL Checks Issues

a. Reset De-assertion Synchronization A fully asynchronous reset is one that both asserts and de-asserts a flip-flop synchronously. Here, asynchronous reset refers to the situation where the reset net is tied to the asynchronous reset pin of the flip-flop. The biggest problem with the asynchronous reset circuit described above is that, it will work most of the time. However, if the edge of the reset deassertion is too close to the clock edge and violate the reset recovery time, then the output of FF goes to metastable. So such kind of issue cause metastability as explained in section 2.2.2. **Solution**: In Figure 2.15, the registers in the

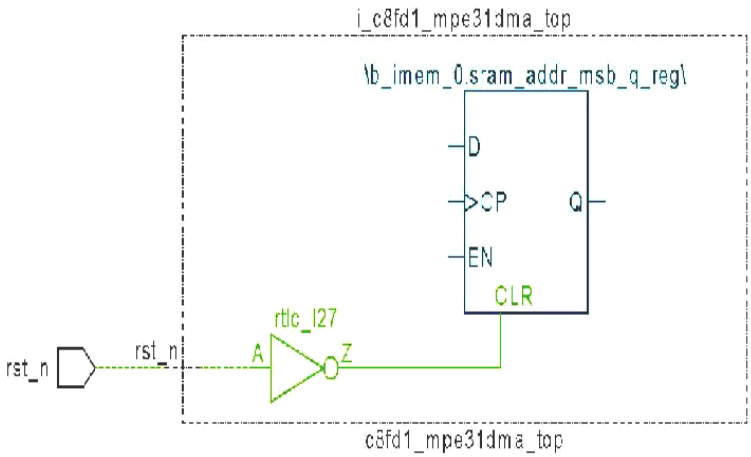

Figure 2.15: Reset Deassertion Synchronization Error

reset circuit are asynchronously reset via the external signal, and all functional registers are reset at the same time. This occurs asynchronous with the clock, which does not need to be running at the time of the reset. When the external reset deasserts, the clock local to that domain must toggle twice before the functional registers are taken out of reset. Note that the functional registers are taken out of reset only when the clock begins to toggle and is done so synchronously.

b. Unsynchronized clock domain crossings This type of Issue can cause metastability of the signals due to violation in reset recovery time. If this metastability

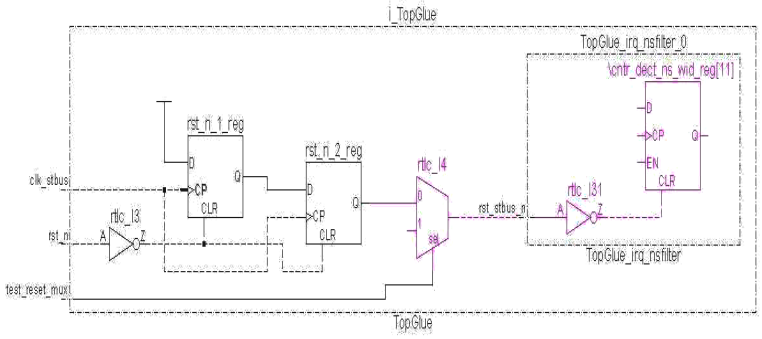

Figure 2.16: Reset Deassertion Synchronization Solution

propogate through output it can cause functionality failure of the design.

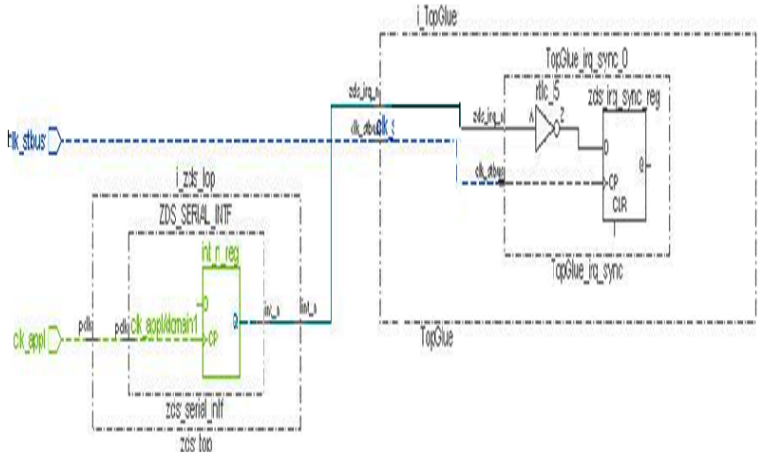

Figure 2.17: Unsynchorinized Clock Domain Crossings Problem

Solution: There are different technique to solve synchronization issues.

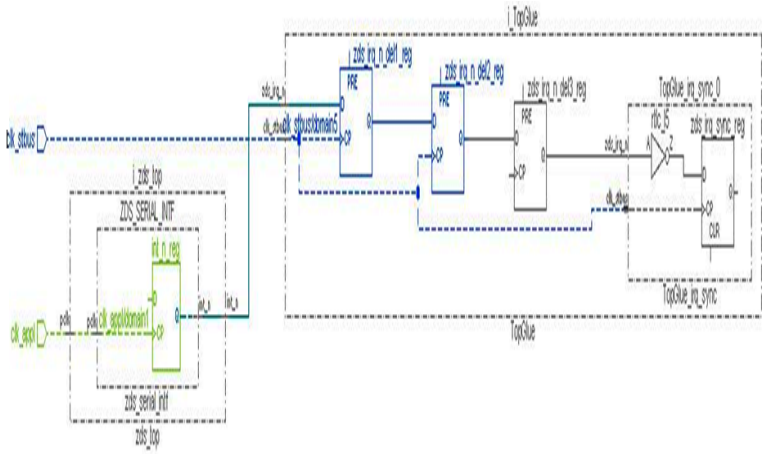

Figure 2.18: Multiflop Synchrinization Scheme

(1) Multiflop Synchrinization Scheme

(2) Enable based Enable scheme

(3) Mux based Reciculation Scheme

c. Convergence of Signals from Different Domains:

In case of convergence of signals other than singular flop outputs, this rule flags if one of the following conditions holds true:

- If the signals are coming from the same source domain but different destination domains
- If the signals are coming from different source domains but same destination domain
- If the signals are coming from the different source and destination domains

All such convergences should be avoided irrespective of the fact that the signals are gray-encoded or not.It is not a good design practice to converge signals from different source domains although in some cases it may be required by design and can be designed so there is no glitch or coherency issue. In all cases convergences of two different domain signals, reported by this rule need to be reviewed.

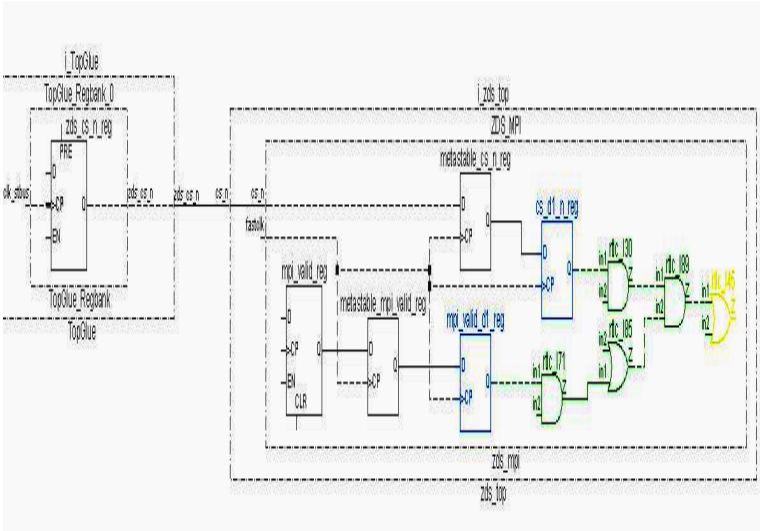

Figure 2.19: Convergence of Signals from Different Domains Problem

Solution: Convergence between signals coming from different domains is a serious issue. The design needs to be analyzed carefully for such cases as to why they are required and should be avoided if possible.

d. Converge of Signals after any no. of Sequential Elements This issue can cause data coherency issue. The issue is as shown in figure, 2.20 where the two different data are converging at AND gate after any number of sequential element.

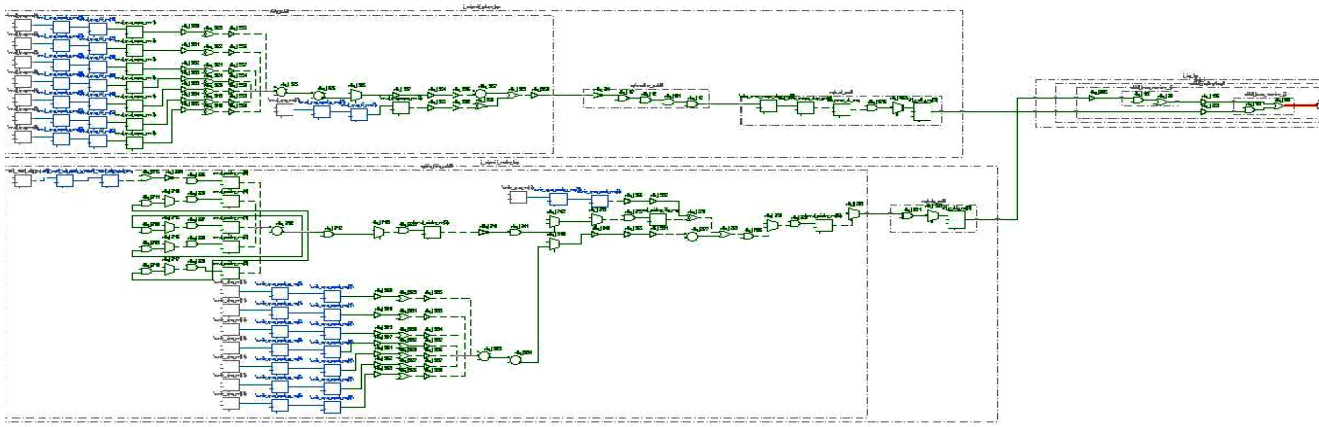

Figure 2.20: Convergence of signals after any number of sequential element

Solution: Convergence between separately synchronized signals can lead to data coherency issues. It is recommended to use a common synchronizer for all signals that are converging or verify that two or more converging signals do not control the net on which it is converging at the same time.

2.8 DFT Checks

a. All clock must be test clock controlled in test mode.

b. All FF should be scanable in test mode.

c. Asynchronous set/reset should be inactive during shift mode.

Problem: In such cases where the derived clock is used for the flops as a clock source, in scan shift mode this combinational logic wil not come in picture so in test mode these flops will not have clock in test mode, so it remains untested & these will cause failure in testing.

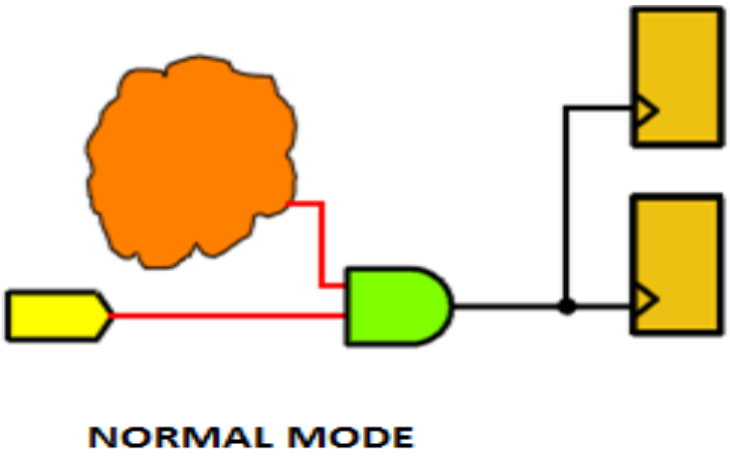

Figure 2.21: DFT Checks in Normal Mode gives Error

Solution: The solution for such kind of errors is to introduced bypass logic for the clocks or resets as shown in fig. 2.22. Here for mux in test mode signal is set to 1 directly test clock will go to the clock pin of the sequential element & when 0 generated clock will go to the same pin.

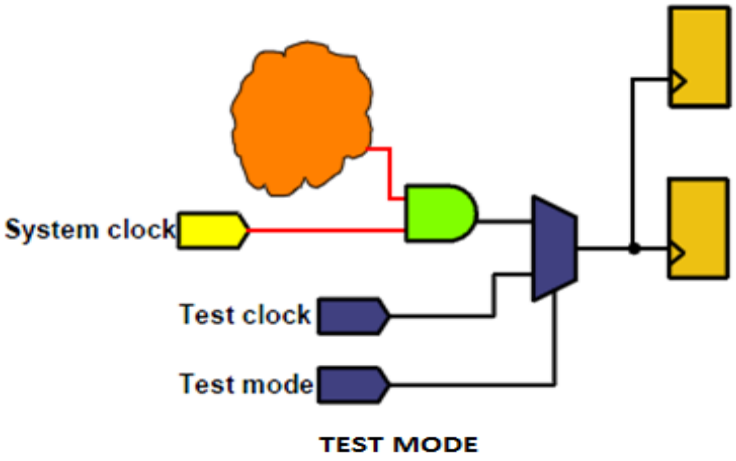

Figure 2.22: DFT in test mode

2.9 Linting Checks

a. Avoid "EXIT" statement.

b. Avoid non-integer indexes in GENERATE statement.

c. Input port not used.

d. Input value should not be greater than size specified in IEEE conversion function.

This kind of error has to be solved before the synthesis as this errors causes failure in synthesis.

2.10 Summary

In this chapter first some of the problems associated with the RTL are discussed. Also solution of these problems also described. At last some problems which are often faced by the designer is presented. In these, CDC problems are very important issues & it has to solved at the RTL level.

Chapter 3

GENSDC

3.1 Introduction

While the functionality of the design is represented by the RTL code, timing constraints play the role of influencing the performance parameters of the design. These performance parameters include power, area and timing. These parameters tend to be inter-related, but the majority of design constraints tend to be related to timing. In the context of timing, constraints convey the following important information about the performance of the design:

- The target operating frequency for various clocks
- The time at which various inputs are available
- The time at which various outputs are desired
- The drive capacity of external drivers
- The external load that needs to be driven
- Certain topological paths that need not be timed (false paths)
- Certain topological paths that are allowed more than one clock cycle to complete (multi-cycle paths)

These constraints are then used by:

- **Synthesis tools:** To direct that it should infer logic and choose gates so as to meet those performance requirements
- **Static timing analysis (STA) tools:** To direct that it should validate that the realized circuit actually meets the stated performance requirements
- **Place and route tools:** To complete placement and routing such that the performance requirements continue to be met.

For that spyglass introduced the tool name **GENSDC** which will generate the constraints with the some input from the designer. As constraints generation is automated, these constraints should have much accuracy compare to Manually generated constraints.

3.2 Why GENSDC?

- In a typical design environment, SDC files are used and modified at various stages of the design (e.g. RTL, pre-layout, post-layout). Generally these SDC files are created either manually by the Designer or with help of some automated ad-hoc scripts.
- It's very difficult to avoid mistakes (such as human error) in SDC written manually. Even the designer has to understand the design to write the SDC file.
- Apart from it, the designer has to have the technical competency for the constraints & various design flow stages. It is a time consuming process too. Even script generated SDC flow also needs continues updating of the scripts for new features.
- The SDC generation tool should have capability to understand the design and generate SDC.

3.3 Various Synthesis Constraints

For the RTL designer there are number of constraints that designer has to take care of. Some of these constraints are mentioned below.

3.3.1 create_clock

This creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

3.3.2 create_generated_clock

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source. The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place and route.

3.3.3 clock_latency

Clock latency of the clock is the delay between the primary clock source to clock pin of the sequential element.There are two types of latency,

- Source Latency: The time for the clock to reach at the source port to input is called source latency as shown in figure 3.1.

- Network Latency: Network latency is the time of clock to reach from input pin to clock pin is called network latency as in figure 3.2.

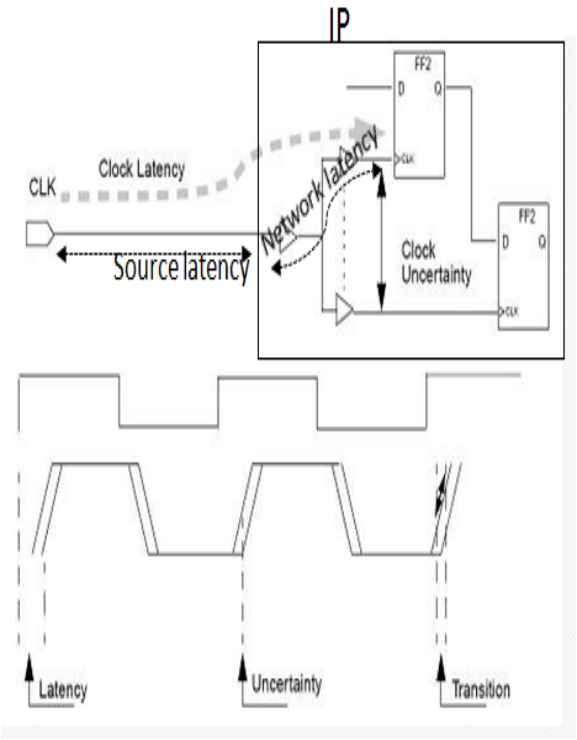

Figure 3.1: Latency & uncertainty[9]

3.3.4 clock_uncertainty

In ideal mode the clock signal can arrive at all clock pins simultaneously, but in real world that perfection is not achievable. So, to anticipate the fact that the clock will arrive at different times at different clock pins, the "ideal mode" clock assumes a clock uncertainty. For example, a 1 ns clock with a 100 ps clock uncertainty means that the next clock tick will arrive in ns plus or minus 50 ps. The main reasons for this is,

a. The insertion delay to the launching ip-op's clock pin is different than the insertion delay to the capturing ip-op's clock pin (one paths through the clock tree can be longer than another path). This is called clock skew.

b. The clock period is not constant. Some clock cycles are longer or shorter than others in a random fashion. This is called clock jitter. Even if the launching

clock path and the capturing clock path are absolutely identical, their path delays can still be different because of on-chip variation. This is where the chip's delay properties vary across the die due to process variations or temperature variations or other reasons. This essentially increases the clock skew.

So this clock skew is called clock uncertainty. It is important constraints to calculate timing violation.

3.3.5 set_input_delay

Specifies the required data arrival times at the specified input ports relative to the clock. The Clock name must refer to an actual clock name in the design. You can specify input delays relative to the rising edge (default) or falling edge (-clock fall) of the clock. If you specify only the minimum or maximum delay for a given port, the same value is used for both. You can specify separate rising (-rise) and falling (-fall) arrival times at the port. If you specify only the rise or only the fall value for a given port, the specified value is used for both rise and fall.

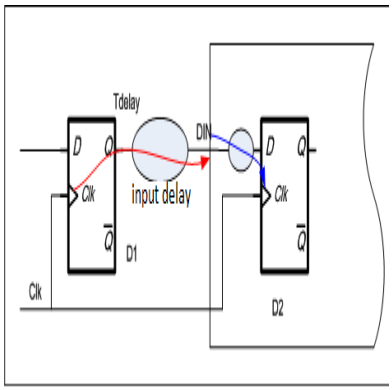

Figure 3.2: Input Delay

3.3.6 set_output_delay

The set_output_delay command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out

(bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

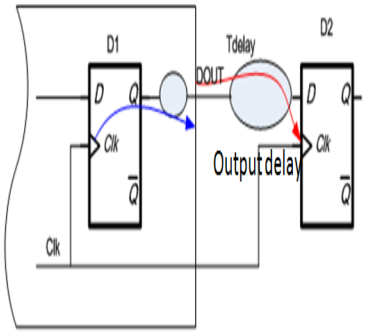

Figure 3.3: output delay

3.3.7 Max Delay & Min Delay

The Maximum Delay of the RTL is the time needed to complete the travelling of signal from input to the output. i.e to the time required to reach from input to output that is possible. This constraint is useful to timing analysis tool to find the Violation condition.

The Minimum Delay of the RTL is the time needed to complete the travelling of signal from input to the output. i.e to the time required to reach from input to output that is possible. This constraint is useful to timing analysis tool to find the Violation condition.

3.3.8 false_path

The false need to be declare for the paths on which designer don't want to check timing violations. Due to this constraints designer saves the effort & time to analyze the wrong violations. Some of the exapmples of such paths are,

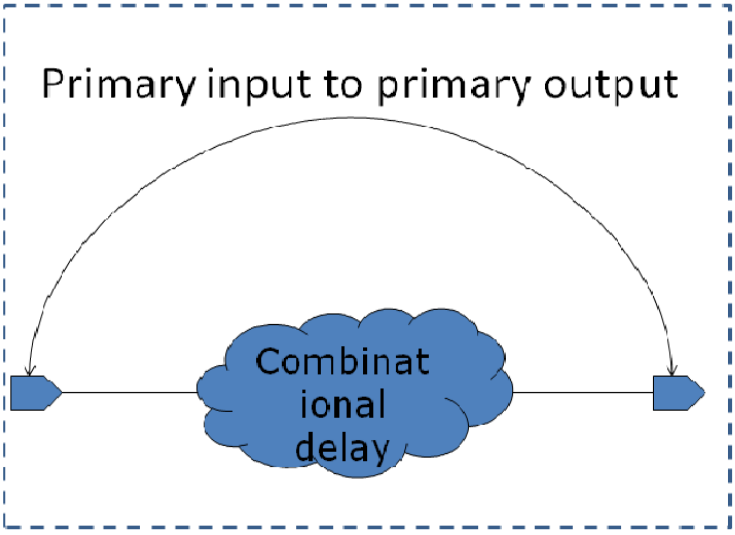

Figure 3.4: Max Delay & Min Delay:

a. False paths are paths in a design which are functionally never be true.

b. A path which has no functional purpose or a path which does not need to be timing constrained.

c. Paths that are physically exists in the design, but they are not logical/functional paths. These paths are never get sensitized under any input conditions.

As we can see from the figure 3.5, it is logically impossible from a1, through f1 and b2, to f2. It also logically impossible from b2, through f1 and a2, to f2. As by setting false path in such a case will minimize the efforts of the synthesis tool & designer need not to observe timing violation on the such paths. So designer declare false path constraints on the path **a1-f1-b2-f2**. Also desinger set this constraints on the synchronizer path as shown in figue 3.6.

3.3.9 MultiCycle Path

Multicycle paths are paths which requires more than one clock cycle to propagate the data due to slow combinationl logic between clocks as shown in figure 3.7. Here

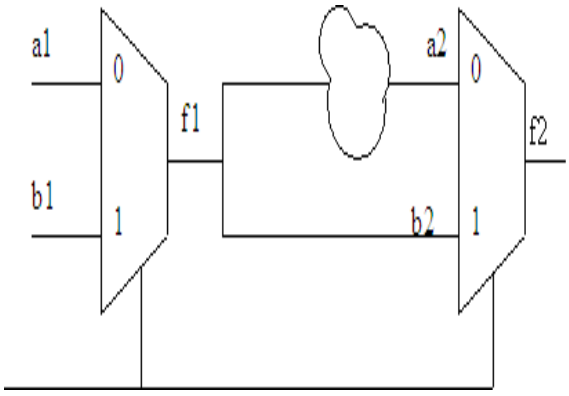

Figure 3.5: False Path

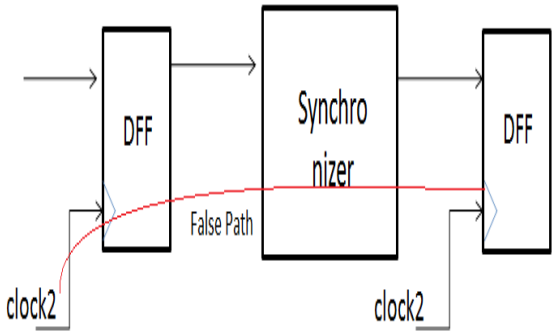

Figure 3.6: False Path on Synchronizer

due to slow combinational logic the multicycle path needs to be declared at the for the path from clock 1 to clock 2.

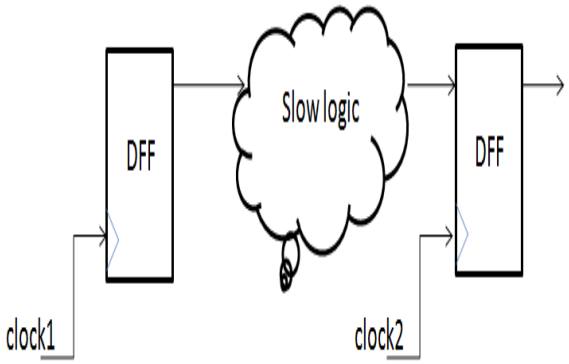

Figure 3.7: Multicycle Path

3.4 Issues Related with GENSDC

While exploring the GENSDC there are some issues that is found out & which is need to be resolved. This issues are,

3.4.1 GENSDC wrongly detect the clock

a. When the mux output pin is going to the clock pin of any flop then it generates the clock at the data pin of the mux even if it is not going to the CP pin of the mux. Normally in this type of design structure the output of the mux is generated clock with the source pin is select pin of the mux.

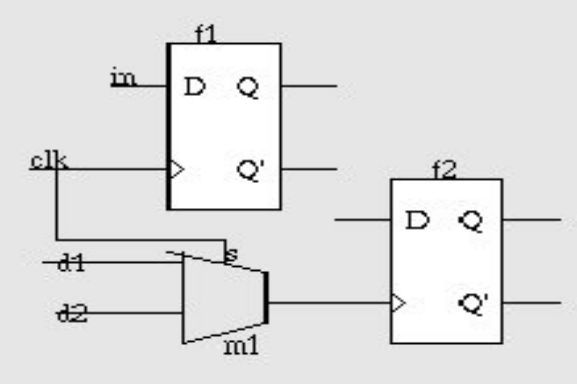

Figure 3.8: ISSUE 1

b. Wrong clocks created at the select of the Mux and the input of the mux whose output is going to select pin of another Mux selecting between 2 clocks and the output of this mux is driving the memory cell as a clock. Such structures is normally used to select clock for the memory.

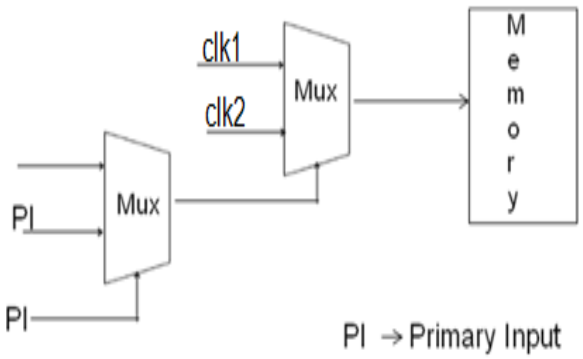

Figure 3.9: ISSUE 2

c. When tool sees a gate (XOR gate in our case) & one of its input is going to the CP pin of any flop then GENSDC generates the clock at the other input of the gate. Which is definitely a extra clock.

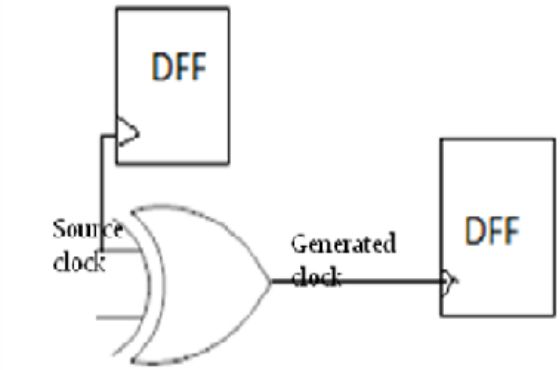

Figure 3.10: ISSUE 3

3.4.2 Other issues

There are some other issues that also be taken care of to increse the tool usability. These issues are,

a. GENSDC is not able to generate set_load constraint for the input ports. Which is need to be set to tell the synthesis about the loading strength of the design.

b. Tool neither can generate set_multicycle_path constraint nor it has some input request for multi cycle path. This is very important constraint for the designer. Designer need to input all multicycle path constraints manually.

c. set_ideal_network constraints is generated on the internal pins of the high fanout objects. This internal nets may get optimized down the flow. So this will useless for the synthesis tool.

3.5 Advantage of GENSDC

- GENSDC can find all the clock domain crossing path related to the different clocks & it also gives the flexibility to the designer to set either false path or clock uncertainty for different clocks.

- GENSDC can find all the input to output paths on its own to generate the constraints set_max_delay & set_min_delay. Only designer need ensures the values for this constraints.

- GENSDC can take different parameter for constraints inputs from the library file of their respective technology (e.g. 40 nm,65nm). So designer need not to care about the value different constraints.

3.6 Results

For the constraints generated on the different IPs, here is the result comparision of the SDC checks errors with respect to constriants generated manually.

Table 3.1: Results for SDC checks:

IP	No. Of Errors for manually generated constraints	No. Of Errors for GENSDC generated constraints
Uniperif_reader_top	32	10
Uniperif_player_top	58	3
TELSS	83	18
HDTVOut	865	133

3.7 Summary

Automation of constraints generation will reduced the generation time. This will also help to improve the quality of the constraints. As shown in table 3.1, the no. of errors in the SDC checks of automated generated constraints is very less compare to manually generated constraints. But as this tool is not able to understand some concept of clocks as described in section 3.4, fully automated constraints are not reliable. So here human interference is necessary to get accurate constraints.

Chapter 4

SDC Checks

As today's designs become more complex, so too do their constraints. Design functionality typically gets a lot of attention through code review, functional verification, etc. However, the constraints themselves also need to get the same level of attention. Because of good quality of constraints designer gets the accurate & the critical implementation and timing analysis of the RTL during synthesis which gives the accurate approximation & timing analysis of that design. So designer can ensure that the design is going to be pass or fail on the Silicon. The complexity of constraints is further increased by the following design characteristics:

- Low power applications (e.g., handheld devices) require extensive clock gating

- Pad limited designs require a high degree of input multiplexing

This chapter highlights and discusses the importance of constraint validation early in the design flow, and analyzes the impact of this validation approach on a real design. After that we will see some issues associated with the SDC checker which has come across in the design during the flow.

4.1 What is SDC Checks?

Validating constraints through out the design flow requires a methodology that will guide designers through each step in the flow, specifying how to clean up and optimize the design constraints. This will not only improve the QoR(Quality of Result) but will reduce expensive re-spins and iterations. This constraints are very useful for the synthesis tool to do the synthesis of the design from the RTL. So checking the constraints at the RTL level will give exact idea about different "timing constraint" which will be useful to predict timing of the design & also to generate the netlist.

4.2 Constraint Problem[5]

Typical Problems within a Constraint File

- Clock Definitions: Clock issues lead to excessive iterations among block synthesis, STA, and P&R. This includes the inconsistencies in specification of clocks, generated clocks, and all related clock data like latency, uncertainty, buffering, and so on.

- Input and Output delays: Inconsistencies in input and output delay specification can lead to incorrect or non-optimum synthesis results. Over-constraining may result in longer synthesis run times and extra buffering on tight paths. Under-constraining will result in not meeting chip level timing goals.

- Exceptions: Exception validation is needed because of Incorrect timing exceptions, especially the ones on timing critical paths, may lead to silicon failing to meet timing as the timing path violations are masked until silicon. Too many exceptions overwhelm the implementation tools. Thus, if there are exceptions on invalid paths, or paths blocked by constant propagation, or functionally incorrect exceptions, it is better to identify them in advance and remove them from the constraints before implementation. Verifying exceptions manually is a

time-consuming and error-prone process. Typical exception issues include false paths set on paths that are not structurally connected, or false paths specified on true paths, or incorrect cycle counts specified for multi-cycle paths.

- The level of support on various commands/options for constraints varies from one tool to another. Thus, maintaining a flow involving multiple vendors requires a tool which can intelligently indicate if the constraint is friendly to a specific tool.

- Constraints issues are not limited to a single constraint file for a block, but can also occur in the hierarchical context involving multiple constraint files for several blocks and the chip level design.

4.3 The Need for Constraint Validation Tools

In the design constraints it may be possible that designer has forgot to introduce some important constraints or it is also possible that unknowingly introduced wrong values in the constraints. These kind of problems can't caught by simulation tools. Though there are an issues in the constraints simulation tool will pass the functionality of that design. So after that the earliest that a designer can catch wrong/missing constraints is during or after synthesis by observing the timing report of the synthesis. Here also, there are several possibilities depending on the nature of the constraint.

- Certain constraints have a default value which is highly optimistic, e.g., set_input_delay. If a user has forgotten to specify set_input_delay, a default value of "0" is assumed, and no issue will be reported. Even a typical STA tool will not be able to catch such an issue. Both synthesis and STA tools will happily assume the "0" value and provide optimistic results. Unfortunately, the silicon might not behave that optimistically, and the device might not meet its performance target. Several other constraints (e.g., set_input_transition) have a highly optimistic default value.

- The majority of constraints are simply assumed to be a "statement of fact" from the user. Implementation/timing tools simply honor those, without any verification or warning. A particular set of constraints which relax a requirement fall under such category, e.g., "set_false_path". If a path is declared as "set_false_path" both synthesis and STA tools will not time the path. Once again, this could cause the silicon to fail.

- Some constraints, when missing, would be reported by a synthesis tool, e.g., "create_clock" or "create_generated_clock".

Even if synthesis tools could be relied upon to catch some of the issues with constraints, they still would not catch issues related to mismatches or inconsistencies among the constraints of various blocks. This is because synthesis tools look at each block in isolation without looking at its surrounding blocks. The earliest that such mismatches would be caught is during STA at the full chip level; though even at this point they are often caught indirectly and manifest as timing problems rather than constraint problems. Unfortunately, by this point the blocks have already been synthesized and we are already too far along in the flow to catch such issues. This will often result in repeated iterations through synthesis onwards.

There is a real need to verify timing constraints much earlier in the design flow, so that we are not caught by surprise later during implementation/STA. For example,

- During silicon testing: The silicon may not meet timing, because of optimistic "defaults" in various tools

- During STA: Even though the individual blocks were meeting the timing, the full chip does not meet timing because of constraint inconsistencies between different blocks

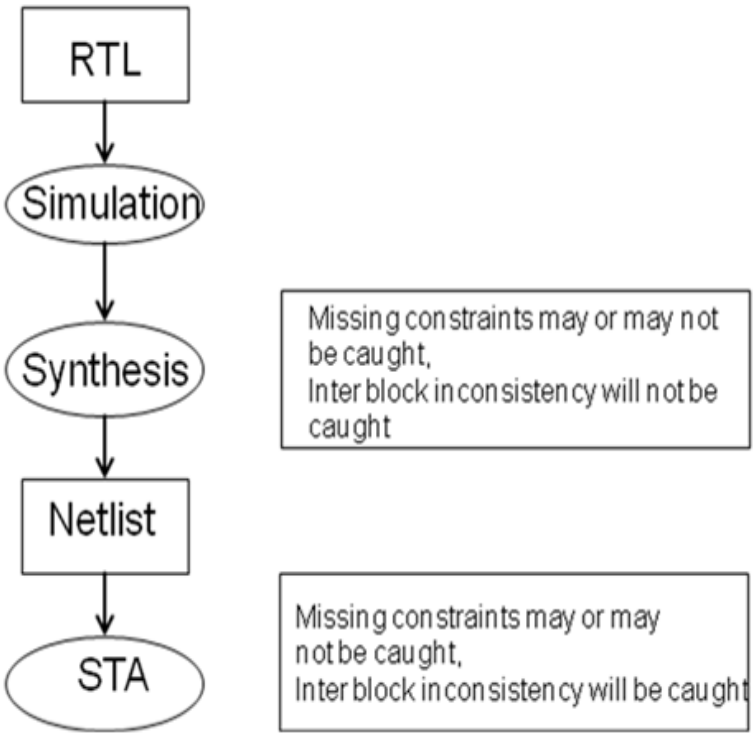

Figure 4.1: Catches/Missing the Erronoues Constraints

4.4 Complexity of Constraints[5]

Conceptually, constraints are very simple. However, there are a number of ways in which high impact mistakes can be made. Here is the some common issues which are designer can made & it has to be avoided.

a. A clock is not declared at a signal/port which is being used as a clock.

b. A clock is declared at a signal/port which is not being used as a clock.

c. A generated clock is being generated,the source specification is wrong.

d. A generated clock is being declared,with the wrong value for divide_by.

e. A simple clock divider, implemented using a flop's output inverted and fed back to its own "data" input. Depending on whether the generated clock is being tapped before or after the inversion-the clock waveform would be different. A

generated_clock is declared with a waveform that is not commensurate with the point of tapping.

These kind of errors in the constraints can cause many serious issues & may cause timing violation in the in the synthesis. So designer needs to correct the constraints & rerun the synthesis. So this rerun may take long time than estimated & increases the TTM(Time to Market.)

Since constraint development/maintenance is traditionally thought of as a "back-end" issue, many RTL designers are not that conversant with these complexities. There is a good chance that they will make some of the mistakes outlined above which can result in silicon failure, or, in the best case synthesis respins. Early constraint validation allows one to avoid issues such as,

- Missing constraints where tools may proceed with "optimistic defaults"
- Constraints which are inconsistent among different partitions which will result in additional synthesis iterations after STA
- Other less commonly used constraint scenarios, it's likely that these may be overlooked during manual review. Many new designers might not be aware of these fine details. The backend teams (who usually have reasonable knowledge of constraints) may not be familiar enough with the design to catch these issues

4.5 SDC Checks Issues

a. Clock driven by a constant value or hanging: Clocks being driven by a constant value is a serious error in the design because the timing analysis results will be invalid without proper clock constraints. The timing results may give a false sense of security and you may get silicon timing failure. This can cause sections of the chip to be non-functional Solution: Here designer has to taken care that

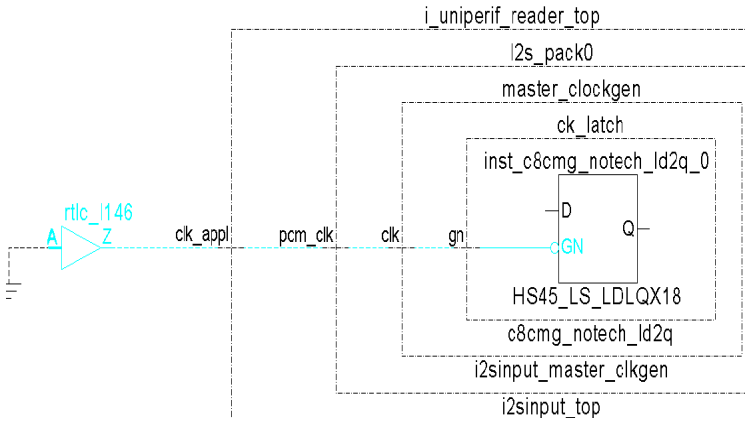

Figure 4.2: clock Driven Problem Problem

such cases will not occur in the design. Designer has to keep in mind that these kind of erros is not occur in the unintentionally design.

b. Combinational port-to-port path is unconstrained Problem: If a combinational path is unconstrained then no timing check would be made. Therefore, the operation of the device at any specified speed cannot be guaranteed. Solution:

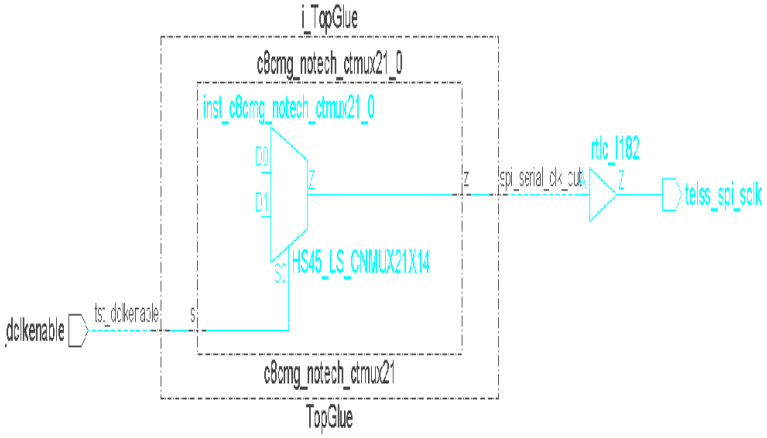

Figure 4.3: Port to Port Unconstrained Path

The schematic shows the unconstrained combinational path starting from an input port to an output port. The path is unconstrained when one of the pair of constraints is missing.

4.6 Issues with SDC Checker Tool

Due to some of the misconcept or misunderstanding of the SDC checker rules ,there are some errors which designer will get during the this run. These kind of errors will consume time of the designer to observe it. So if these kind of errors are avoided then time will be saved. Some the rules are as described below,

a. Rule-1: Incorrectly Defined Generated clock: One of the rule in the for the SDC checker is, "If there is a combinational path from the source clock to the generated clock pin, the divide by factor should be 1" Above figure shows that, if this kind of structure is there in the design divided by factor has to be 1. But

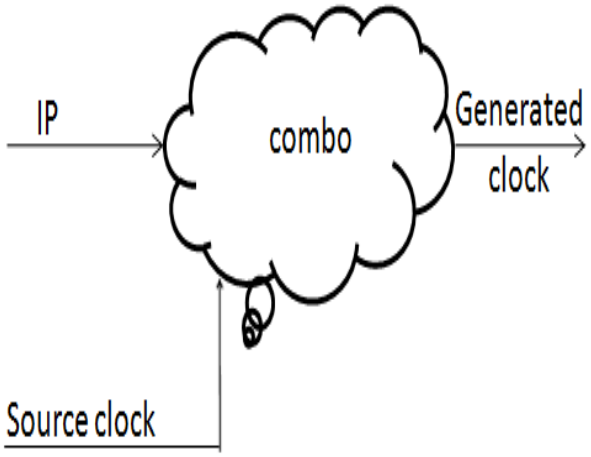

Figure 4.4: Rule 1:

for the mux base clocks divider this can not be the case as shown in figure. For this case the divide by factor can be any depending upon the mux inputs. So this rule has to be modified for generated clock generated by combo logic.

b. Equal value for setup & hold time uncertainty: SDC checker checks for equal uncertainty / skew values in case of set-up and hold checks. This seems to be conceptually true but as per the front end design flow at the IP level, hold checks are not done and so we set the uncertainty for hold checks to zero so that there should be minimum no. of violation in the timing report of the synthesis.

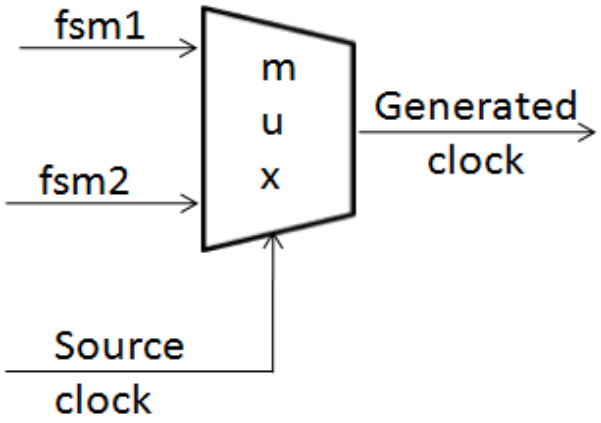

Figure 4.5: mux based divider:

This means here we have a case of unequal set-up and hold uncertainty. So for front end design flow this rule has to be modified.

c. Errors related to Notech cells optimization: In the constraints file if some of the constraitns is set on the internal nets of the IP then SDC checker flags the error in that these nets will optimized down the flow. But for some cases of special cells like Notech cells, these kind of cells will not optimized down the flow esspecially in the synthesis. So these errors should not there for notech cells.

4.7 Summary

In this chapter we have seen the importance of SDC checker tool for the for the designer to do synthesis. This tool will find almost every issues at the related to constraitnts. So the sythesized design will be fully optimized & designer will find some real issues related to the design at the netlist level. Also to there are some errors in the tool which are to be moified to improve the understanding of the tool.

Chapter 5

POWER ESTIMATION

5.1 Introduction

As IC design sizes and frequencies increase, power consumption goes up. Some years ago, only a few design teams were concerned about power. Now all RTL designers consider power to be critical, and many designers of wall-powered applications are also concerned about power. In order to address these concerns, there is a need to estimate the power consumption of a design very early in the design cycle. If the estimated power is higher than the power target, then designers should be able to reduce the power consumption in a time efficient way. Typical power reduction techniques include the use of voltage domains, power domains, and clock gating. Since voltage domains and power domains require special logic circuits and make the power grid design more complicated, there is a need to verify that the voltage and power domain implementation is correct.

5.2 Components of Power Dissipation

For CMOS designs, the total power consumed by a circuit falls into 2 major categories: static power, which is the power consumed when the transistors are not switching, and dynamic power, which represents the power consumed by switching

logic states.

Static Power

Static power is dissipated in several ways. Some are due to the reverse-biased diode leakage from the diffusion layers and the substrate, but the largest percentage of static power results from source-to-drain sub-threshold leakage current. This is caused by reduced threshold voltages which prevent the gate from completely turning off and hence allow this leakage current (I_{leak}).
The leakage power is dependent on the voltage, temperature and state of the transistors.
Leakage Power = V * I_{leak}

Dynamic Power

a. Switching Power: The switching power is determined by the capacitive load and the frequency of the logic transitions on a cell output.

$$SwitchingPower = \frac{1}{2} * C_{load} * V^2 * f \tag{5.1}$$

where the total load capacitance (C_{load}) is the sum of the net and gate capacitances on the driving output, and the frequency (f) is the rate of state transitions.

b. Internal Power: The internal power is caused by the charging of internal loads as well as by the short-circuit current between the N and P transistors of a gate when both are on.

$$InternalPower = \frac{1}{2} * C_{int} * V^2 * f + V * I_{sc} \tag{5.2}$$

As the input signals transition, both N and P type transistors can be on simultaneously. During this time, current I_{sc} flows from V_{dd} to Gnd causing

the dissipation of short circuit power. The short circuit power is affected by the dimensions of the transistors, the load capacitance on the output, and the transition time of the input signals. Circuits with slow transition times can dissipate excessive short circuit power as both N and P transistors are on for an extended period. ASIC or Library vendors provide power models for the internal power consumption of CMOS cells, which are characterized with different driver output loads and input signal transition times. The diagram below describes the power components for a simple buffer cell.

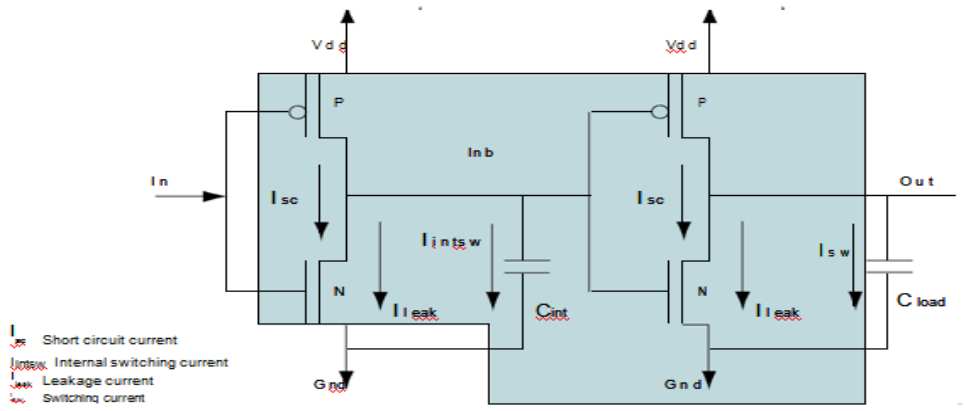

Figure 5.1: Components of Power Dissipation[2]

The leakage current I_{leak} can vary based on the transistor states. For example, when the input signal In is high, and the N transistor is on, the leakage will differ compared to when the N transistor is off. When a rising signal is applied at the input, internal power is dissipated due to I_{sc} and I_{intsw}. During the transition from low to high, the N transistor turns on and the P type transistor turns off resulting in I_{sc} from V_{dd} to Gnd. Additionally, internal switching power is incurred in charging and discharging of Cint. The switching power on the Out net is due to I_{sw} charging and discharging C_{load}.

5.3 Power Analysis Requirements0[2]

In order to analyze the power consumption of a design at RTL, it is important to consider all the factors which contribute to both the static and dynamic power. These are shown in the diagram below: In order to analyze the power consumption, the following are required:

a. RTL Data : The RTL is required to determine the design connectivity and type of cells used in the design and to accurately compute the capacitance on the drivers.

b. Cell Library Power Models : In order to compute the internal power of the CMOS cells used in the design, ASIC or Library vendors must provide cell models which specify both the static and dynamic power consumption internal to the cell.

c. Signal Activity : The signal activity of a design affects both the static and dynamic power consumption. The static power (cell leakage) is often state dependent, and the dynamic power is directly proportional to the toggle rate of the pins. So for the signal activity VCD or SAIF file generated from the simulation is as input for the RTL level.

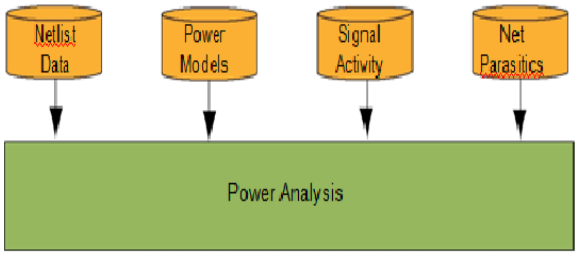

Figure 5.2: Power Analysis Requirements

d. Net Parasitics / Transition Times : Net parasitics (or capacitances) affect the dynamic power of the design. Switching power is directly proportional to the net capacitance. Internal power depends on both the input signal transition

times, which are determined by the net parasitics, as well as the output load, which is a combination of the net parasitics and input pin capacitances of the fanout. The example below illustrates the relationship between the transition times and the dynamic power consumption.

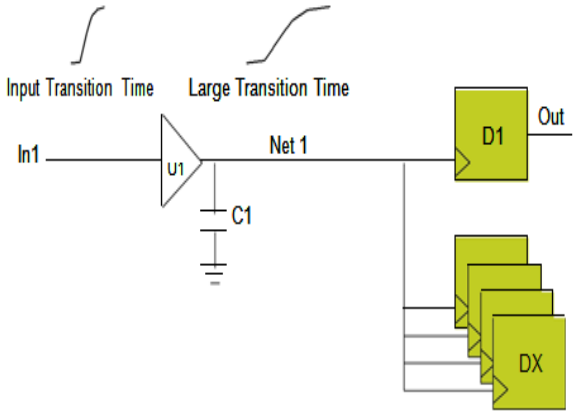

Figure 5.3: Transition Times Affect Dynamic Power Consumption

The internal power consumption of U1 is determined by the transition time on In1 and the capacitive load on the output Net1, which is the sum of the input pin capacitances of the gates it drives (gate fanout) and the net capacitance, C1. Now if the net capacitance C1 is large, the switching power of Net1 certainly will increase, but additionally, the transition time on Net1 will be affected. A large transition time on a net with a large number of fanout loads results in excessive internal power consumption for all of the gates to which it is connected. This is because the internal power consumption of the fanout gates, the registers D1-DX in this example, is considerable because the long transition time on the net is allowing for a longer period of time in which short circuit power is being dissipated. In summary, the accuracy of the power analysis is dependent upon the accuracy of the inputs provided. The netlist, cell library power model, signal activity, and parasitics/transition times need to be accurate to provide accurate power consumption analysis.

5.4 Signal Activity Flow for Power Analysis[7]

At RTL level no netlist data is available. So for accurate power estimation at the RTL level there is a simulation file should be provide to the tool. This file are VCD(Value Changed Dumped File) & SAIF(Switching Activity Interchanged Format). This file can be generated from the simulation.

- VCD File:

  a. VCD is a ASCII-based format for dumpfiles generated by EDA logic simulation tool

  b. A VCD is an ascii file (that's both an advantage and disadvantage) describing a digital waveform from a VHDL/Verilog simulator.

  c. vcd file - contains value changes of a signal. i.e. at what times signals changes their values. saif does not contains this information. it contains cumulative information of vcd.

  d. it contains the waveforms for all the signals in the RTLs

  e. Any application which needs time stamps of individual value changes must have to use vcd e.g. time based power estimation or if you have a vcd file

- SAIF File:

  a. The limitation of VCD can overcome by SAIF file.

  b. SAIF is the EDA industry's most widely used power format that provides switching activity information to power optimization and analysis tools

  c. saif file - contains toggle counts and time information like how much time a signal was in 1 state(T1), 0 state(T0) , x state (TX). Also in backward saif file you can have timing information.

5.5 Power Analysis Flow for Signal Activity

Accurate power analysis depends on accurate signal activity. Power analysis supports both average and peak power analysis based on the signal activity provided. For average power analysis, supports the propagation of switching activity based on either tool defaults, user-defined switching activity, or switching activity derived from a logic simulation (either RTL or gate-level), typically saved as a SAIF (Switching Activity Interchange Format) file. For peak power analysis, requires a timing logic simulation and generation of a VCD that captures the activity and time of every event on each net. The VCD-based analysis is extremely accurate since all the factors contributing to power consumption are supported in an accurate form. Peak and average power can be calculated, and detailed, time-based waveforms can be generated. If event-based simulation activity is not available, accurate average power can still be calculated by providing gate-level toggle rates, typically with a SAIF file.

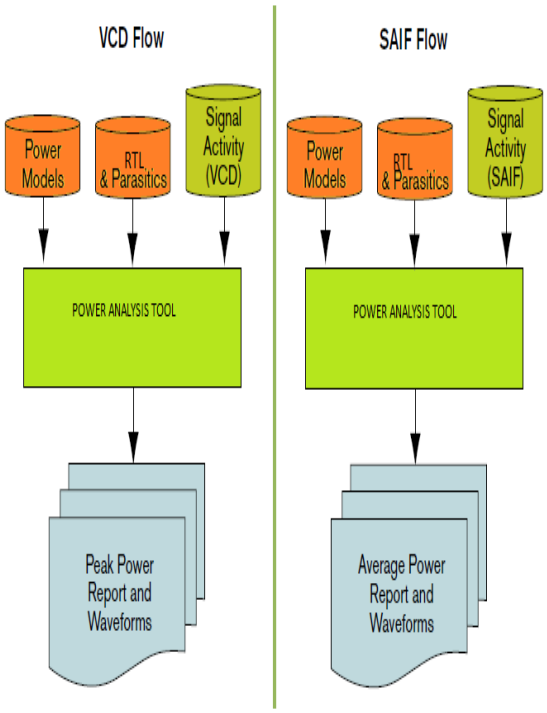

Figure 5.4: Power Analysis Flow

In the SAIF file, the activity information includes the number of times the signals toggled per net, as well as the percentage of time that the signals were at a given state. This toggle rate information enables both dynamic and leakage power to be accurately determined.

5.6 mkPower

For the RTL level power analysis Spypower tool is used to estimate the power. But results of these tool only be valid if its results is compared with the some standard tool. mkPower is such kind of tool. It calculates the power at the netlist level. The flow for these tool to calculate the power is as shown below.

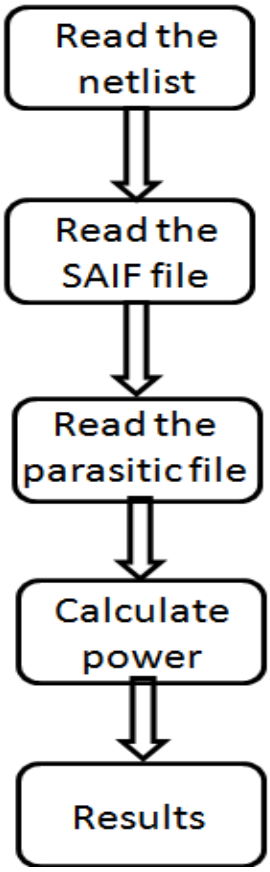

Figure 5.5: mkPower Flow

5.7 Power Results for Different IPs

Here is the result for power estimation of different IP's. Here the spypower has used RTL to generate the power reports.

Table 5.1: Spypower Results with RTL

IP1	TELSS			
Power	Leakage	Internal	Switching	Total
Combinatinal	1.71mW	853uW	2.09mW	4.66mW
Sequential	2.6mW	8.195mW	1.642mW	12.45mW
Memory	22.7uW	6.69mW	14.1uW	6.73mW
Total	4.33mW	15.7mW	3.75mW	23.8mW
IP2	Uniperif_reader_top			
	Leakage	Internal	Switching	Total
Combinatinal	244uW	0.7uW	1.78uW	246uW
Sequential	496.6uW	334.5uW	19.5uW	851.2uW
Memory	0	0	0	0
Total	741uW	335uW	21.3uW	1.1mW
IP3	Uniperif_player_top			
	Leakage	Internal	Switching	Total
Combinatinal	266uW	1.76uW	5.02uW	272uW
Sequential	850.5uW	517uW	4.22uW	1.47mW
Memory	0	0	0	0
Total	1.12mW	618uW	9.25mW	1.74mW

Here is the result of the standard tool called mkPower with netlist for different IPs.

Table 5.2: Power Results with mkPower

IP1	TELSS			
Power	Leakage	Internal	Switching	Total
Combinatinal	2.616mW	913uW	1.476mW	5mW
Sequential	3.99mW	7.087mW	616uW	11.6mW
Memory	1.4mW	6.9mW	15.22uW	8.31mW
Total	8.006mW	14.6mW	2.335mW	24.9mW
IP2	Uniperif_reader_top			
	Leakage	Internal	Switching	Total
Combinatinal	804.6uW	1.095uW	1.66uW	807.3uW
Sequential	1.05mW	283.6uW	7.66uW	1.36mW
Memory	0	0	0	0
Total	1.868mW	284.7uW	8.45uW	2.162mW
IP3	Uniperif_player_top			
	Leakage	Internal	Switching	Total
Combinatinal	637uW	56.8uW	61.3uW	755uW
Sequential	902uW	902uW	6.5uW	2.17mW
Memory	0	0	0	0
Total	1.95mW	1.03mW	72.7uW	3.05mW

Table 5.3: Power Results of Spypower at Netlist

IP1	TELSS			
Power	Leakage	Internal	Switching	Total
Combinatinal	2.66mW	906uW	1.38mW	4.95mW
Sequential	4.05mW	9mW	1.844mW	14.9mW
Memory	1.4mW	6.84mW	17.7uW	8.26mW
Total	8.11mW	16.7mW	3.25mW	28.1mW
IP2	Uniperif_reader_top			
	Leakage	Internal	Switching	Total
Combinatinal	789uW	31.6uW	49.5uW	870uW
Sequential	1.07mW	506.1uW	5.18uW	1.5mW
Memory	0	0	0	0
Total	1.86mW	537uW	54.68uW	2.87mW
IP3	Uniperif_player_top			
	Leakage	Internal	Switching	Total
Combinatinal	637uW	56.8uW	61.3uW	755uW
Sequential	1.26mW	902uW	6.5uW	2.17mW
Memory	0	0	0	0
Total	1.95mW	1.03mW	72.7uW	3.05mW

So here in the above tables we have seen the power results for the tool spypower & power for different cases on the different IP's. The result should be same for both the tools but there are mismatches in the results which is described in section 5.8.

5.8 Summary

With the transition to finer process geometries, signal integrity effects and power consumption have become a top concern. When ignored, circuits can either fail in silicon, or not meet performance specifications. Given the effects of parasitics on timing, and the effects of timing on power consumption, designers need an environment in which the interdependencies between power and timing analysis are accounted for.

Use of spyglass to calculate power at RTL will help the designer to reduce the power at the early stage of the design when power reduction can be done easily with the different power reduction techniques. But still there are result differences as shown

Table 5.4: Summary Table

Power	TELSS				
	mkPower	Spypower with RTL	% Difference	Spypower with netlist	% Difference
Leakage	8.006mW	4.33mW	45.91%	8.11mW	1.25%
Internal	14.6mW	15.7mW	7.5%	16.7mW	14.38%
Switching	2.335mW	3.75mW	60.6%	3.25mW	39.18%
Total	24.9mW	23.8mW	4.41%	28.1mW	12.85%
	Uniperif_reader_top				
	mkPower	Spypower with RTL	% Difference	Spypower with netlist	% Difference
Leakage	1.868mW	741uW	60.33%	1.86mW	0.4%
Internal	284.7uW	335uW	17.71%	537uW	88.83%
Switching	8.45uW	21.3uW	152%	83.9uW	546%
Total	2.162mW	1.1mW	49.12%	2.44mW	12.85%
	Uniperif_player_top				
	mkPower	Spypower with RTL	% Difference	Spypower with netlist	% Difference
Leakage	1.93mW	1.12mW	42.56%	1.95mW	1%
Internal	968mW	618uW	38.1%	1.03mW	6.75%
Switching	6.2uW	9.25uW	49.2%	72.7uW	1096%
Total	2.906mW	1.74mW	12.27%	3.05mW	5.17%

in table 5.4. The maximum result difference for big IP like **TELSS** is almost 46% for the leakage power at the RTL. Ideally the leakage power should be almost same for both the tools. So from these results, it can be concluded that Spypower is not a reliable tool to calculate the power at RTL level till now.

Chapter 6

Conclusion

With a growing number of clocks in today's SoC designs, increased design complexity, and pressure for first silicon success to achieve "Time to Market", all clock and timing issues have become a critical factor in the IC design.

RTL Checks will ensure the correctness of the CDC problems which will cause metastability in the design & further failure in functionality. So solving CDC problems at the early stage of the design will help to reduce the verification efforts. RTL checks also ensure the synchronized reset with the clocks.Also RTL checks does the Linting checks which will ensure the synthesizability & also make sure that there should be no mismatch between simulation & synthesis of the same design.

GENSDC will helps the designer to automate these constraints generation process which can causes minimum human errors & makes the task easy and faster for the designer. As shown in table 3.1 the no. of SDC Checks errors can greatly be reduced campare to manually generated constraints. But still due to some issues in this tool due to which tool not seems mature enough to automate the whole constraints generation process. But human intervene is required to get accurate constraints.

SDC checks will help to generate accurate netlist from the RTL, it is must to have a good & accurate design constraint. SDC checks help the designer to find the errors in the constraints if any which will help the designer to avoid design flaws and costly respins. So SDC Checks will ensure ,

a. constraints are correct and consistent throughout the design flow, from RTL through floorplanning

b. Results in better-optimized silicon

c. Prevents design errors and re-spins

Power Analysis also very important for any design. If this point is not taken into consideration then design may fail at silicon. So estimating power at the RTL level will help the designer to ensure that how much power will be consumed from the silicon. This will help the designer add some techniques such as clock gating, to reduce the consumption of power. As this is done at RTL level, it will save the design time. But as shown in table 5.4 the result is not accurate at RTL level compare to netlist level. So still tool is not reliable to do power estimation at RTL.

So now a days Early Design Analysis is prime concerned for the RTL designer now a days to reduce design respin & time-to-market time.

References

[1] Technical Paper by Cadence: Cadence Inc, Technical Paper- CLOCK DOMAIN CROSSING :**CLOSING THE LOOP ON CLOCK DOMAIN FUNCTIONAL IMPLEMENTATION PROBLEMS**, **2004**.

[2] Gordon Yip, Product Marketing Manager, Synopsys, Inc.,Technical Paper by Synopsys: **Expanding the Synopsys PrimeTime Solution with Power Analysis** ,**2006**, **June**

[3] Atrenta Inc, **Spyglass Tool's Manuals**,**2009**, **June**

[4] `http://asic-interview.blogspot.com/2010_01_01_archive.html`,accessed on 23-10-10.

[5] `http://www.design-reuse.com/articles/21289/verification-generation-constraints.html`, accessed on 11-12-2010

[6] `http://www.atrenta.com/solutions/spyglass-family/spyglass.html`, accessed on 11-12-2010

[7] `www.wikipedia.org`, accessed on 04-02-2011

[8] `http://www.asic-digital-design.eu/basics-in-digital-design/asic-digital-design-flow`, accssed On 23-03-2011

[9] `http://blog.ednchina.com/olivernie/829200/message.aspx` accesed on 01-04-2011.