

Logical Convergence of High Speed Design using Formal Verification

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(VLSI Design)

By

Raj Chokshi

(15MECV05)



Electronics & Communication Department
Institute of Technology
NIRMA University
Ahmedabad-382 481
May 2017

Logical Convergence of High Speed Design using Formal Verification

Major Project Report

*Submitted in partial fulfillment of the requirements
for the degree of*

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Raj Chokshi
(15MECV05)

Under the guidance of

External Project Guide:

Mr. Sandeep Asija
Engineering Manager
Intel Technologies
Bangalore

Internal Project Guide:

Dr.N.P.Gajjar
PG Coordinator of Embedded System
Institute of Technology,
Nirma University,Ahmedabad.



Electronics & Communication Engineering Department
Institute of Technology
NIRMA University
Ahmedabad-382 481
May 2017



Certificate

This is to certify that the Major Project entitled “**Logical Convergence of High Speed Design using Formal Verification**” submitted by **Raj Chokshi (15MECV05)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design , 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.

Dr. N.P.Gajjar

Internal Guide

Dr. N. M. Devashrayee

PG Coordinator (VLSI Design)

Dr. Dilip Kothari

Head, EC Dept.

Dr. Alka Mahajan

Director, IT-NU

Date:

Place: Ahmedabad



Certificate

This is to certify that the Project entitled "**Logical Convergence of High Speed Design using Formal Verification**" submitted by **Raj Vimalbhai Chokshi (15MECV05)**, towards the submission of the Project for requirements for the degree of Master of Technology in VLSI Design, 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.

Mr. Sandeep Asija
Engineering Manager
Intel Technology India Pvt. Ltd.
Bangalore

Declaration

This is to certify that

- a. 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.
- b. Due acknowledgment has been made in the text to all other material used.

-Raj Chokshi

Disclaimer

"The content of this thesis does not represent the technology, opinions, beliefs, or positions of Intel Technology India Pvt. Ltd., its employees, vendors, customers, or associates."

Acknowledgement

Though only my name appears on the cover of this dissertation, a great many people have contributed to its production. I owe my gratitude to all those people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever.

I would like to thank my Manager, Mr. Aman Malhotra (Engineering Manager, Intel Technology India Private Limited, Bangalore) for the valuable suggestions, feedbacks and persuasion to do something new and challenging. His guidance helped me a lot during my internship time not only for technical but on non-technical part too.

I would like to thank my Project Manager, Mr. Sandeep Asija (Engineering Manager, Intel Technology India Private Limited, Bangalore) for great support during my internship and give me freedom to work in my way. His immense knowledge and experience have a great impact not just on my project work but also at personal level too.

I would like to express my sincere gratitude to **Dr. Alka Mahajan** (Director of Institute of Technology, NIRMA University, Ahmedabad) for her continuous guidance and support. I Would like to take this opportunity to thank **Dr. N. M. Devashrayee** (Professor and Program Coordinator, M. Tech - EC (VLSI Design)), Internal Guide **Dr. N.P.Gajjar**(Professor and Program Coordinator, M. Tech - EC (Embedded System)) and all the faculties for their vision, support, and encouragement to provide me with the opportunity to carry out my project work in such a renowned and esteemed organization.

I would like to thank Ms. Navni Modi and Mrs. Chetna Halkati for great support to learn basics and give me a proper guidance and extended support in my difficult time.

Last but not the least I wish to thank my family and friends for their delightful company which kept me in good humor throughout the year and thus helping me complete the degree program successfully.

- Raj Chokshi
15MECV05

Abstract

Verification of SoC consumes about 70 percentage of the total turnaround time of design process. In this project the main focus area is to implement and verify high speed SoC(System on Chip), which will be working on data rate of 5Gbps. Also very new methodology i.e. PIP (Partition in partition) is being used in this project. So, this project presents a novel approach, which significantly reduces the formal equivalence verification challenges and also reduces the manual time spent by designers on debugging.

This project also presents the power aware equivalence verification problems, which has become an important part of SoC design in evolving technology nodes. With emerging trends in extended RTL modeling and shrinking time-lines, design team needs to adopt robust FEV(Formal Equivalence Verification) methodologies leaving no gaps in verification process to ensure quality.

These methodology enhancements and automated paranoia infrastructure to converge functional and power aware FEV are addressed in this project. In precise, FEV is one of the key activities in SOC design cycle and is a key convergence to ensure correct implementation of VLSI designs, by verifying the equivalence between the implementation and specification (RTL or gate) along with the low power implementation.

Another key objective of this project is ECO (Engineering Change Order), which is one of the crucial stage of product design cycle. In very less time, designers need to take care of such engineering change orders. Criteria for successful ECOs is that not only functionality but also timing and congestion need to be taken care of. This project discusses basics of ECO, some of the challenges and flow of Conformal ECO tool flow.

The deliverable of this project is to verify High speed SoC using formal verification. PIP methodology is been used to verify blocks and debug non equivalentents. Also successful ECO is generated at very crucial stage of Project.

Contents

Certificate	iii
Certificate	iv
Declaration	v
Disclaimer	vi
Acknowledgements	vii
Abstract	ix
List of Figures	xiv
1 Introduction	1
1.1 Introduction	1
1.2 Motivation	1
1.3 Objective	1
1.4 Requirements	2
1.5 Thesis Organization	2
2 Literature Review	3
2.1 Formal Equivalence Verification	3
2.2 Formal Equivalence Verification Vs Simulation	3
2.3 FEV In Implementation Design Cycle	4
2.4 Formal Equivalence Verification Tool Working Concept	5
2.5 Formal verification of high speed SoC	6
3 Formal Equivalence Verification Flow	10
3.1 Conformal Tool Flow	10
3.1.1 Setup Mode	11
3.1.2 Saving Log File	11
3.1.3 Specifying Black Boxes	11
3.1.4 Reading Libraries And Designs	11
3.1.5 Specifying Design Constraints	11
3.1.6 Specifying Modelling Directives	12
3.1.7 Switching To LEC Mode	12
3.2 LEC mode	13
3.2.1 Mapping Process	13

3.2.2	Compare Process	13
3.3	Partition in Partition methodology	15
3.3.1	Partition Level	16
3.3.2	Unit Level	16
4	Power Aware Formal Equivalence Verification	18
4.1	Low Power Verification	18
4.2	Need Of Low Power Verification	18
4.3	Conformal Low Power Equivalency Checks:	19
4.4	Power Aware Equivalency Checking Flow	20
4.4.1	Set Low Power Options	20
4.4.2	Read The Libraries And Designs	21
4.4.3	Read Power Intent And Comparison	21
4.4.4	Perform Equivalency Checking	21
4.4.5	Perform State Element Domain Consistency Checking	21
4.5	Power Aware FEV Flow Used In Industry	21
4.5.1	U2C flow	22
4.5.2	Native UPF Flow	22
5	Formal Equivalence Verification Challenges	23
5.1	Common Issue found in all Partition	23
5.1.1	Sequential Merge	23
5.1.2	Sequential Constant	24
5.2	Challenge 1: Timing	25
5.3	Challenge 2: Latch is moved from clock cone to data cone in synthesis	28
5.4	Challenge 3:Use of multi bit flops in synthesis causing mapping issue	29
5.5	Challenge 4: Abort Points Findings And Solutions	30
5.5.1	What Is Abort Points?	30
5.5.2	Resolutions For Aborts	31
5.6	Challenge 5: Failing Points Due To Extra Dummy Power Port	32
5.7	Challenge 6: Latches Folded To Flop Causes Non Equivalent	32
5.8	Challenge 7: Logical Connection Being Altered In APR flow	33
5.9	Challenge 8: ExtraBBOXOnGolden Side, NoOf Not-Mapped Points On Revised Side Because Of Maximum Number Of Iterations Limit.	34
5.10	Challenge 9: NEQs Due To Unmapped Flops In Golden And Revised Design In RTL2SYN	34
5.11	Challenge 10: Debugging Gets Difficult Due To Gated Clock Modeling Directive For SYN vs APR	35
6	ECO: Engineering Change Order	37
6.1	Basic of ECO	37
6.2	Importance of Conformal ECO Tool	38
6.3	Conformal ECO Flow	39

6.4	Challenges Faced For Generating Good Quality Patch	41
6.5	Challenge 1: Remove False NEQs To Generate Optimized Patch . . .	42
6.6	Challenge 2: NEQs At Different Hierarchies Cause Wrong Patch Gen- eration	42
7	Conclusion and Future Scope	43
7.1	Conclusion	43
7.2	Future Scope	43
	References	44

List of Figures

2.1	Simulation Vs Formal Verification	4
2.2	FEV In Design Cycle	5
2.3	ROBDD of Function	6
2.4	Role of FEV	7
2.5	Block diagram of high speed SoC	8
3.1	Conformal Tool Flow	10
3.2	Need Of Constraint To Disable DFT Logic	12
3.3	Need Of Sequential Constant Modelling	13
3.4	Mapping In Design	14
3.5	Block diagram of PIP	15
3.6	Partition Level	16
3.7	Unit Level	17
4.1	Power Aware LEC Flow	20
5.1	Sequential merge	23
5.2	Command used as solution	24
5.3	Result after applying required command	24
5.4	Sequential constant	25
5.5	Result after applying required command	25
5.6	Issue in timing	26
5.7	Use of VSDC file and manual mapping-1	27
5.8	Use of VSDC file and manual mapping-2	27
5.9	Use of multi bit flip flops causes mapping issue	29
5.10	Required solution for multi bit flop	30
5.11	Required command to solve multi bit flop	30
5.12	Logical Connection Changes In Apr	33
5.13	Unmapped Flop In Design	35
5.14	Gated Clock Modeling Directive	36
6.1	Basic ECO flow	38
6.2	Conformal ECO Tool Flow	39

Chapter 1

Introduction

1.1 Introduction

With the increasing complexity of designs, to make sure that the designs implementation is desired as per the RTL specification at the entry level is very crucial nowadays. The simulation-based approach is very much time consuming to validate the design and has various limitations. Due to these reasons, we are verifying the design using formal verification to ensure the correctness of the implementation of the design. Formal Equivalence Verification (FEV) has become an essential part of the implementation design flow. So it is being done at different stages of the implementation design flow.

1.2 Motivation

This is the era of speed. Every day in day to day life, the key focus is how work could be done efficiently and quickly. This project is all about 5G. Timing is major concern in this project. To meet timing designer is doing optimization in synthesis, due to this FEV is failing. There is always trade off between FEV and Timing. Thus identifying right solution for the FEV of high speed design is important aspect of this project.

1.3 Objective

With the increasing complexity of designs, to make sure that the design's implementation is desired as per the RTL specification at the entry level is very crucial nowadays. The simulation-based approach is very much time consuming to validate the design and has various limitations. Due to these reasons, we are verifying the design using formal verification to ensure the correctness of the implementation of the design. Formal Equivalence Verification (FEV) has become an essential part of

the implementation design flow. So it is being done at different stages of the implementation design flow. The reason to do FEV is to find any bugs or differences are there between the desired implementation and the actual one. We can track this using FEV very easily and quick manner. So main objective of this project is to find bug between RTL vs gate level implementation, also between gate and gate, and to fix them within product life cycle time.

1.4 Requirements

To complete this project at Intel, requires knowledge of backend design flow from synthesis to place route. Also knowledge of formal verification required.

Tools : Conformal by cadence, design compiler by synopsys.

1.5 Thesis Organization

Chapter 2 describes the literature review survey on formal equivalence Verification, its tool working concept, how FEV is better than simulation and Formal verification of High Speed Soc. It also describes PIP methodology.

Chapter 3 describes the importance of low power verification. It also describes about Power Aware Equivalency checking flow and Native UPF flow.

Chapter 4 describes about various challenges faced during Formal equivalence of High speed Soc.

Chapter 5 describes about Engineering Change order and how it is implemented in the very crucial stage of project.

Chapter 6 conclusion and future work of this project is presented.

Chapter 2

Literature Review

2.1 Formal Equivalence Verification

It is another way to verify the design rather than simulation. It will check the functionality of the design. It uses the mathematical technique to compare the two designs i.e. the golden and revised designs.

As we know that simulation will need input patterns in order to verify the design. If one has n number of inputs then to check the total functionality you need to give $2n$ input patterns to verify it. Time taken to verify it will also increase exponentially. But, to verify the same design using formal equivalence verification, we do not need any input patterns with complete functionality. However, in Formal Equivalence Verification, it will check only functionality independent of any physical properties.

2.2 Formal Equivalence Verification Vs Simulation

As we know that, we can verify the design using simulation which is a conventional approach to verify the design. However, we know that using this approach there will be several limitations.

In current scenarios, design complexity of the SoC is increasing exponentially. To verify this design is also a challenging work. As per the graph, you can see that as the design size increases, the verification effort that we will need, will increase further and we would not be able to verify the design in the less time. But as we know that from the current market perspective this type of time consumption approach would not be a good way to deal the verification. So in order to satisfy the requirement we are using Formal Equivalence Verification approach which not only less time consuming but it will assure the quality of verification.

Compared to Simulation we have some advantages of FEV which are listed below:

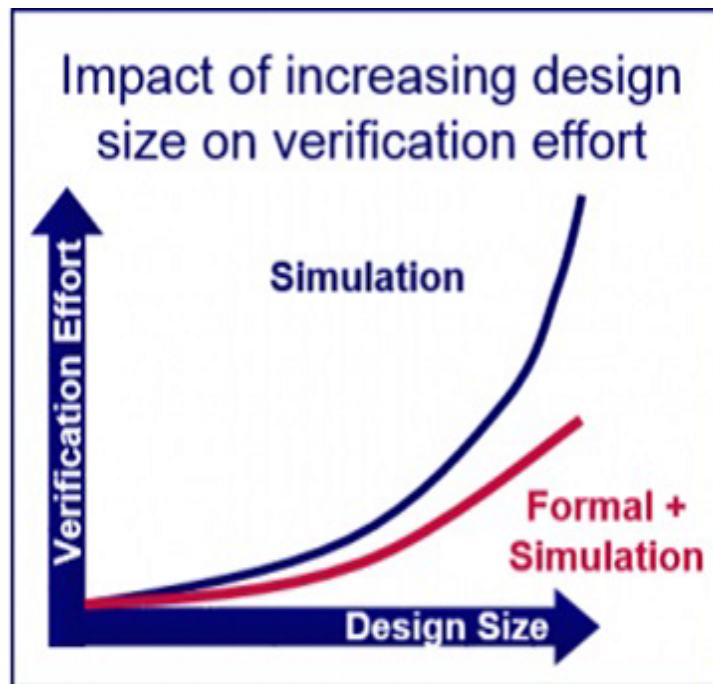


Figure 2.1: Simulation Vs Formal Verification

- No input patterns
- Earlier error findings
- More speed
- Complete functionality check

2.3 FEV In Implementation Design Cycle

As shown in figure 2.2, we can see that at each stage we are checking that functionality of the design is not getting lost in the whole implementation design cycle. Another purpose of checking the design at each stage is that if we check the design only in the last phase of the design then we would not be able to find at which stage the bug is coming. So for the easy debugging purpose we are checking the design at each and every stage of the design.

Based on the different types of Golden and Revised designs there are three types of FEV runs which we are running in design cycle:

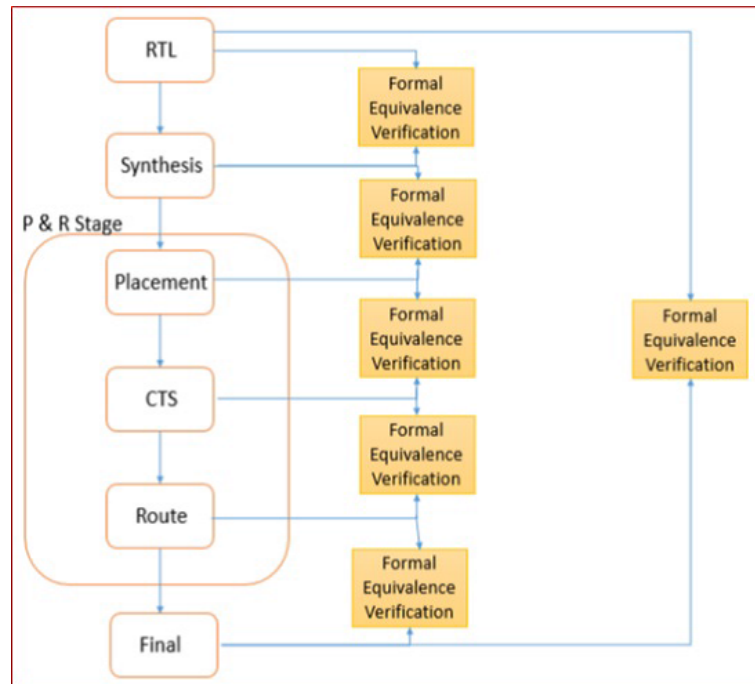


Figure 2.2: FEV In Design Cycle

- **RTL to Gate** : In this type of FEV, the Golden design would be RTL and on the revised side it will be a netlist. In the given physical design flow, RTL2SYN, RTL2APR, RTL2FINAL are under this category.
- **RTL to RTL**: In this type of FEV, Golden and Revised designs are RTL only. Usually, this kind of FEV run is being used between two RTL releases in the whole design cycle.
- **Gate to Gate**: In this type of FEV, Golden and Revised designs are netlists only. In the given physical design flow, SYN2APR, SYN2FINAL are under this category.

2.4 Formal Equivalence Verification Tool Working Concept

It uses basically mathematical equations. So tool will verify the designs using Reduced Order Binary Decision Diagram (ROBDD).

ROBDD is a canonical representation. If two functions have the same ROBDD then we can say that these two functions are equivalent. By this, we can say that there is no requirement of input vectors in order to verify the design.

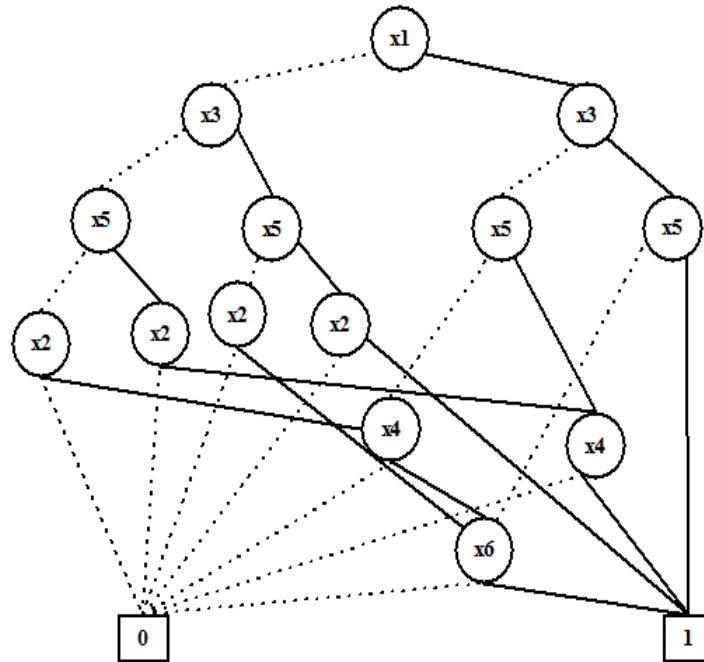


Figure 2.3: ROBDD of Function

Limitation of this method is that, the order of variables will change the ROBDD size, which in turn affects the memory, required to store it. Usually, some more sophisticated approach is required do the mathematical computation.

2.5 Formal verification of high speed SoC

In this project major focused area is to design high speed SoC design as a new emerging 5G technology. There is always trade off between timing and FEV. Here FEV team has to check logical equivalence between RTL and implementation team.

The modem's baseband chip pairs with a new 5G transceiver that enables both sub-6 Ghz and mmWave capabilities. This powerful combination incorporates key 5G NR (new radio) technology including low latency frame structures, advanced channel coding and massive MIMO to deliver faster connectivity and ultra-responsiveness.

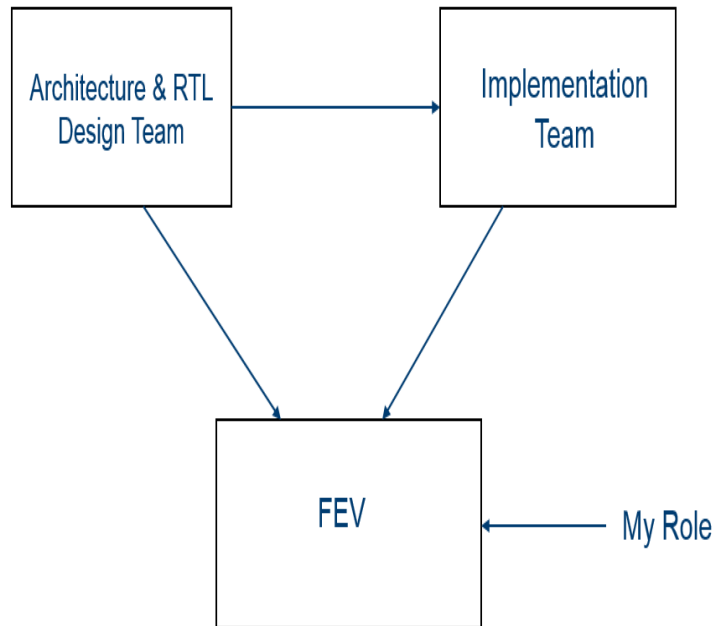


Figure 2.4: Role of FEV

More importantly, 5G will enable industries to improve our daily lives. With 5G, autonomous vehicles will be able to make decisions in milliseconds to keep drivers and vehicles safe. Drones will aid in disaster recovery efforts, providing real-time data for emergency responders. Smart cities will monitor air and water quality through millions of sensors, giving them insights needed to provide a better quality of life.

5G is a historic inflection point for the technology industry bringing seamless connectivity, massive computing power and rapid access to cloud resources for every person, thing and industry. This is at the forefront of the move to 5G with an unparalleled expertise across wireless, networking, cloud computing and data analytics the foundation of the 5G era.

Key Features: 5G Modem

- World's first single chip to support 5G operation in both sub-6 GHz and mmWave bands
- Achieves key 5G requirements, including expected speeds exceeding 5 Gbps,

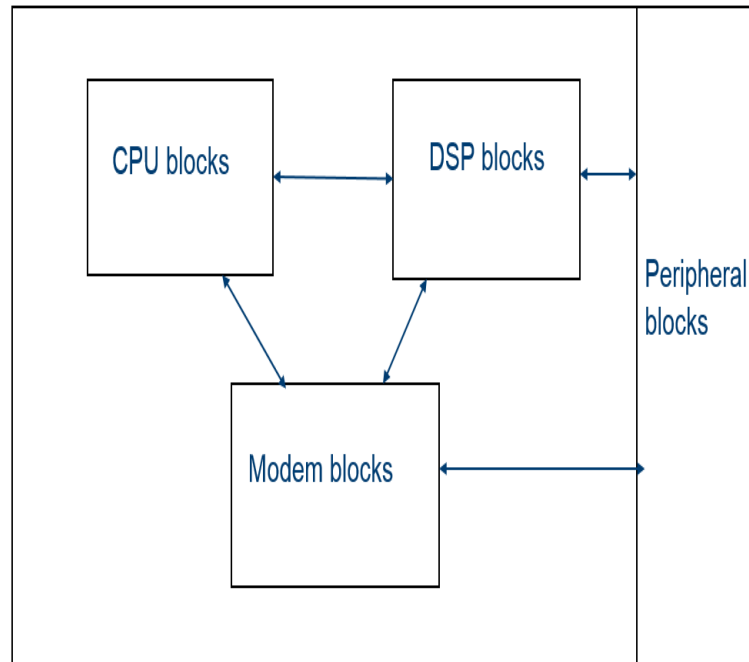


Figure 2.5: Block diagram of high speed SoC

hundreds of MHz of aggregated bandwidth and ultra-low latency

- Pairs with the world's first 5G sub-6 GHz RFIC and the mature 28 GHz 5G mmWave RFIC
- Compliant to multiple industry forum 5G specifications
- Supports key 5G NR technology features, including low latency frame structure, advanced channel coding, massive MIMO and beam forming

Key Focus Areas

- Autonomous Driving

This technology will enable the autonomous driving car. This car is capable of sensing its environment and navigating without human input.

- Virtual Reality

it will be used in Virtual Reality which uses headsets to generate the realistic

images, sounds and other sensations that replicate a real environment or create an imaginary setting.

- Smart Cities

5G will help enhance our homes through the emerging Internet of Things (IoT)

- Artificial Intelligence

There will be massive benefit in Artificial Intelligence from 5G

Chapter 3

Formal Equivalence Verification Flow

3.1 Conformal Tool Flow

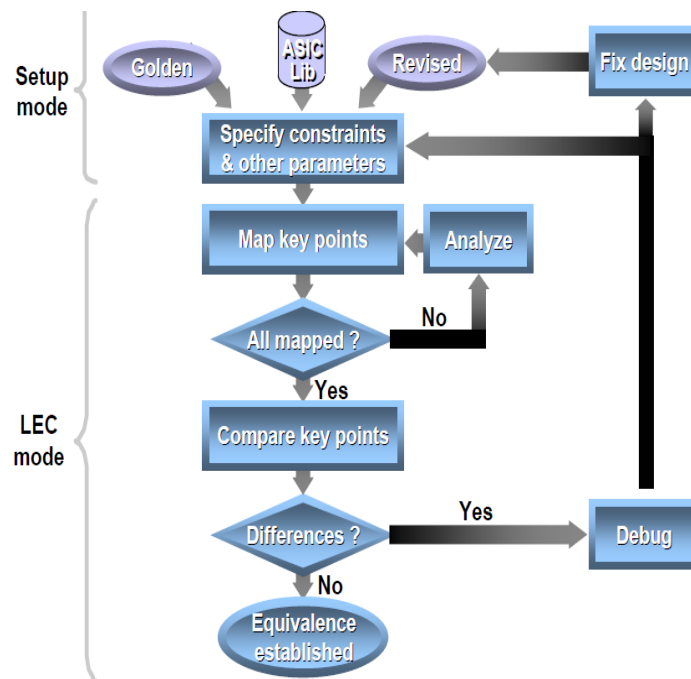


Figure 3.1: Conformal Tool Flow

In Conformal Tool there are basically two modes of operation:

1. Setup
2. LEC

In Setup Mode

Saving Conformal transcript to a log file

Specifying black boxes

Reading libraries and designs

Specifying design constraints

Specifying modeling directives

Switching to LEC mode

 In LEC Mode

Mapping process

Compare process

Reporting the run statistics

3.1.1 Setup Mode

In this stage, we will read all the collaterals required for the designs to compare, which means we will read libraries, RTL files, netlist and other inputs to run LEC.

3.1.2 Saving Log File

It is always a good practice to write log in order to track the errors or warnings we need to write the log file.

3.1.3 Specifying Black Boxes

In the given designs some modules which we do not want to check the functionality at the block level so for that we will black box those modules.

3.1.4 Reading Libraries And Designs

In this step, we will read all the required libraries and designs i.e. either RTL or netlist.

3.1.5 Specifying Design Constraints

In this step, we have to provide constraints in order to verify only logic, which we need to verify. As we know that in the design there will be scan logic, which is to improve yield. DFT (Design For Testability) logic would not be there in RTL. In order to avoid such extra designs, we need to constraint them. For that, we need to provide design constraints.

As you can see in order to verify design with scan logic we need to provide constraint for mux select signal. Here, we need to constraint select signal to 0 in order

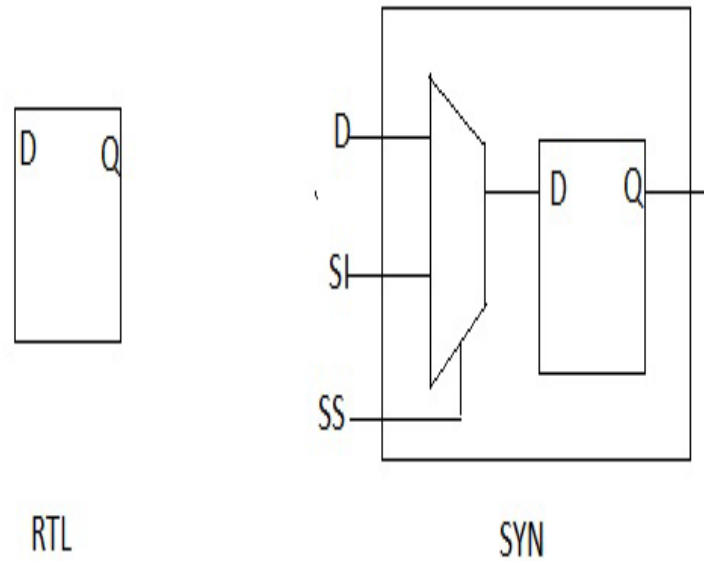


Figure 3.2: Need Of Constraint To Disable DFT Logic

to verify it properly.

3.1.6 Specifying Modelling Directives

In this step synthesis, tools will do optimization. So when we are comparing RTL to synthesized netlist the designs would not be equal because of such optimization. So in order to verify it, we need to provide modeling directives.

For an instance as shown in figure 3.3 we will have one flop whose input is always connected to 1. It is like just constant 1 signal. This kind of redundancy Design Compiler tool will optimize. Due to that, both side we will have a mismatch. It can cause false nonequivalent. To prevent such case in design, we need to provide modeling directive, which will make LEC understand the design.

3.1.7 Switching To LEC Mode

In this mode, it will map all the key points and logic cones in both designs. After mapping, there can be two results either it will be mapped or unmapped.

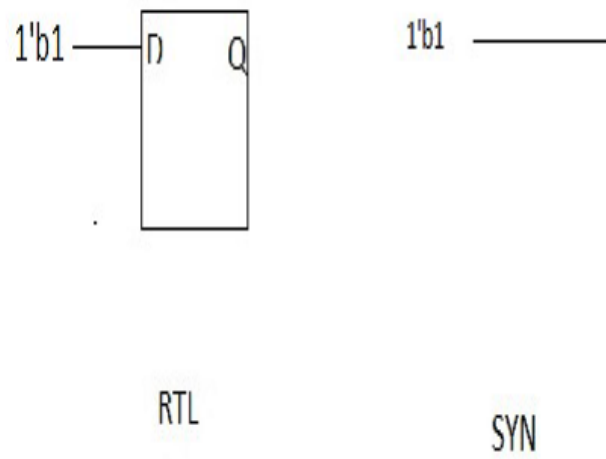


Figure 3.3: Need Of Sequential Constant Modelling

3.2 LEC mode

3.2.1 Mapping Process

Mapped Points: Key points are mapped on both golden and revised sides.

Unmapped Points: Key points are not mapped on either golden or revised sides.

In unmapped points, there are different types of unmapped points.

Extra: This type of key points is there on either golden or revised side.

Unreachables: This type of key points, which are not going to affect the output of the design.

Not Mapped: In this type of key points, which are not, they are on another side of the design.

3.2.2 Compare Process

After mapping is done, compared points are being compared.

Compared Points: POs, DFFs, D-Latches, Cut Points, and BBOX.

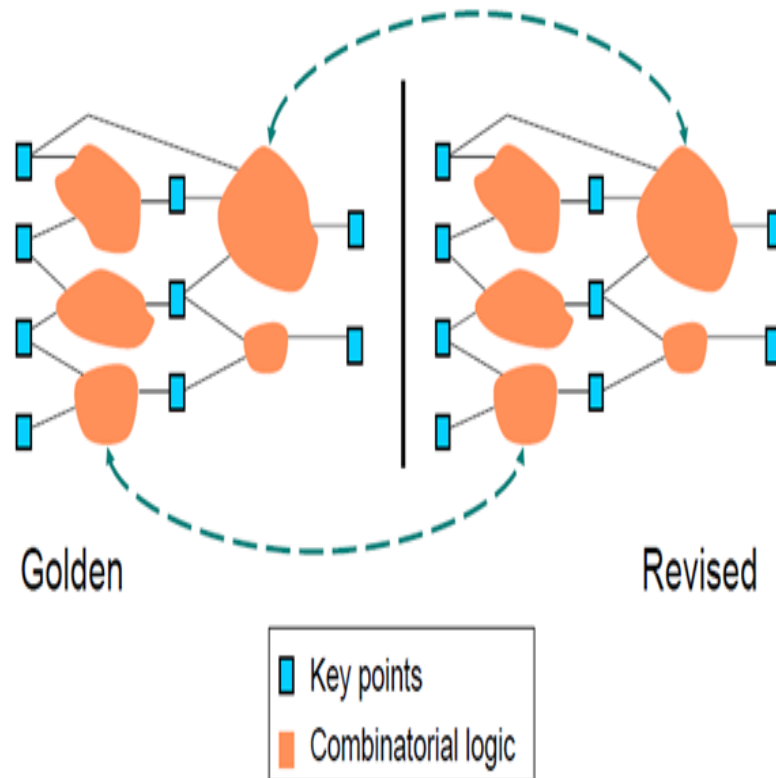


Figure 3.4: Mapping In Design

Comparison results for the design are listed below.

1. Equivalent: Compare points proven to be equivalent.
2. Inverted-Equivalent: Compare points proven to be complementary.
3. Non-Equivalent: Compare points proven to be different.
4. Abort: Compare points not yet proven to be equivalent or non-equivalent due to timeout or other system parameters.
5. Not-Compared: Compare points not yet compared.

The comparison can be done in two ways.

1. Flat
2. Hierarchical

In Flat Type: In this the whole design will be flattened one so there would not be any hierarchies to compare. Here the top module of design will get compared.

In Hierarchical Type: In this type of comparison, at each hierarchy design will get flattened and compared. In hierarchical comparison, there are two types of comparison.

1. Dynamic: In this type each hierarchy is getting flattened and compared. After comparison if it is equivalent then it will black box that module and go above hierarchy. If it is not equivalent then it will flatten the hierarchy and compared it.
2. Static: In this type each hierarchy is getting flattened and compared. After comparison if it is equivalent or not it will black box the current module and go above one hierarchy.

3.3 Partition in Partition methodology

In this Project very new methodology is used which is PIP. There is one partition which is called parent partition , contains child partition and unit also. While doing FEV child partition should be black boxed and it will be compared as separate unit. In this block the parent partition .which includes child-1 , child-2 and child-3 as child

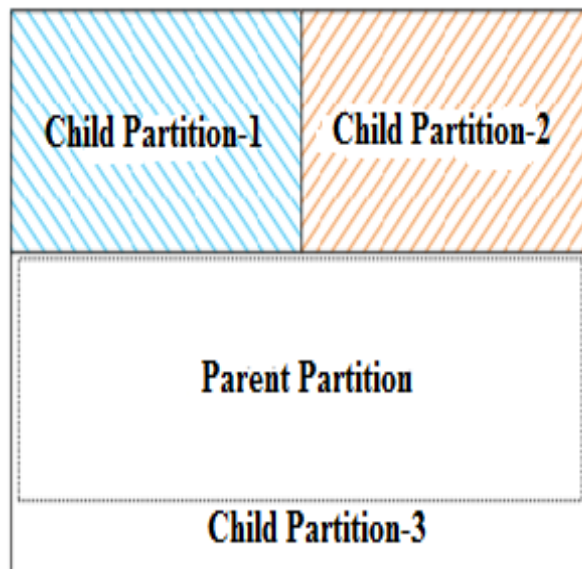


Figure 3.5: Block diagram of PIP

partition.

3.3.1 Partition Level

```

# Generated by: FEVE 3.1.07

# RTL FE env variables
# based on PATH_REGULATOR/-full_env section in finalised config
if ( ![info exists ACE_ENG] ) {
    if ( [info exists env(ACE_ENG)] ) {
        set ACE_ENG $env(ACE_ENG)
    } else {
        set ACE_ENG /nfs/pdx/stod/stodc04n02bL2/w.stnishim.100/soc-gor-a0-16m47-1115
    }
}
if ( ![info exists ACE_FWA_DIR] ) {
    if ( [info exists env(ACE_FWA_DIR)] ) {
        set ACE_FWA_DIR $env(ACE_FWA_DIR)
    } else {
        set ACE_FWA_DIR target/soc
    }
}

#All variables to be declared here (as depicted above)#

# partition RTL file
lappend VERILOG_SOURCE_FILES /nfs/pdx/stod/stodc15n03bH/w.zsheth.100/rfctrl_MMS2.4_RTL48.1_CWS2.1_flib_upf/collateral/rtl/rfctrl_top_part.v
if ( ![info exist G_DISABLE_DONT_TOUCH_DESIGN] ) {
    set G_DISABLE_DONT_TOUCH_DESIGN ""
}

# Units DDC files
lappend DDC_SOURCE_FILES SREPO_BOOF/target/soc/results/DC/soc_cdu_1/syn/outputs/soc_cdu_1.uniquify.ddc
lappend G_DISABLE_DONT_TOUCH_DESIGN soc_cdu_1
lappend DDC_SOURCE_FILES SREPO_BOOF/target/soc/results/DC/rfctrl_top_unit/syn/outputs/rfctrl_top_unit.uniquify.ddc
lappend G_DISABLE_DONT_TOUCH_DESIGN rfctrl_top_unit

```

Figure 3.6: Partition Level

This is the partition level RTL, its consist of its child unit's RTL. When we are reading partition RTL,we have to read all child unit's RTL also while doing FEV.

3.3.2 Unit Level

This is unit level RTL.When unit is compared stand alone,the parent level partition is black boxed.

```

if ( ! (info exist TCL_ARGS tdp_dma_top) ) {
  set Tdp_ARGS tdp_dma_top 1
  # setting ACE_ENG
  if ( ! (info exists ACE_ENG) ) {
    if ( ! (info exists env(ACE_ENG)) ) {
      set ACE_ENG $env(ACE_ENG)
    } else {
      set ACE_ENG /nfs/pdx/retod/stodc04m02bL2/w.stnshim.100/soc-gpr-a0-16m047-1115
    }
  }
}

#All variables to be declared here (as depicted above)

# source files from -
# /nfs/pdx/retod/stodc04m02bL2/w.stnshim.100/soc-gpr-a0-16m047-1115/target/soc/results/lint/tests/tdp_dma_top_test/tdp_dma_top_test/tdp_dma_top_rtl_lib.tdp_dma_top
lappend VERILOG_SOURCE_FILES /nfs/site/stod/stodc04m02bL2/w.stnshim.100/soc-gpr-a0-16m047-1115/subBlock/mms5/mms5_tdp_dma_top/modules/rmi_decoder/vlog/rmi_decoder.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_reg/GORAOP05RTLIV3/vlog/registers/tdp_registers_a0.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/tdp_ldpc_enc.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ldpc_packer.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ldpc_barrel_shifter.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ldpc_unpacker.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ldpc_lambda_mem.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ram_8021in_n196_r12.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ram_8021in_n196_r13.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ram_8021in_n194_r12.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ram_8021in_n194_r13.v
lappend VERILOG_SOURCE_FILES /nfs/site/diska/npshdk.ipr.002/ip_releasees/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_ldpc_enc/src/ram_8021in_n196_r23.v
...

set VLOG_LIBRARIES ( tdp_top_rtl_lib )
set file_list_vhdl_tdp_top_rtl_lib "/p/hdk/rtl/ip_releasees/hdk73/gpr/sip/mms5_tdp/GORAOP05RTLIV4/modules/tdp_soc_mem/src/tdp_mem_pucch_wrapper.vhd"
set file_list_vhdl_tdp_top_rtl_lib_p ""
lappend file_list_vhdl_tdp_top_rtl_lib_p \
/nfs/site/stod/stodc04m02bL2/w.stnshim.100/soc-gpr-a0-16m047-1115/subBlock/mms5/ram_delivery/aspens/common/util_pkg.vhd \
/nfs/site/stod/stodc04m02bL2/w.stnshim.100/soc-gpr-a0-16m047-1115/subBlock/mms5/ram_delivery/aspens/common/beh_ram_package.vhd \
/nfs/site/stod/stodc04m02bL2/w.stnshim.100/soc-gpr-a0-16m047-1115/subBlock/mms5/ram_delivery/aspens/common/sp_ram.vhd \
/nfs/site/stod/stodc04m02bL2/w.stnshim.100/soc-gpr-a0-16m047-1115/subBlock/mms5/ram_delivery/aspens/common/dp_ram.vhd \
...
if ( (info exist CTECH_TYPE) && (info exist CTECH_VARIANT) ) {
  set VERILOG_CTECH_FILES_ADD (concat $VERILOG_CTECH_FILES_ADD (glob /p/hdk/cad/ctech/cv16m043e_hdk141/source/p1273_3/././v/ctech_lib_*.sv))
}
set VERILOG_CTECH_FILES_REM (concat $VERILOG_CTECH_FILES_REM (glob /p/hdk/cad/ctech/cv16m043e_hdk141/source/p1273_3/././v/ctech_lib_*.sv))
set VERILOG_CTECH_FILES_ADD (lsort -unique $VERILOG_CTECH_FILES_ADD)
set VERILOG_CTECH_FILES_REM (lsort -unique $VERILOG_CTECH_FILES_REM)

```

Figure 3.7: Unit Level

Chapter 4

Power Aware Formal Equivalence Verification

4.1 Low Power Verification

Now a days increasingly complex SoCs are typically used in portable systems that must also support increasingly longer battery life and therefore must minimize power consumption. Even non-portable systems must avoid wasting energy, to minimize both power and cooling costs. In both cases, active power management is required to ensure energy efficiency. Although active power management enables the design of low power chips and systems, it also creates many new verification challenges.

Power-aware Formal Equivalence Verification will verify low power cells like retention registers, isolation cells, and power switches. It will check the power domains also. It will check the consistency of cells in same power domain as defined in golden UPF (Unified Power Format). In power aware formal equivalence verification, it will read UPF along with golden and revised design.

4.2 Need Of Low Power Verification

As per the design methods in Industry, in RTL we would not have information related to power supply. All the low power cells are just defined in the design but in which power domains the cell will get connected such information is available in UPF only. During implementation cycle, it is possible that implementation tool would not implement as per the specification in UPF. So in that case, Conformal Low Power verification is a promising solution to verify it. It will try to verify all low power cells, which are given below:

1. Clock Gating: It will help to reduce switching of the clock in order to reduce

dynamic power by disabling clock switching when the flop is not switching.

2. Power and Ground Clamp Cell: The diode clamp cell protects the data input from ESD (Electro Static Discharge) and also from voltage overshoots and undershoots on the signal net.
3. Isolation Cell: It is one of the low power cells, which prevents the un-driven signals in OFF block, which drives the signals of active or ON block. Isolation cells will at least provide 0 or 1 so that do not care conditions would not occur in the active block.
4. Level Shifter: It is one of the techniques to transport a signal from one voltage value to another. It can be for high to low voltage or low to high voltage shift.
5. Power Domain : It is the domain of which the power supplies will remain same for all the circuits under that power domain. All the circuits, which share the same, power source, and if switchable, will share the same power enabling and disabling control.
6. Power Gating: It is the technique to save leakage power, which switches a power domain's connection to a power source ON to OFF.
7. Signal Gating: It is the same as the clock gating technique. But here rather than clock data signals will be gated in order to reduce the power.
8. State Retention: It is a power management technique that saves the states before a block is powered down. It will save the state of memory elements before main power gets The circuit, which saves the state, will require non-interruptible power and ground to ensure the state preservation.

4.3 Conformal Low Power Equivalency Checks:

This tool will do the following checks:

1. Low Power Design Equivalency Checks
2. State Element Domain Consistency Checks
3. Can handle cell modeling, switch modeling and retention instances
4. Power Intent Comparison
5. State retention strategy comparison

4.4 Power Aware Equivalency Checking Flow

In the power aware equivalency checking flow involves the following tasks:

1. Set Low Power Options for power aware equivalency checking
2. Read the libraries and designs
3. Read power intent and then compare it
4. Perform equivalency checking
5. Perform state element domain consistency checking

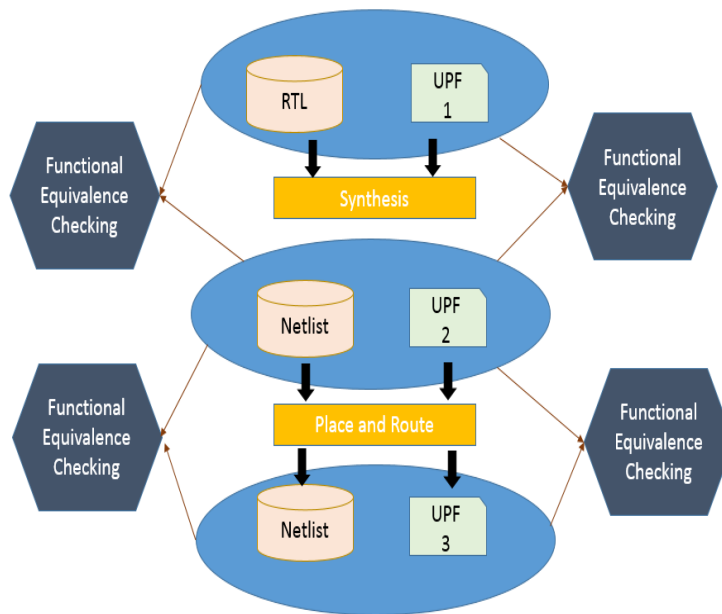


Figure 4.1: Power Aware LEC Flow

4.4.1 Set Low Power Options

In this option, we will define which type of netlist we wanted to read for low power verification. The netlist can be either logical or physical. So for RTL2SYN we will read logical netlist but for SYN2APR we will read logical netlist for golden design and physical netlist for revised design. In physical netlist, we will have netlist with power and ground connection.

4.4.2 Read The Libraries And Designs

In this step, we will read libraries and design. In this, we will read all the libraries with pg-pin option. Moreover, we will read libraries for all the low power cells like isolation cells, level shifters etc.

4.4.3 Read Power Intent And Comparison

In this step along with design and library, we will read the UPF / CPF (Common Power Format used in Cadence tools) for golden as well as revised design. In this step, it will report the power strategy differences like isolation rules or even naming differences also.

4.4.4 Perform Equivalency Checking

After this, we will verify the equivalency between two designs from the supply point of view. It will be done by commit power intent instruction. In this depends on the rules which we are checking, it will do equivalency checking for it.

4.4.5 Perform State Element Domain Consistency Checking

In this check, we will ensure that the state elements i.e. sequential elements are in the desired power domains as per specified in UPF. It will be checked by check low power cells instruction. It will check the retention strategy also as the desired one or not.

4.5 Power Aware FEV Flow Used In Industry

For Power Aware FEV there are two flows which are currently being used in Industry supported by Cadence Conformal Tool.

1. U2C flow
2. Native UPF Flow (Unified Power Format)

4.5.1 U2C flow

This is old flow, which is supported by Cadence. In this flow for two designs, which are being compared, the tool will read UPF for them and then it will convert into CPF (Common Power Format) internally. After that tool will compare the power intent of both designs. However, this flow is not giving correct violations in the design. Due to U2C conversion, it has reported some false violations in the design. This flow is not that much used in Industry.

4.5.2 Native UPF Flow

This is the latest flow, which is supported by Cadence. In this flow, both designs UPF will get compared. This flow has reported correct violations compared to U2C flow. In addition to, this flow has had several more rigorous checks compared to U2C flow, which is mentioned below:

- (a) UPF Lint Checks: In this check, it will read the UPF of design and then it will check the syntax of each and every file, which is used inside UPF. It will also check supply connection of the library of every cell or Macros in the design too.
- (b) Power Grid Comparison: In this check, it will check the supply connection of each cell in design to the respected power grid mentioned in design.
- (c) Power State Table: In this check, it will check the supply connection of each cell in design to the respected power grid mentioned in design.
- (d) Power Consistency Check: In this check, it will compare the supply connection for DFFs, DLATs, PIs, and POs of both designs. In designs there will be feed thru will be there at the block level. These blocks will be abutted at the full chip level. In order to maintain the supply connection of such feed thru ports, there will be set related supply net will be mentioned along with UPF. This kind of ports power consistency will get compared to the design using this check.

To conclude, Native UPF flow is having more advantages compared to U2C flow. Due to that, in Industry Native UPF flow is being used now a days.

Chapter 5

Formal Equivalence Verification Challenges

5.1 Common Issue found in all Partition

5.1.1 Sequential Merge

Duplicated register in the clock and/or data logic cone. It causes comparing problem. To resolve this problem it is required to perform figure 5.2 commands on tool.

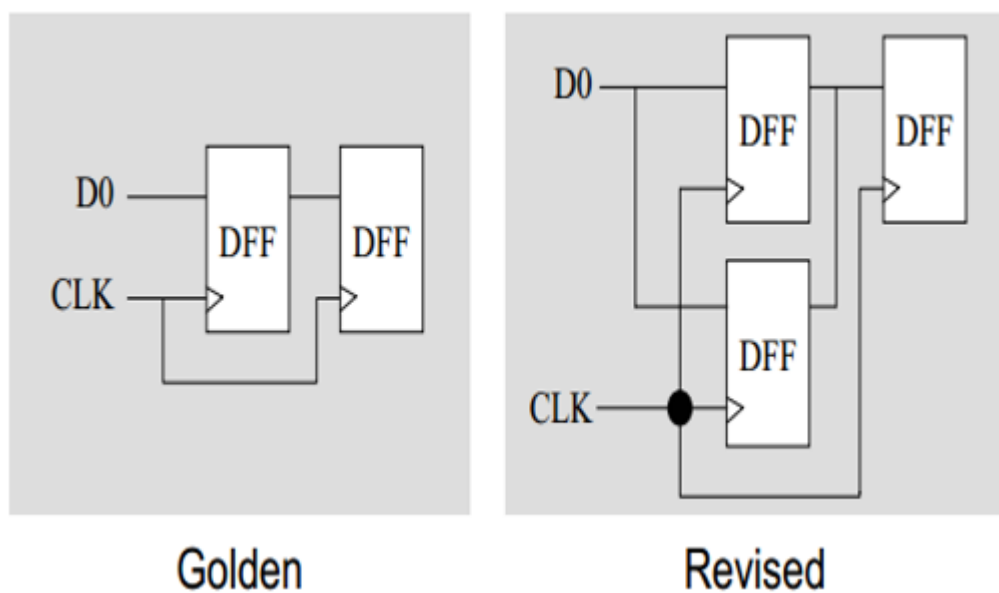


Figure 5.1: Sequential merge

As a result after applying required command on design figure below shows golden and revised both the side are same.

```

-----
// Command: set flatten model -nodff_to_dlat_zero
// Command: set flatten model -nodff_to_dlat_feedback
// Command: set flatten model -seq_merge
// Command: set flatten model -seq_constant
// Command: set flatten model -gated_clock
// Command: set compare effort auto

```

Figure 5.2: Command used as solution

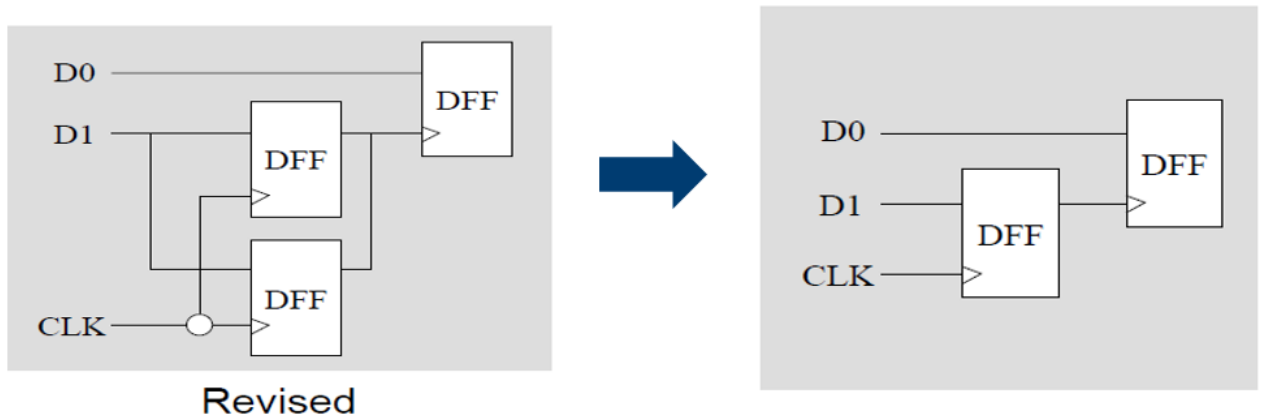


Figure 5.3: Result after applying required command

5.1.2 Sequential Constant

Sequential constant occurs due to the way the circuit is designed or a designer's preference to constrain the data port. It causes comparing problem. To resolve this problem it is required to perform figure 5.4 commands on tool. As a result after applying required command on design figure below shows golden and revised both the side are same.

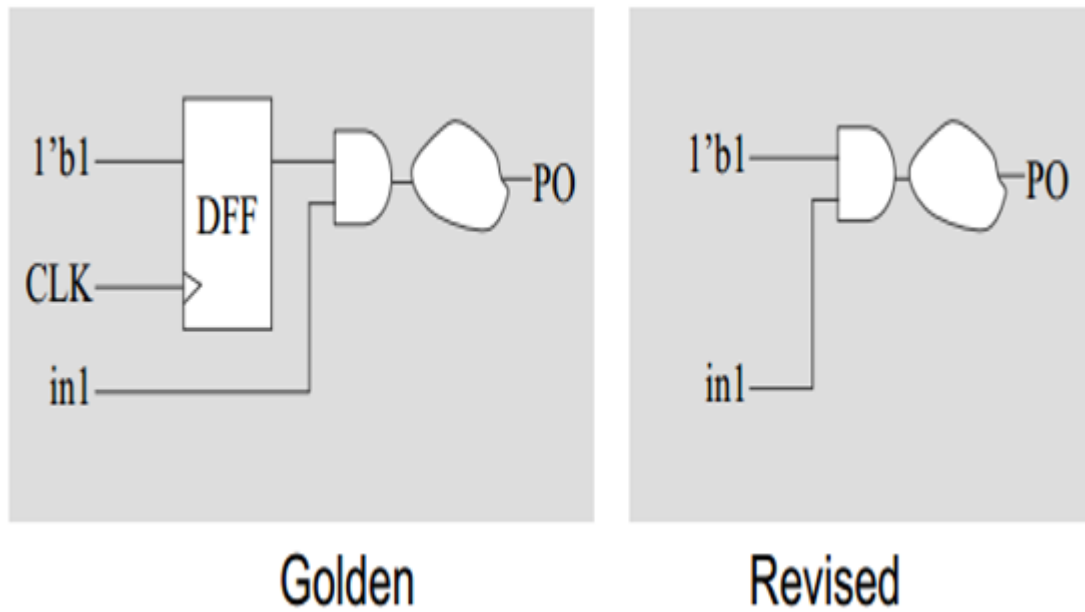


Figure 5.4: Sequential constant

```

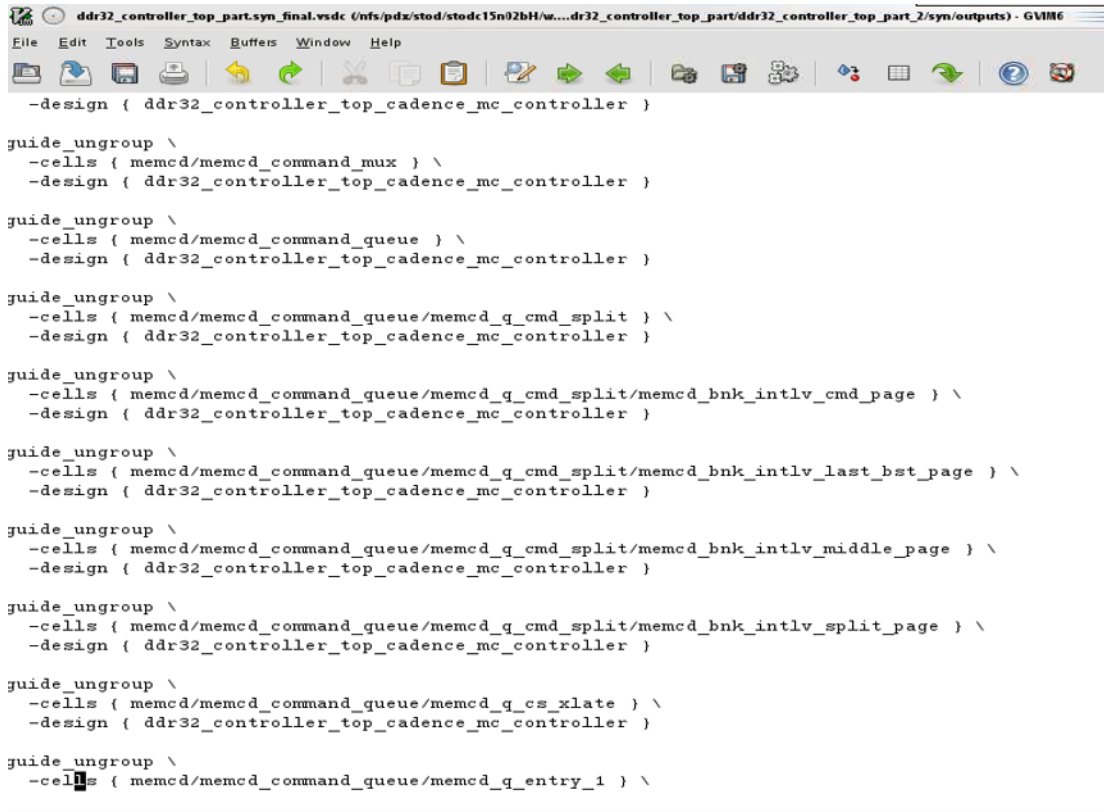
-----
// Command: set flatten model -nodff_to_dlat_zero
// Command: set flatten model -nodff_to_dlat_feedback
// Command: set flatten model -seq_merge
// Command: set flatten model -seq_constant
// Command: set flatten model -gated_clock
// Command: set compare effort auto

```

Figure 5.5: Result after applying required command

5.2 Challenge 1: Timing

Timing is very critical for many partition in this project. To meet timing ,designer is doing lot of optimization in synthesis. Due to this FEV tool is not able to compare properly golden and revised design. Due to ungrouping hierarchies timing is defiantly improved but FEV was failing.



```

-ddesign { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_mux } \
-design { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_queue } \
-design { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_queue/memcd_q_cmd_split } \
-design { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_queue/memcd_q_cmd_split/memcd_bnk_intlv_cmd_page } \
-design { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_queue/memcd_q_cmd_split/memcd_bnk_intlv_last_bst_page } \
-design { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_queue/memcd_q_cmd_split/memcd_bnk_intlv_middle_page } \
-design { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_queue/memcd_q_cmd_split/memcd_bnk_intlv_split_page } \
-design { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_queue/memcd_q_cs_xlate } \
-design { ddr32_controller_top_cadence_mc_controller }

guide_ungroup \
-cells { memcd/memcd_command_queue/memcd_q_entry_1 } \

```

Figure 5.6: Issue in timing

Solution

To resolve this issue, it is required to use VSDC file as shown in figure 5.7 and 5.8 and also to do manual mapping of some of the flops. After that tool is able to do correct mapping of golden as well as revised design. After this LEC is passing on this design.


```

guide_change_names \
  -design { ddr32_controller_top_part } \
  { { net *Logic0* n_Logic0_ } }

guide_change_names \
  -design { ddr32_controller_top } \
  { { net *Logic1* n_Logic1_ } \
    { net *Logic0* n_Logic0_ } }

guide_change_names \
  -design { ddr32_controller_top_cadence_mc_controller } \
  { { net *Logic0* n_Logic0_ } }

guide_change_names \
  -design { ddr32_controller_top_mc_dfs_control } \
  { { net *Logic1* n_Logic1_ } \
    { net *Logic0* n_Logic0_ } \
    { net N51 lp_ext_cmd_strb } }

```

Figure 5.7: Use of VSDC file and manual mapping-1

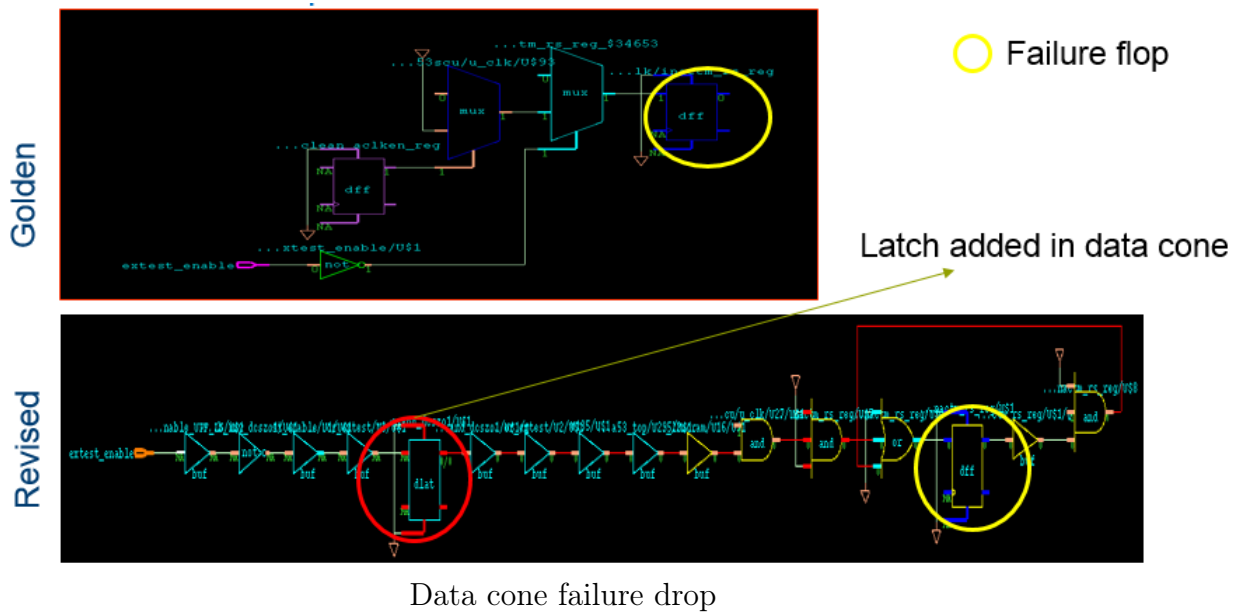
```

guide_change_names \
  -design { ddr32_controller_top_cadence_mc_memcd_param }
  { { net *Logic1* n_Logic1_ } \
    { net *Logic0* n_Logic0_ } \
    { net _cse_31 n_cse_31 } \
    { net _cse_33 n_cse_33 } \
    { net _cse_2795[0] n_cse_2795[0] } \
    { net _cse_2798[0] n_cse_2798[0] } \
    { net _cse_2829 n_cse_2829 } \
    { net _cse_2834 n_cse_2834 } \
    { net _cse_2881[37] n_cse_2881[37] } \
    { net _cse_2881[36] n_cse_2881[36] } \
    { net _cse_2881[35] n_cse_2881[35] } \
    { net _cse_2881[34] n_cse_2881[34] } \
    { net _cse_2881[33] n_cse_2881[33] } \
    { net _cse_2881[32] n_cse_2881[32] } \
    { net _cse_2881[31] n_cse_2881[31] } \
    { net _cse_2881[30] n_cse_2881[30] } \
    { net _cse_2881[29] n_cse_2881[29] } \
  }

```

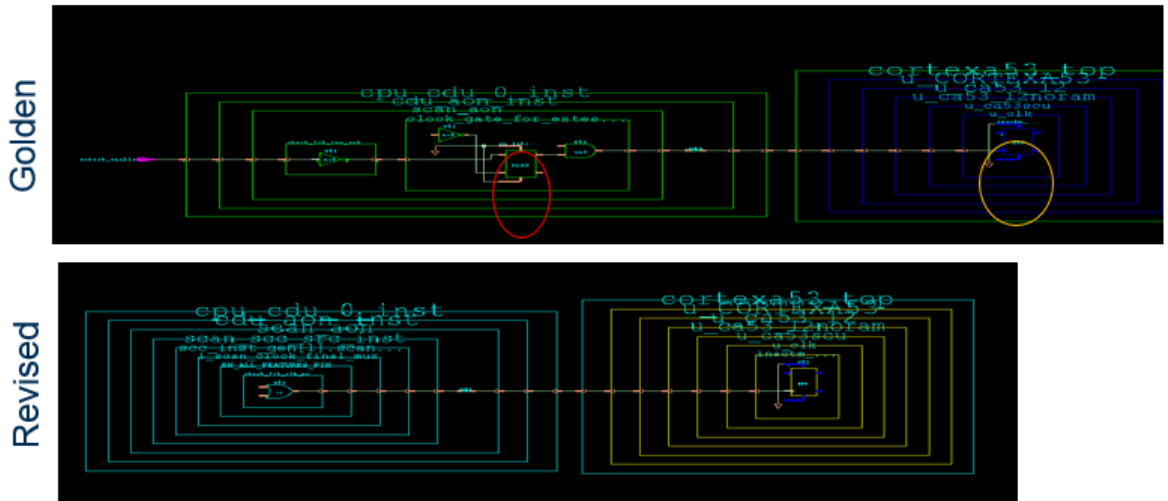
Figure 5.8: Use of VSDC file and manual mapping-2

5.3 Challenge 2: Latch is moved from clock cone to data cone in synthesis



In one of the block during synthesis latch is moved from clock cone to data cone. As shown in figure Same Latch which is was there in data cone of revised netlist (latch followed by AND forming a clock gate). This latch is a ctech cell. This latch followed by AND gate not present in Revised.

Solution: This was bug in synthesis tool, which is reported to designer. To solve this issue designer reported this issue to application engineer of tool and fixed it.



Clock cone cone failure drop

5.4 Challenge 3: Use of multi bit flops in synthesis causing mapping issue

```

d04f4u00wd8a5 \mbit_clusteridaff2_reg[7]_mbit_clusteridaff2_reg[6]_mbit_clusteridaff2_reg[5]_mbit_clusteridaff2_reg[4] (
    .si1(n2525), .d1(gov_clusteridaff2_i[7]), .ssb(n2888), .clk(n3084),
    .o1(clusteridaff2[7]), .si2(n2525), .d2(gov_clusteridaff2_i[6]), .o2(
        clusteridaff2[6]), .si3(n2525), .d3(gov_clusteridaff2_i[5]), .o3(
            clusteridaff2[5]), .si4(n2525), .d4(gov_clusteridaff2_i[4]), .o4(
                clusteridaff2[4]) );
d04f4u00wd8a5 \mbit_clusteridaff1_reg[3]_mbit_clusteridaff1_reg[2]_mbit_clusteridaff1_reg[1]_mbit_clusteridaff1_reg[0] (
    .si1(n2525), .d1(gov_clusteridaff1_i[3]), .ssb(n2888), .clk(n3084),
    .o1(clusteridaff1[3]), .si2(n2525), .d2(gov_clusteridaff1_i[2]), .o2(
        clusteridaff1[2]), .si3(n2525), .d3(n3560), .o3(clusteridaff1[1]),
    .si4(n2525), .d4(gov_clusteridaff1_i[0]), .o4(clusteridaff1[0]) );
d04f2u00wd8a5 \mbit_cpu_id_rs_reg[1]_mbit_cpu_id_rs_reg[0] ( .si1(n2525),
    .d1(cpu_id_i[1]), .ssb(n2888), .clk(n3084), .o1(cpu_id_rs[1]), .si2(
        n2525), .d2(n2532), .o2(cpu_id_rs[0]) );
d04f4u00wd8a5 \mbit_ctr_cwg_rs_reg[3]_mbit_ctr_cwg_rs_reg[2]_mbit_ctr_cwg_rs_reg[1]_mbit_ctr_cwg_rs_reg[0] (
    .si1(n2525), .d1(ctr_cwg_i[3]), .ssb(n2888), .clk(n3084), .o1(
        ctr_cwg_rs[3]), .si2(n2525), .d2(ctr_cwg_i[2]), .o2(ctr_cwg_rs[2]),
    .si3(n2525), .d3(ctr_cwg_i[1]), .o3(ctr_cwg_rs[1]), .si4(n2525), .d4(
        ctr_cwg_i[0]), .o4(ctr_cwg_rs[0]) );
    
```

Figure 5.9: Use of multi bit flip flops causes mapping issue

In synthesis by using multi bit flops we can reduce the clock power and also area. By using multi bit flops power is reduced from 39 mW to 12 mW. Due to multi bit tools are not able to do mapping correctly.

Solution: To solve this problem I have used following command to tell tool that

```

##setvar G_EXIT_ON_UPF_ERROR 0
setvar G_COMPARE_PST 0
setvar G_CONF_ANALYZE_SETUP 1
setvar G_CONF_COMPARE_THREADS 4
##setvar G_REF_PAR_RTL 0
setvar G_SET_MULTIBIT_OPTIONS "-prefix \"mbit\" -delimiter \"_mbit_\""
setvar G_CONF_PHASE_MAP 1

```

Figure 5.10: Required solution for multi bit flop

```

// Command: set multibit option -prefix "mbit" -delimiter "_mbit_"
// Command: report modules -inst -golden > reports/ca53_cpu.inst.golden.rpt
// Command: report modules -inst -revised > reports/ca53_cpu.inst.revised.rpt

```

Figure 5.11: Required command to solve multi bit flop

these are the multi bit flops. After that tool is able to map correctly and also did compare successfully and LEC was pass on this block.

5.5 Challenge 4: Abort Points Findings And Solutions

5.5.1 What Is Abort Points?

An abort point is a compare point that has not been conclusively compared. So there can be different reasons to have aborts in the design. Due to large sized cones, complex logic implementation and several do not care in the design will cause aborts in the design. In order to compare the design, the tool will set some time limit to compare it but if it goes beyond that then it would not compare, results in aborted points.

5.5.2 Resolutions For Aborts

Step1 In order to resolve the aborts we first need to know the cause of aborts. By analyzing abort instruction tool will help us to understand the causes of aborts.

Step2 After that if there are any nonequivalent points then we need to debug it.

Step3 Afterward we can try techniques to solve the abort points, which are given below.

Approaches:

1. Increase the compare effort which will consider more complex algorithms to resolve abort points.
2. Try to run it using the hierarchical run. Logic cone size will be less compared to flatten one.
3. Abort points can come due to the complex data path. Here, the problem is in RTL design any complex operation addition or multiplication can be implemented in synthesis in multiple manners depend on design constraints. Due to that, LEC would not be able to figure it out how the design is being implemented in synthesis in less time. Due to that, abort can occur. In order to resolve such aborts, synthesis tool will provide resource report. In that report, synthesis tool will provide how it implemented complex operations. After parsing this report LEC will understand the design and it will able to resolve abort.

In Flat full chip timing analysis we need to gate level netlist along with SDF/SPEF, the timing libraries and the design constraints. Using this approach designers wait till a blocks completion prior to performing full chip timing. Hierarchical STA flows allows us to partition different blocks using the Timing Models which should completely model the input/output timing characteristics without requiring the complete netlist of the block and it do not need to model every path in the block.

5.6 Challenge 5: Failing Points Due To Extra Dummy Power Port

In one of the design, there are some failing points. Usual analysis steps are given below:

1. Before diagnosing the failing points, we need to check some unmapped points in the design.
2. All unmapped points are z-points. To know the reason behind unmapped z points is coming we need to see the modeling messages report. In that, there is a modeling message, which is F32, which says that it will tie z gate to floating ports.
3. By the name of the port, it was a power port. That port is not defined in the UPF.
4. To make sure that no other problem in the design I have constrained the port to 1.
5. With constraining the port to 1 since the dummy port was power port, design is equal. This extra dummy power port is inserted by design flow of ICC tool. We reported this dummy port information and then it got resolved.

5.7 Challenge 6: Latches Folded To Flop Causes Non Equivalent

Modeling directives are required in order to give information to LEC about how synthesis tool have modeled the design. So if it is not properly given then the design will have nonequivalent points. In order to make sure that LEC will get proper modeling directives, the Synthesis tool will dump the .vsdc file, which is for 3rd party verification tool. But sometimes LEC tool will not able to understand the design and it will do over modeling in the design. In one of the design, we have seen some unmapped points in the design as stated in the problem statement. Here the problem was somewhat unusual. Analysis steps are given below:

1. First, we have checked the unmapped points, so as per our analysis, on the golden side we have the unmapped latch and on the revised side we have an unmapped flop, the interesting part was that both have the same name.
 2. After that when we check the modeling messages, in that we have seen F5 modeling message, which is related to Latch Fold. According to this modeling directive, two latches on revised side have been modeled as a flop.
 3. From it means there are two latches on revised side. It is being modeled as a flop in revised. By seeing the unmapped latch on the golden side we analyzed that on the golden side also there are two latches. To make sure that two latches will get mapped on both sides we have disabled this modeling directive. This will resolve the issue.
- Solution: In order to solve the problem we have disabled the latch unfold modeling directive in order to map those key points.

5.8 Challenge 7: Logical Connection Being Altered In APR flow

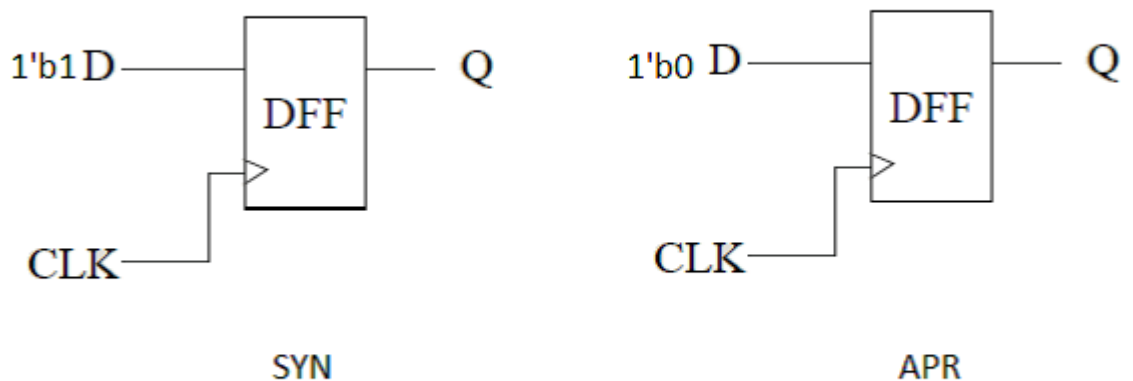


Figure 5.12: Logical Connection Changes In Apr

As shown in above diagram, above in SYN design some of the D-flop Dinput is

connected to 1. But somehow in APR flow after optimized placement stage, all these D-flop D input is connected to 0. Due to that, all these flops are coming in Non-Equivalent points. It was a bug in ICC tool.

Solution: In order to resolve this problem, we have to disconnect all these flops connection from ground to Tie high cells or Supply.

5.9 Challenge 8: ExtraBBOX On Golden Side, No Of Not-Mapped Points On Revised Side Because Of Maximum Number Of Iterations Limit.

In one of the design, there is RTL file, which has loop unrolling inside it. By unrolling the loop will implement the same kind of logic at the hardware level. Based on loop index value, the hardware will be implemented in synthesis stage. When LEC is running for RTL Vs SYN, the loop index value was by default 8192 in LEC tool. But as per the RTL 17 requirement, loop index value required was 30000. Due to the default value, while reading RTL, LEC was not able to replicate logic as they are in Synthesis. Due to that, it was causing extra BBOX on RTL side and another way, in synthesis LEC, was not able to map design properly, which in turn caused not mapped points on synthesis side.

Solution: In LEC there is command set HDL options -MAX FOR LOOP SIZE 30000 inch, which increases the loop index value as required one.

5.10 Challenge 9: NEQs Due To Unmapped Flops In Golden And Revised Design In RTL2SYN

As per the description, one of the failing point in the design is PO (Primary Output). One of the unmapped flop output is driving the same PO on both sides. Because these flops are not mapped, will not get compared. Usually, flops should map based on name or functional base. The reason behind the flops neither being mapped as

name based nor functional based is because of a vsdc file. This file is used for 3rd party verification tool, dumped by Synopsys DC tool, which will capture all name changes information, sequential constant flops name etc. This file was being read by LEC was not up to date one. Due to that, these flops are not being mapped properly, which causes PO as one of the FP.

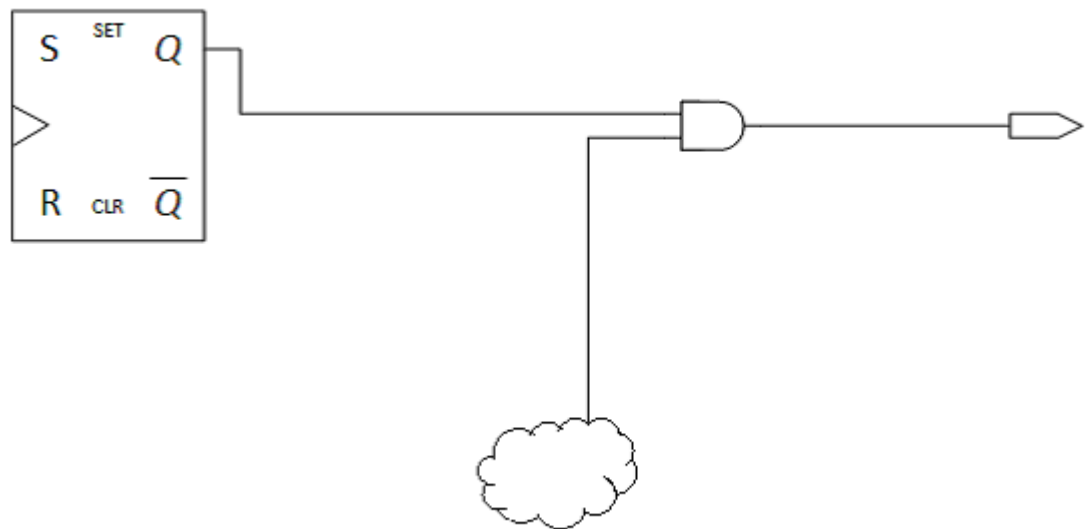


Figure 5.13: Unmapped Flop In Design

Solution: Use up to date vsdc file or disable the use of a vsdc file.

5.11 Challenge 10: Debugging Gets Difficult Due To Gated Clock Modeling Directive For SYN vs APR

In one of the design, between SYN (Synthesis) Vs Route (APR) netlist run, there are 8 NEQs. In all the NEQs debugging gets difficult. One of the reasons is big logic cone size. Another reason, there is a huge mismatch in terms of logic cone size. For an instance on SYN side, it is around 800 key points, on APR side; it is around 400

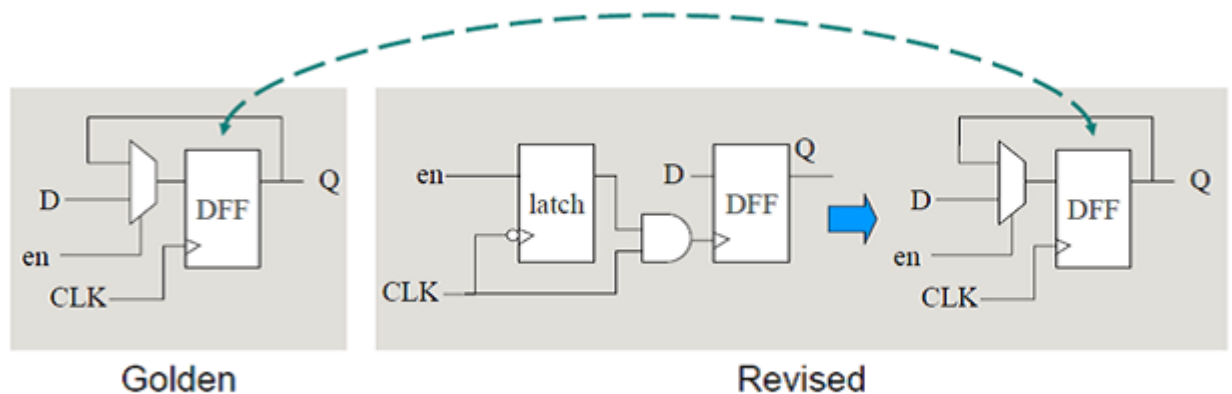


Figure 5.14: Gated Clock Modeling Directive

key points. Many flops are coming as Non corresponding support on SYN side. All these flops on SYN side are mapped on APR side. Using LEC we are not able to debug the design. In order to resolve this NEQ, I have tried to generate a patch using Conformal ECO tool. In ECO flow, while generating a patch, it also compared the design as well. No of NEQs should match to normal LEC runs. But in ECO-run, I got some more NEQs in the design. All these extra NEQs are DLAT, which is part of a clock gate. The reason behind these NEQs, which are not there in LEC run, was gated-clock modeling directive. In both SYN and APR design, clock gate latch gets modeled as MUX structure. All these DLATs will go into unreachable. So it would not get compared at all. After debugging these DLATs the problem of mismatch in terms of logic cone size get resolved. Since the logic cone mismatch in terms of non-corresponding support gets reduce to zero then it helps us to address the real problem in design, which gets resolved using Conformal ECO tool by generating a patch.

Solution: For this case, between SYN and APR, if we have disabled gated-clock modeling directive then it would have resolved the problem with much less effort.

Chapter 6

ECO: Engineering Change Order

6.1 Basic of ECO

Engineering Change Order is a process to make changes in design without running whole design cycle like synthesis, APR etc.

There are two types of ECOs which are mentioned below:

1. Functional ECO
2. Non - Functional ECO

1. Functional ECO: These types of ECOs are related to functional changes in design. This type of ECOs is because of some RTL bugs, which got identified, in the last phase of the design cycle.

2. Non-Functional ECOs: It can be related timing, cross talk, DRV, and routing violations with minimal effort.

ECOs can be in two stages:

1. Pre Mask ECOs: Pre-mask ECOs are performed during Place and Route and before the design is taped out. Uses normal logic gates to implement change.
2. Post Mask ECOs: Post-mask ECOs are performed after the design goes to manufacturing. Uses spare gates only to implement change.

In order to implement functional changes in the design, conformal ECO tool is being used in Industry. Using this tool you can only implement functional ECOs

6.2 Importance of Conformal ECO Tool

In product design cycle, RTL verification and implementation are running in parallel. In this process, in the last phase of a design cycle, some of the RTL bugs will get detected. Depend on the RTL bug, the feasibility of Functional ECO will be decided. If ECO is simpler one then it will be implemented by careful manual work. If ECO is complex one then it can't be implemented manually. In that case to incorporate ECO, re-synthesis the design, will be the only option left with the designer. But that will cause the delay for tape - out. That will affect directly on Return on Investment significantly. In order to implement even complex ECO, Conformal ECO is one of the promising solutions as per the Industry Standards.

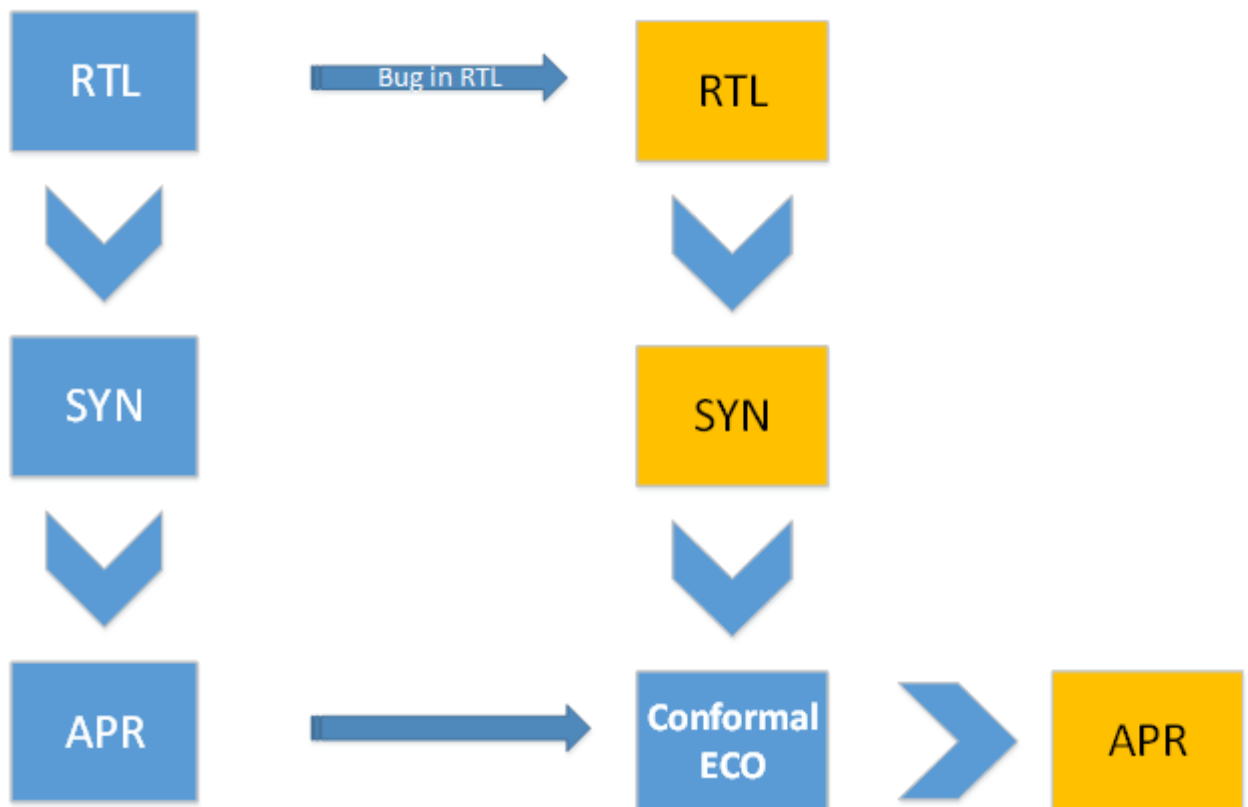


Figure 6.1: Basic ECO flow

6.3 Conformal ECO Flow

Before using conformal ECO tool, we have to take care of few steps.

1. RTL1 Vs G1, verification has to pass.
2. Resynthesize the design [G2 netlist generation] using the same way used for G1 generation for new RTL.
3. Verify new RTL vs G2 netlist.

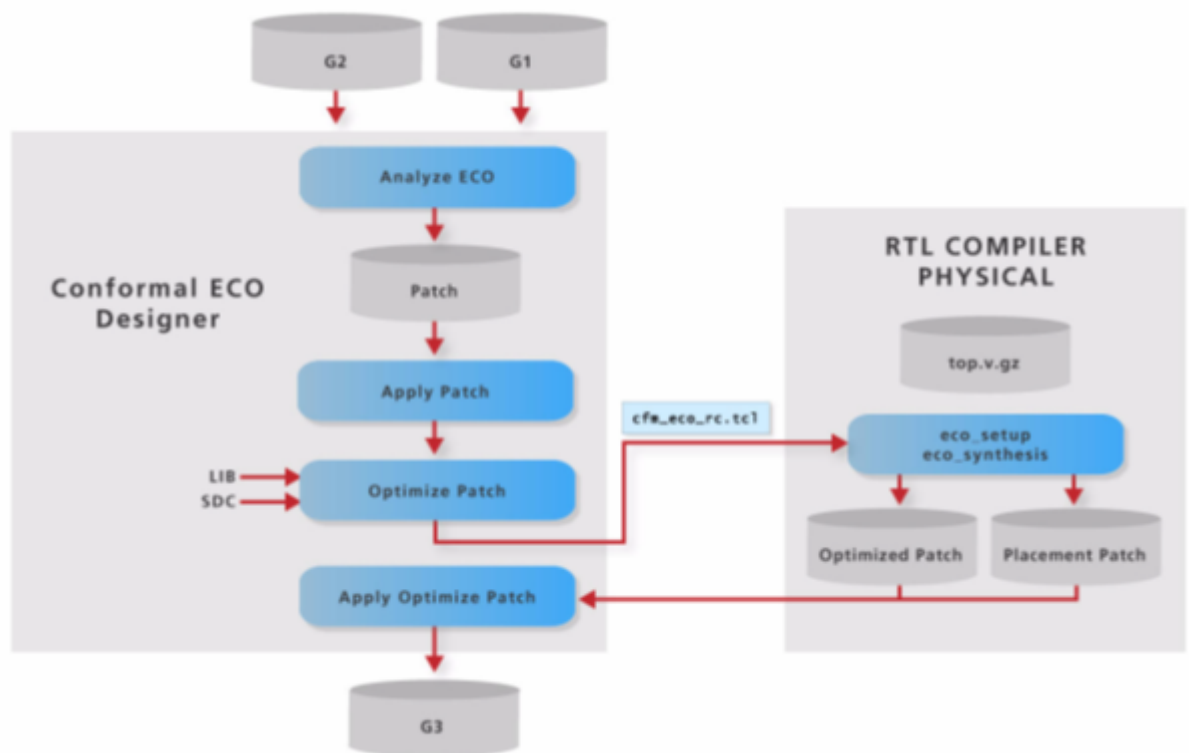


Figure 6.2: Conformal ECO Tool Flow

If all these steps are done then we can proceed for patch generation using ECO tool.

Conformal ECO flow as following stages:

1. Read library

2. Read Golden design [Pre-ECO netlist]
3. Read Revised design [Post-ECO netlist]
4. Map Setup
5. Gen ECO
6. Verify
7. Reports

1. Read library

In this stage, the tool will read all standard cells library.

2. Read Golden design [Pre-ECO netlist]

In this stage Pre ECO netlist, which does not have ECO, implemented. It can be synthesis with scan netlist or APR final netlist.

3. Read Revised design [Post-ECO netlist]

In this stage Post ECO netlist, which has ECO, implemented. It will be either SYN netlist without a scan or with a scan.

4. Map Setup

In this stage, the mapping is being done. If a scan is there in the design then scan constraints also get applied in the design so that only functional path will be there.

5. Gen ECO

In this stage there are multiple steps, which are mentioned below:

(a) Gen Patch:

- In this step, both Pre-ECO and Post-ECO design will get compared. Comparison again can be done in either flat/hierarchical manner. There will be NEQ in the design since the design has some change in that.

Note: Flat comparison is recommended way to generate patch.

- It will analyze the NEQ and generate patch based on the difference between two designs at each hierarchy.

- In patch, it will use generic library cells in order to implement in design. It won't be optimize one.

(b) Apply Patch:

In this step, whatever the patch tool has generated it will apply on design.

(c) Optimize Patch:

- In this step, RC tool will be invoked and map the generic library cells into standard cell library cells.

- RC tool even tries to optimize the patch in this step.

- After optimized patch, it will apply to design. It will generate the netlist as well as Patch after it.

6. Verify In this stage, it will verify the ECOed golden netlist with the Post-ECO netlist. From this stage, we will be sure that patch what tool has generated is successful or not.

7. Reports In this stage it will dumped out patch file and Verilog netlist.

6.4 Challenges Faced For Generating Good Quality Patch

For ECO stage, only patch generation is not the only task but a patch should be optimized one in terms of no of changes and also should not affect the routing congestion and timing. Patch generation can be done in using Flat comparison or Hierarchical Comparison or Flattened ECO. Flattened ECO is one of the recommended ways for patch generation.

6.5 Challenge 1: Remove False NEQs To Generate Optimized Patch

Before the generation of the patch, we need to run FEV between latest RTL [With ECO] and netlist [Without ECO]. In that case, no of NEQs are 2. This number, will give us idea how many NEQs you should get while generation of Patch. However, for one of the block, we are getting around 3K NEQs. Once we generated the patch, size of a patch was too large and another outcome was patch generation was not successful. Solution: In order to solve the above problem we need to find the reason of those False NEQs then after we have generated the optimized patch. The reason behind those False NEQs was no phase mapping. Once we enable in the flow, no of NEQs came down to 2 as expected.

6.6 Challenge 2: NEQs At Different Hierarchies Cause Wrong Patch Generation

In one of the design, there are numerous complex logic changes between RTL with ECO and netlist without ECO. Those changes are there between in several hierarchies. In order to generate a patch using Flat approach was not at all successful. Another problem was, due to optimization happened between Synthesis and APR.

Solution: In order to solve this, we have analyzed the hierarchies which getting affected. After that, we have run hierarchical FEV between RTL with ECO and netlist without ECO. After that, we have generated a patch for that particular hierarchy without flattening it. Once we are successful in one hierarchy we have repeated the same procedure for all the hierarchies, which have RTL changes. Here, we have followed Hierarchical approach without flattening rather than flattened one [recommended one].

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

In Industry Formal Equivalence Verification is one of the key activities for verification of the design. It is being done at a different level of design cycle from RTL to Final Layout. It supports different kind of format like RTL, Gate-level netlist, and Spice netlist and library verification too.

For low power implementation verification, we can rely on Power Aware FEV. Native UPF flow is one of the promising solutions supported by Cadence. It is being used nowadays in Industry because of more rigorous checking.

This project has covered how to verify high speed design and its challenges faced. Also using PIP how to do FEV is also discussed.

7.2 Future Scope

The main focus of this project is High Speed SoC. This chip will be used in Autonomous driving and virtual reality. Drones will aid in disaster recovery efforts, providing real-time data for emergency responders. Smart cities will monitor air and water quality through millions of sensors, giving them insights needed to provide a better quality of life.

References

- [1] Erik Seligman and Itai Yarom, 'Best Known methods of using Cadence Conformal LEC at Intel' , Intel Corporation
- [2] S Tamil Selvi and P sukumar 'Clock Power Reduction using Multi-Bit FlipFlop Technique'
- [3] Cadence Design systems, Encounter Conformal ECO User guide, product version 15.2.
- [4] Cadence Design systems, Encounter Conformal Equivalence Checking Reference guide, product version 15.2.
- [5] Cadence Design systems, Encounter Conformal Equivalence Checking Low power User Guide, product version 15.2.