

Development of Verification Environment for Behavioural Models of Analog Mixed Signal IP PLL

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(VLSI Design)

By

Sahil Narang

(13MECV31)



Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2015

Development of Verification Environment for Behavioural Models of Analog Mixed Signal IP PLL

Major Project Report

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

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

By

**Sahil Narang
(13MECV31)**

Under the guidance of

External Project Guide:

Mr. Hardik Parekh

Technical Leader

ST Microelectronics Pvt. Ltd.,

Greater Noida.

Internal Project Guide:

Prof. Akash Mecwan

Professor (EC Dept.),

Institute of Technology,

Nirma University, Ahmedabad.



**Electronics & Communication Engineering Branch
Electrical Engineering Department**

Institute of Technology

Nirma University

Ahmedabad-382 481

May 2015

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.

- **Sahil Narang**



Certificate

This is to certify that the Major Project entitled “**Development of Verification Environment for Behavioural Models of Analog Mixed Signal IP PLL**” submitted by **Sahil Narang (13MECV31)** 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.

Prof. Akash Mecwan

Internal Guide

Dr. N. M. Devashrayee

PG Coordinator (VLSI Design)

Dr. P.N.Tekwani

Head of EE Dept.

Dr. Ketan Kotecha

Director, IT-NU

Date:

Place: Ahmedabad

Acknowledgements

I would have never succeeded in completing my thesis without the cooperation, encouragement and help provided to me by various people. Firstly, my sincere thanks to the Design, Automation and Integrity System team during this training. Their wisdom, clarity of thoughts and support motivated me to bring this project to its present state.

I wish to thank **Deepshikha Singh (Manager)** and **Hardik Parekh (Mentor)** for giving me an opportunity to work with them. I wish to place on record my gratitude to **ST Microelectronics Pvt. Ltd., Greater Noida**, for providing me an opportunity to work with them. My stay in the organization has been a great learning experience and a curtain raiser to an interesting and rewarding career.

I would like to express my sincere gratitude to **Dr. Ketan Kotecha (Director, Nirma University, Ahmedabad)** for his continuous guidance, support and enthusiasm. I would take this opportunity to thank **Dr. P.N. Tekwani (Head of Department, Electrical Engineering)**, **Dr. N. M. Devashrayee (Professor and Program Coordinator, M.Tech - EC (VLSI Design))**, **Internal Guide Prof. Akash Mecwan** and all the faculties at Nirma University (VLSI Design), for their vision and relentless effort, support, and encouragement to provide me with this excellent opportunity to carry out my project work in such a highly renowned and esteemed organization, ST microelectronics Pvt. Ltd. I am equally thankful to ST microelectronics Pvt. Ltd. for providing me the invaluable exposure to the industry and the current market trends. Finally, I would like to thank my family for their interest and never-ending support during my studies.

- Sahil Narang
13MECV31

Abstract

To accomplish the design chip of an integrated circuit, the first step is to know the design specification. The next step is to make a behavioral model of that design and then this behavioral design is converted to a RTL description with the help of synthesis tools. This is the basic building block of any RTL to GDS flow. There are some designs which are not synthesizable. These designs are basically analog models. Therefore there is no RTL description written for these designs. Transistor level netlist is provided for the complete verification of any SoC. The accuracy of the verification from SPICE tools is very high but with that the time requirement is also huge. Therefore only behavioral models are made for analog designs so that the verification for SoC can be done with a less amount of time. When the behavioral description is written in a HDL language then the equivalence checking is done between behavioral description and transistor level netlist.

Phase Locked Loops (PLLs) are electronic circuits used for frequency control. For every SoC, basic clock generation circuit is required. PLL is an analog IP which is not synthesizable. Modeling and verifying analog designs in behavioral way such that the time requirements for making the design and also to verify that can be reduced. There are different types of PLL designed but the basic functionality is same for all. Making of a generic test bench for all the types of PLL can reduce a huge time also with all the tests applied to the PLL. This technique can be used effectively by the help of scripts.

Contents

Declaration	iii
Certificate	iv
Acknowledgements	v
Abstract	vi
List of Figures	ix
1 Introduction	1
1.1 Behavioural Model of PLL	3
1.1.1 Input Divider Bits	4
1.1.2 Phase/Frequency Detector	5
1.1.3 Charge Pump and Loop Filter	5
1.1.4 Voltage Controlled Oscillator	5
1.1.5 Loop Frequency Divider	5
1.1.6 Output Frequency Divider	5
1.1.7 Lock Detector	6
2 Modes of PLL	7
2.1 Integer PLL	7
2.2 Fractional PLL	8
2.3 Spread Spectrum Clock Generation PLL	8
2.4 Clock Insertion PLL	10
2.5 Free Running PLL	11
2.6 Bypass Mode PLL	12
3 Generic Test Bench for Behavioural Models of PLL	13
3.1 Frequency Verification Tests	14
3.1.1 Random Frequency Verification	14
3.1.2 Minimum Frequency Verification	14
3.1.3 Maximum Frequency Verification	15
3.1.4 Select Bits Frequency Verification	15

3.2	Regression Tests	15
3.2.1	OKIN Test Cases	16
3.2.2	Power Supply Test Cases	16
3.2.3	Power Down Test Cases	17
3.2.4	Input Clock Test Cases	18
3.2.5	Input Divider Test Cases	20
3.2.6	Loop Divider Test Cases	20
3.2.7	Output Divider Test Cases	22
3.2.8	Fractional Mode Test Cases	24
3.2.9	SSCG Mode Test Cases	24
3.2.10	Clock Insertion Test Cases	24
3.2.11	Free Running Test Cases	24
3.3	Verification Checkers	25
3.3.1	Frequency Check	25
3.3.2	Duty Cycle Check	25
3.3.3	Status Check	25
3.3.4	Stuck Check	26
3.3.5	Lock Protocol Check	26
3.4	Final Report	27
4	Verification Environment for PLL	28
4.1	Compiler Configuration File/ User Input Configuration File	29
4.2	Generation of Configuration File for User Inputs	29
4.3	Generation of Test Bench Configuration File + Test Selection Database	29
4.4	Generic Driver and Checker	30
4.5	Regression Script	30
4.6	Algorithm for Generic Verification for all types of PLL	30
5	Result	32
6	Conclusion and Future Work	33
6.1	Conclusion	33
6.2	Future Work	33
	References	34

List of Figures

1.1	Basic operation of PLL	2
1.2	Behavioural Model of PLL	4
2.1	Integer Mode PLL	7
2.2	Fractional Mode PLL	8
2.3	SSCG Mode PLL	9
2.4	Spread frequency over a wide range	9
2.5	SSCG PLL Output in the centre and Down Spread Modes	10
2.6	Clock Insertion PLL Application	11
2.7	Clock insertion PLL by Design	11
2.8	Free Running PLL	12
3.1	Power pin test case applied to PLL	17
3.2	Power down test case is applied to PLL	18
3.3	INFF test case applied to PLL	19
3.4	IDF test cases applied to PLL	21
3.5	NDIV test case applied to PLL	22
3.6	ODF test case applied to PLL	23
3.7	Lock protocol for dual lock	26
3.8	Final Report	27
4.1	Flow diagram of verification for PLL	28
4.2	Overview of Generic Test Bench	31

Chapter 1

Introduction

Timing signals are essential for the proper operation of digital circuits and communication networks to synchronize the operations of circuits. These signals synchronize the flow of digital signals among various synchronous circuits. These timing signals manage the data signals which sent out to govern interconnected digital blocks.

A simple approach for the generation of such a signal is to have an oscillator, which originate the clock. Now a days, complex systems require a diverse clock frequencies with high accuracy and less disturbance. A designer may have to put as many oscillators as the different number of frequencies required on the system. This process takes most of the design area, increases complexity of the design and design cost.

Phase locked loops are widely used to generate on-chip clocks in high-frequency digital systems. This clock generators are incorporated into almost every large-scale analog and mixed-signal system-on-chip (SoC). A PLL based clock distribution system can take any clock source as input and generate multiple numbers of clock outputs with a frequency that is lower or greater compared to the input clock.

This clock distribution system has an additional functionality that all the outputs clock have a fixed phase relationship to each other. This functionality is required in the systems where it is mandatory to deliver varying frequency signals to different digital blocks while keeping their operation in synchronization.[1]

There are several common applications of PLL:

- Noise and jitter reduction
- Zero delay buffer
- Clock de-skew application
- Frequency synthesis
- Spread spectrum clock
- Clock and data recovery

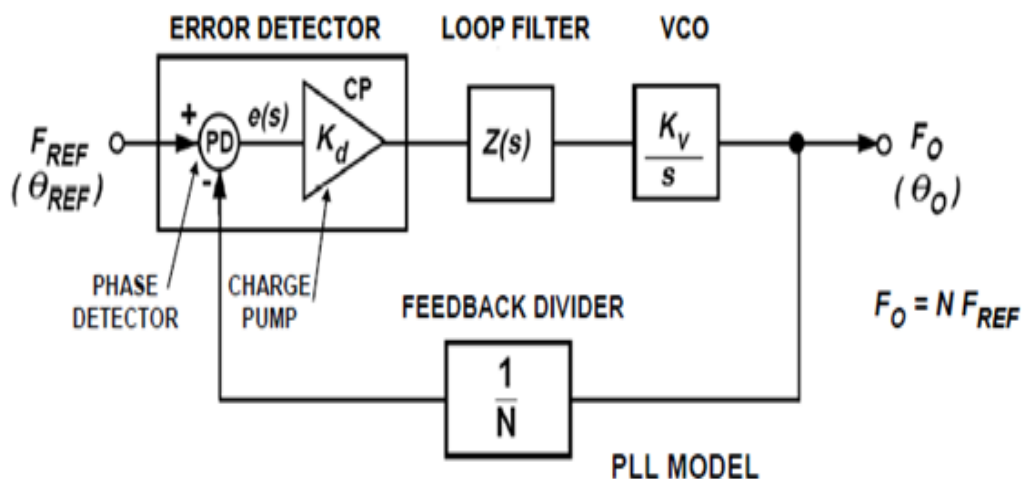


Figure 1.1: Basic operation of PLL

[2]

PLL incorporates a phase/frequency detector, a charge pump, a low-pass filter and a voltage-controlled oscillator. An input reference frequency is sent to one of the phase/frequency detector inputs. The other input of phase/frequency detector is driven by a feedback divided version of voltage controlled oscillator output signal to provide a negative feedback. The phase/frequency detector detects differences in phase and frequency between two inputs i.e. reference and feedback inputs to generate compensating up or down signals.

If reference input occurs before the feedback input, indicating that the VCO is running too slowly, the PFD produces an up signal that lasts until the rising edge of the feedback input. If the feedback input occurs before reference input, the PFD produces a down signal that is triggered on the rising edge of the feedback input and lasts until the rising edge of reference input. If the feedback frequency is less than that of reference, the pulse-width of the up signal is greater than the width of down signal and vice versa. In this way, the PFD produces a control signal which is unique for any phase and frequency relationship between reference and feedback signal.

This control signal is then passes through charge pump and loop filter. The output of the loop filter generates a control voltage for voltage controlled oscillator. This voltage is used to control the frequency of the voltage controlled oscillator. The frequency of voltage controlled oscillator can be forced to run faster and slower according to the input controlled voltage. As a result voltage controlled oscillator frequency locks to oscillate a fixed frequency when two of its inputs at phase/frequency detector are phase and frequency aligned. The output of the voltage controlled oscillator is an internally generated oscillator waveform.

The PLL is designed to operate within a limited band of input frequencies. If reference frequency is outside the from a predefined range then there will be no lock and frequency of voltage controlled oscillator will be different from the expected one.

Two main parameters of a phase locked loop system are... Capture Range: The time taken by the PLL to capture the lock is called capture range. It is smaller than the lock range. Capture range is mainly related to the bandwidth of low-pass filter. Lock Range: The range in which PLL can withstand its lock functionality with its reference frequency. It is also known as the holding range.

1.1 Behavioural Model of PLL

The functional behavior of a PLL is to generate a frequency, which is multiple of input frequency with same phase. Input frequency is divided through IDF and that

is $INFIN$ compared with the $FBCLK$ for generating a Lock Signal. This lock signal will go high when input frequency $INFIN$ and output frequency $FVCO$ will have same frequency and zero or constant phase. $FVCO$ output frequency is a multiplied version of $INFIN$ input frequency where multiplication factor is LDF .

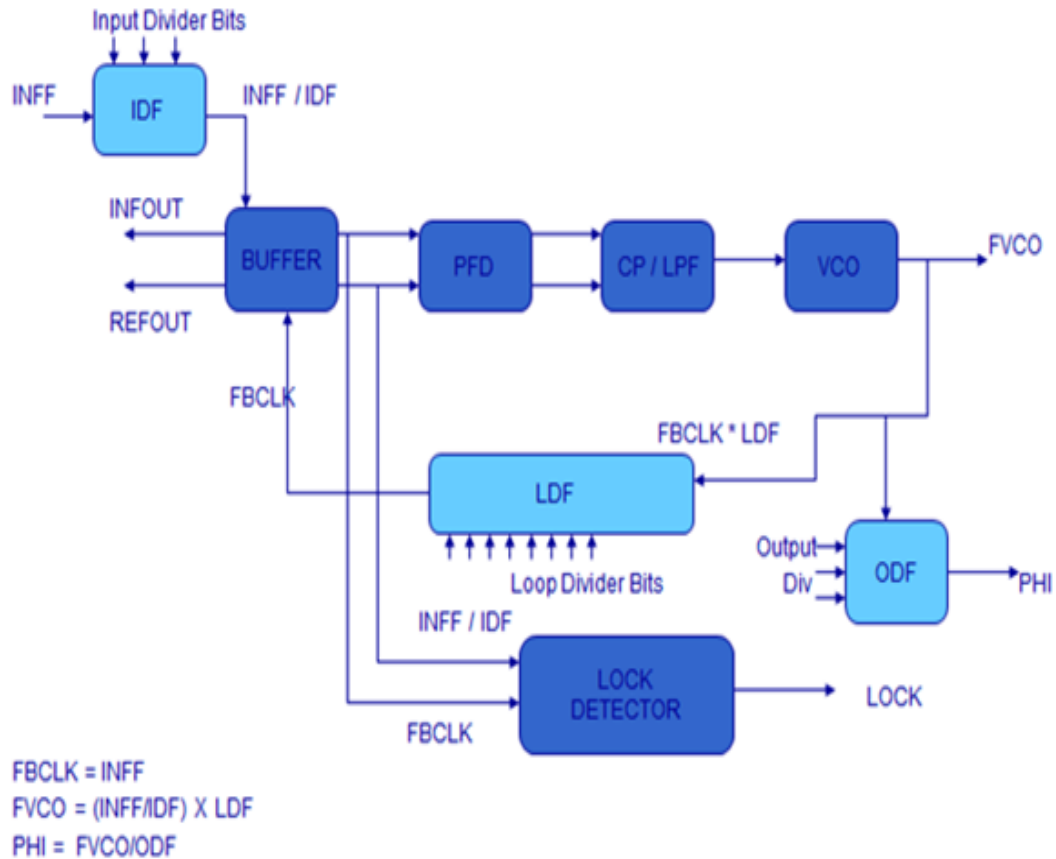


Figure 1.2: Behavioural Model of PLL

1.1.1 Input Divider Bits

Input frequency divider is present within the PLL for dividing the input frequency by a factor called the Input Division Factor (IDF). The output of this block is input of phase/frequency detector.

$$INFIN \text{ or } INFOUT = INFF / IDF$$

1.1.2 Phase/Frequency Detector

This block compares the phase difference between the corresponding rising edges of Reference frequency and feedback clock by generating voltage pulses with widths proportional to the input phase error.

1.1.3 Charge Pump and Loop Filter

This block converts the voltage pulses from the Phase/Frequency Detector to current pulses, which charge the Loop Filter and generate the Control Voltage for the Voltage Controlled Oscillator.

1.1.4 Voltage Controlled Oscillator

This is the oscillator inside the PLL, which produces an output frequency proportional to the Input Control Voltage.

1.1.5 Loop Frequency Divider

Frequency Divider is present within the PLL for dividing the VCO frequency by a factor called the Loop Division Factor (LDF). The output of this block is the feedback clock.

1.1.6 Output Frequency Divider

Output Frequency Divider is present within the PLL for dividing the VCO frequency by a factor called the Output Division Factor (ODF). The output of this block is the output clock PHI. This is the primary output of PLL.

$$\text{PHI} = \text{FVCO}/\text{ODF}$$

1.1.7 Lock Detector

Input frequency $INFIN$ and feedback frequency $FBCLK$ is compared and if they both are in phase and frequency is same then this signal generates a signal high that is $LOCKP$.

Chapter 2

Modes of PLL

There are total 7 modes of PLL as described below

2.1 Integer PLL

This type of PLL generates the output frequency which is an integer multiplication of input frequency. The input and output frequency of the PLL is same and the phase difference between input and output frequency is same or constant.

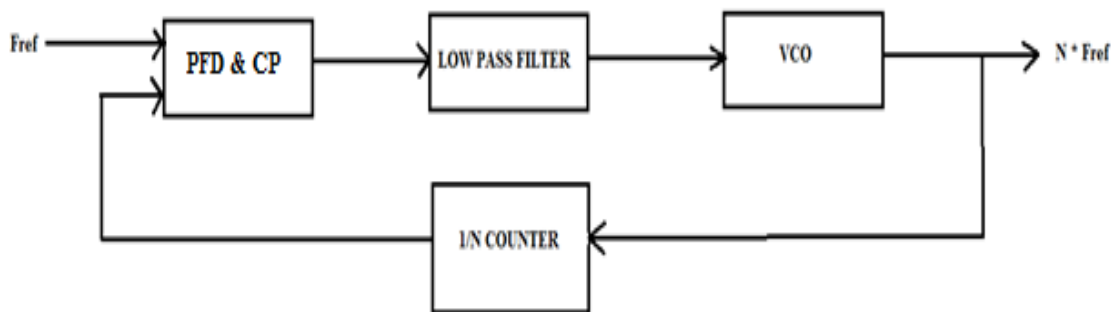


Figure 2.1: Integer Mode PLL

2.2 Fractional PLL

In this type of PLL the output frequency can be a fractional multiplication of input frequency. The LDF bits changes its division to the FVCO output frequency such that the average output frequency is the fractional multiplication of input frequency.[4]

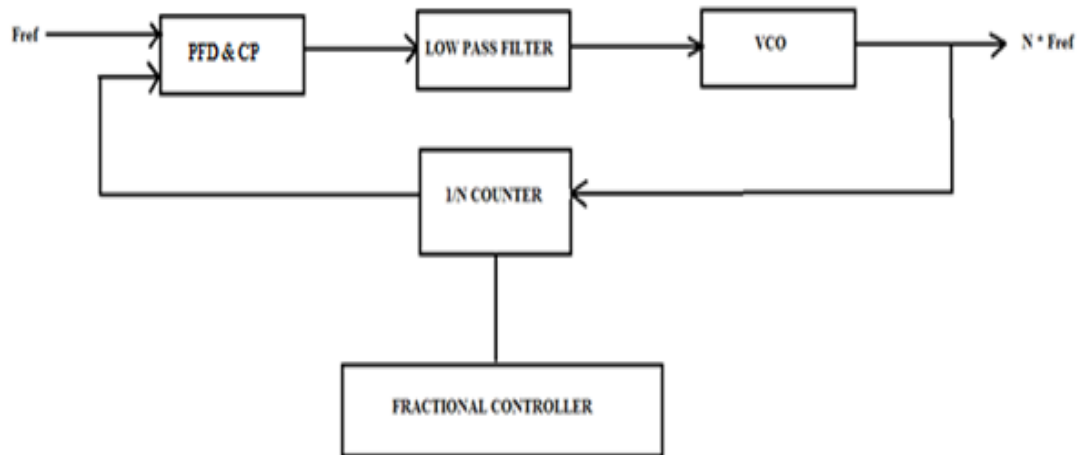


Figure 2.2: Fractional Mode PLL

2.3 Spread Spectrum Clock Generation PLL

Spread spectrum clock generator (SSCG) PLL is used to reduce electro-magnetic induction caused by the high frequency output of the PLL. This electro-magnetic radiation can be reduced by modulating the output clock frequency. Thereby spreading the radiative energy over a wide frequency range wider than the bandwidth of the receiver used to measure emissions. The technique of spreading energy is done by modulating the feedback loop division factor at a very fast speed.

The fundamental clock oscillator signal and often its harmonics produce peak radiation that exceeds regulations, but spread spectrum clocking spreads the overall energy in many cases below the maximum.

There are two profiles of SSCG mode. First one is center spread, in which the frequency is modulated above and lowers over the output frequency in a triangular manner and second one is down spread, in which the frequency is modulated lower over the output frequency in a triangular manner as shown in figure 2.5. One triangular period is called modulation period and the depth at which frequency can modulate is called modulation depth.

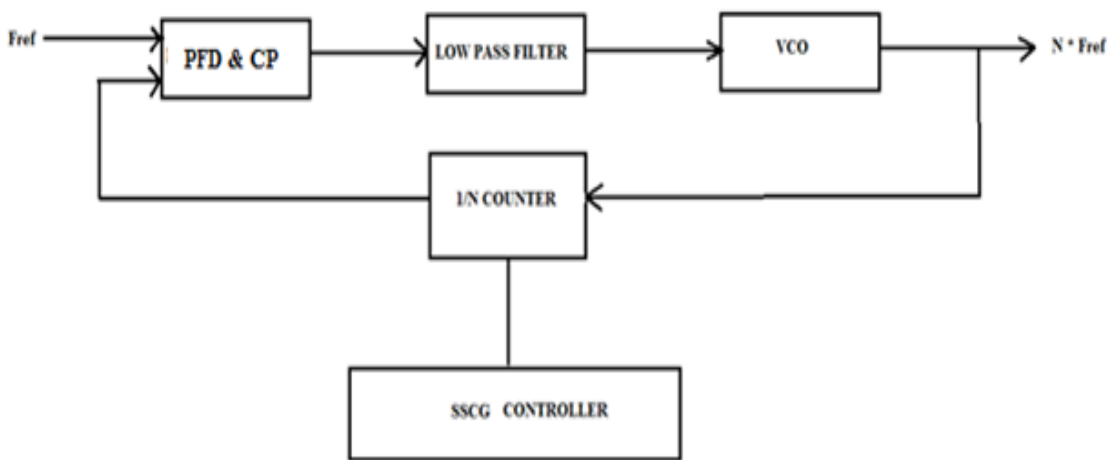


Figure 2.3: SSCG Mode PLL

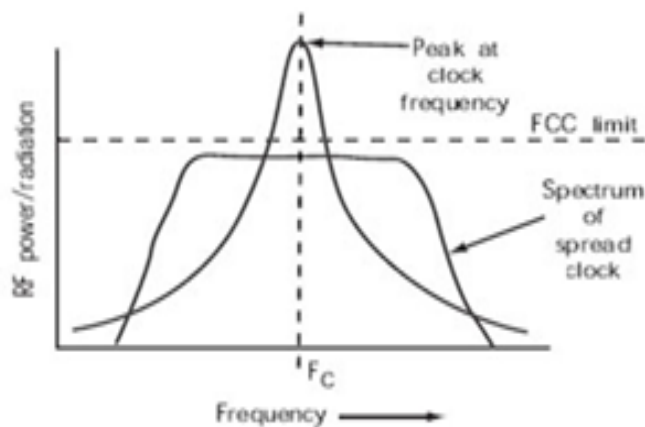


Figure 2.4: Spread frequency over a wide range

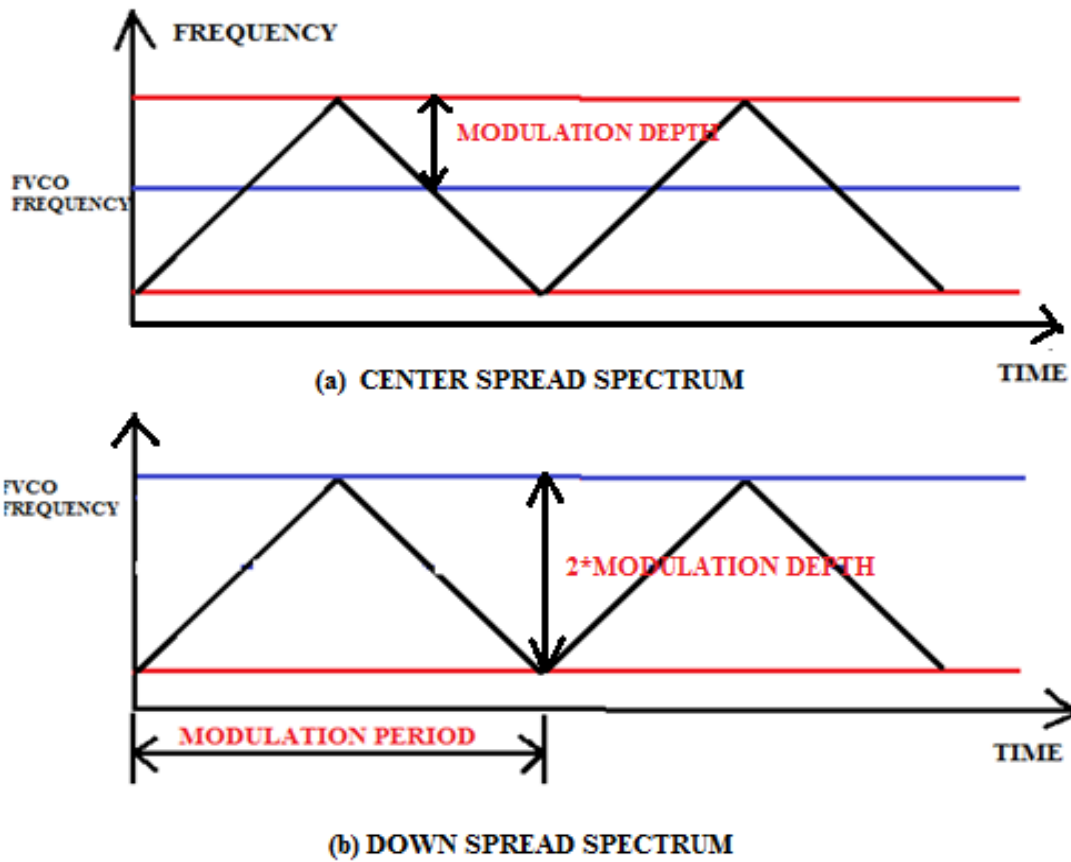


Figure 2.5: SSCG PLL Output in the centre and Down Spread Modes

2.4 Clock Insertion PLL

Reference clock enters the chip and drives a phase locked loop, which then drives the system's clock distribution. The clock insertion is usually balanced so that the clock arrives at every endpoint simultaneously. One of those endpoints is the PLLs feedback input. The function of the PLL is to detect the comparison between distributed clock and incoming reference clock and vary the phase and frequency of its output until the reference and feedback clocks are phase and frequency matched.

In the behaviour mode of clock insertion PLL, an external divider is added which

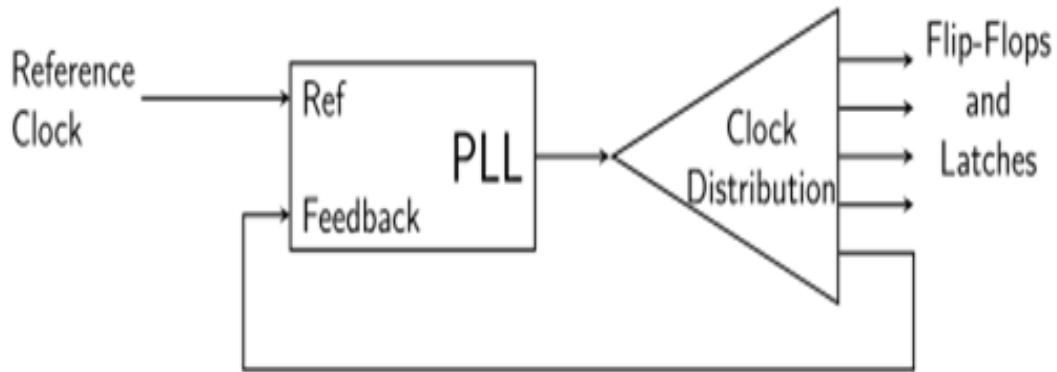


Figure 2.6: Clock Insertion PLL Application
[3]

can be optional as per the requirement. This external divider input is the output of PLL clock i.e. PHI and the output is delayed or divided version of PHI, which is given as input feedback clock to the phase frequency detector.

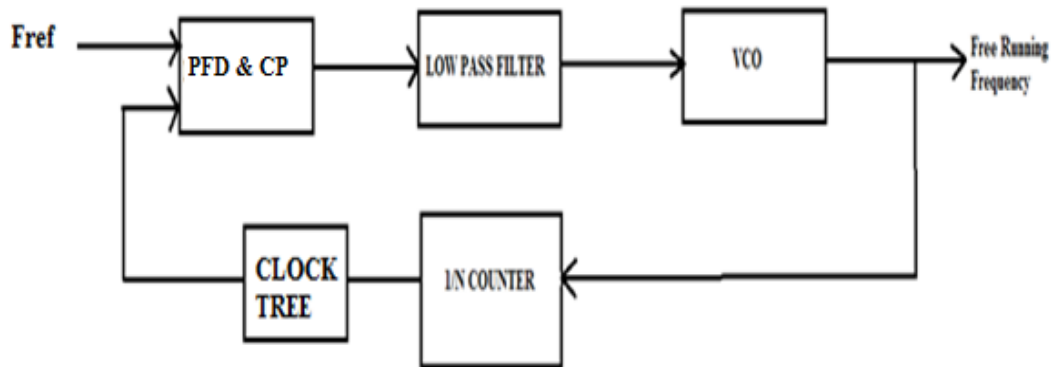


Figure 2.7: Clock insertion PLL by Design

2.5 Free Running PLL

In this type of PLL the input frequency is stuck to 0/1, then the input of control voltage of VCO becomes zero and VCO generates its own free running frequency.

PLL can go out of free running mode as the input frequency starts toggling at the input.

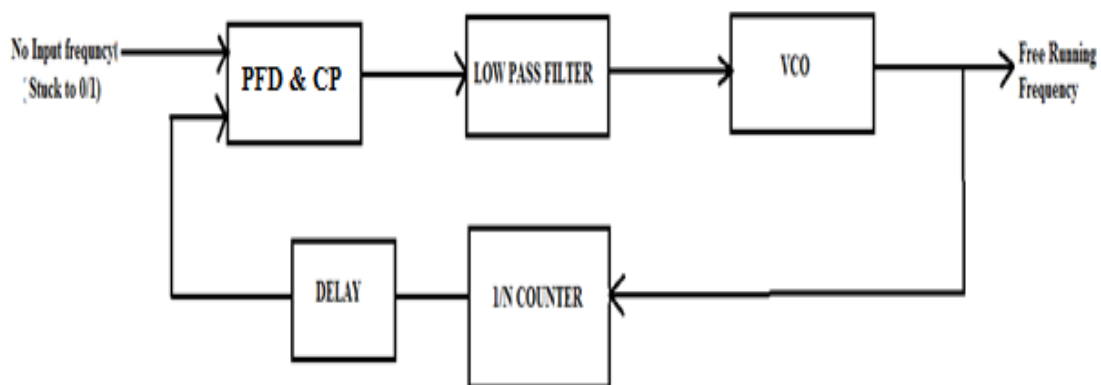


Figure 2.8: Free Running PLL

2.6 Bypass Mode PLL

In this mode, the input frequency is bypassed to the output as there is no role of PFD, LP and VCO. This technique is used when there is no requirement of PLL functionality and the input frequency is bypassed to output.

Chapter 3

Generic Test Bench for Behavioural Models of PLL

There are numbers of steps carried out for development of verification of an IP such as Phase Locked Loop. The number of tests required for the verification is large and also not applicable for all the PLL IP's. Every model has its own functionality which is not supported by others. So there is a need to write a test bench which supports all types of functionality and can be used wherever needed. This chapter includes all the test cases required for the verification. There are numbers of test cases described for testing the PLL with different types of modes used simultaneously. Total time required for testing a PLL depends on the functionality supported with that. If PLL has all modes supported then the total time required for testing is only 2 to 3 hours. Some of the basic tests done on a PLL which is only integer mode functionality is described below. Total time for testing this integer PLL was 45 minutes.

The verification of a model is done in three parts which includes...

- Frequency Verification Tests
- Regression Tests
- User Defined Tests

3.1 Frequency Verification Tests

In this mode as many as possible frequencies are given as an input with its supported divider configuration. The primary inputs of this verification are minimum and maximum input frequency of the PLL. This frequency verification is further classified into four parts...

- Random Frequency Verification
- Minimum Frequency Verification
- Maximum Frequency Verification
- Select Bits Frequency Verification

3.1.1 Random Frequency Verification

In this part there is a random frequency is taken from the minimum and maximum frequency. This random frequency is driven to the PLL with their supported divider values and expected frequency is compared with the model output frequency. If there is any discrepancy in the model frequency and the expected frequency then the verification environment throw an error by displaying a message of frequency error at a particular time. The number for random frequency verification can be changed by the user.

3.1.2 Minimum Frequency Verification

In this part the minimum frequency is taken as an input to the test bench and as many as possible divider values are driven one by one to the model. After every one test the model frequency is compared with the expected frequency. This frequency verification checks all the supported values of the divider configurations with a particular minimum frequency.

3.1.3 Maximum Frequency Verification

In this part maximum frequency is taken as an input to the test bench same as minimum frequency verification and all possible divider configuration is calculated from the test bench which are driven by the test bench to the model one by one to check the frequency.

3.1.4 Select Bits Frequency Verification

Some of the PLL supports different type of divider configuration. In this type of configuration numbers of input selected for the fix values of divider configuration.

3.2 Regression Tests

In this mode only one particular frequency is driven to the dut and supporting their input, feedback and output divider configurations are also driven to the dut. Now using one input at a time, all the input test cases driven one by one and dut result is compared with the expected result. This regression is further classified into these parts...

- OKIN Test Cases
- Power Supply Test Cases
- Power Down Test Cases
- Input Clock Test Cases
- Input divider Test Cases
- Loop Divider Test Cases
- Output Divider Test Cases
- Fractional Mode Test Cases

- SSCG Mode Test Cases
- Test Mode Test Cases
- Clock Insertion Test Cases
- Free Running Test Cases
- Bypass Mode Test Cases
- Toggling Test Cases

3.2.1 OKIN Test Cases

This pin is required for the proper starting functionality of the PLL. When power supply is stabled then only after some time period this OKIN signal goes to high. According to the functionality of this pin there are numbers of test cases written to verify the maximum functionality of the PLL with respect to the OKIN pin.

3.2.2 Power Supply Test Cases

For any IP , power supply stability is must. When PLL is working into the normal mode then if power supply is toggled model should corrupt the outputs. This functionality is checked by the verification environment.

- VDD and GND = L : Supply pins are low at the start of simulation
- VDD = X or Z OR GND = X or Z : Invalid inputs at supply pins at the start of simulation
- VDD = H/L to X or Z, GND = L : VDD changes state from H/L to X or Z
- VDD = H/L, GND = L to X or Z: GND changes state from L to X or Z

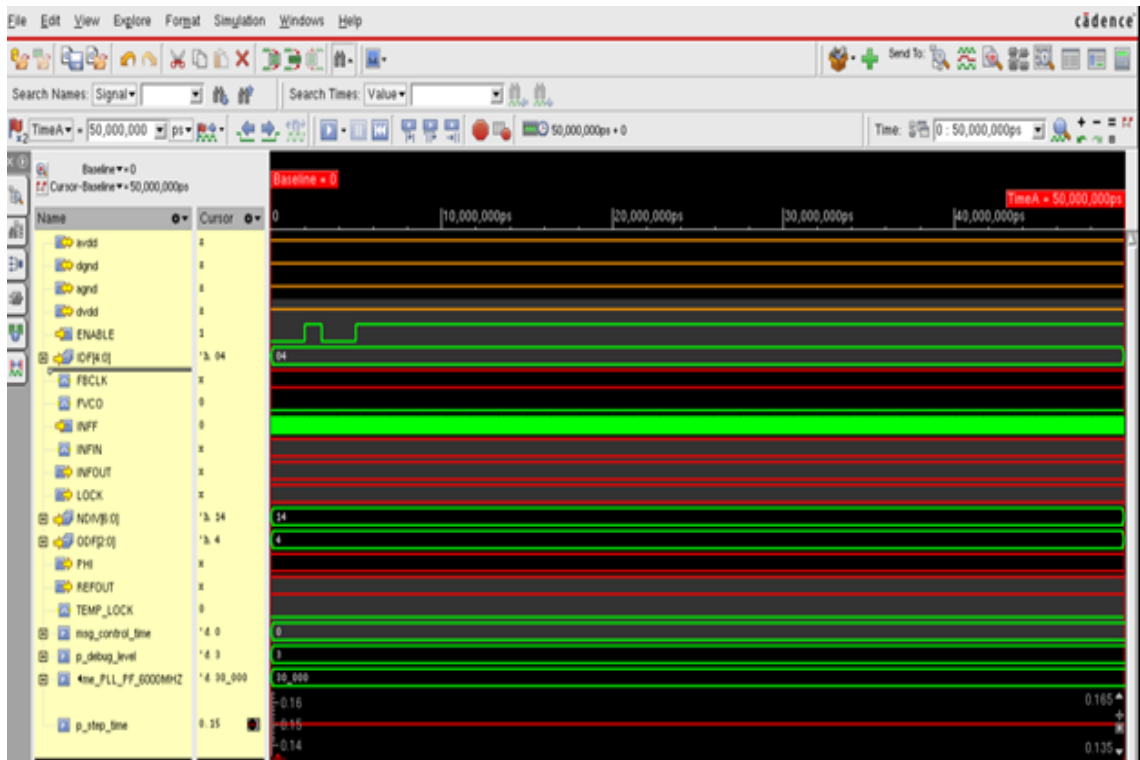


Figure 3.1: Power pin test case applied to PLL

3.2.3 Power Down Test Cases

This pin is required to disable the PLL functionality. When PLL is in power down mode, all the outputs of the PLL must go into power down that is the outputs will be low. For every test case starting minimum time is required by the PLL in which the controlled voltage of the voltage controlled goes to zero. Now if the power down width is less then the minimum width of power down then outputs of PLL must be corrupted. Any change in the inputs of PLL without having any supply does not change the outputs. As Enable goes from high to low and then again low to high does not affect the outputs because there is no proper supply is given to PLL. As a result this test case is passed.

- ENABLE = L/H/X : At the beginning of simulation
- ENABLE = L/H to X: To check the power down mode.

- ENABLE = H to L: Enable becomes high from low.
- ENABLE = X to H/L: Enable becomes high/low from X.
- ENABLE = X to L: Enable becomes low from X.

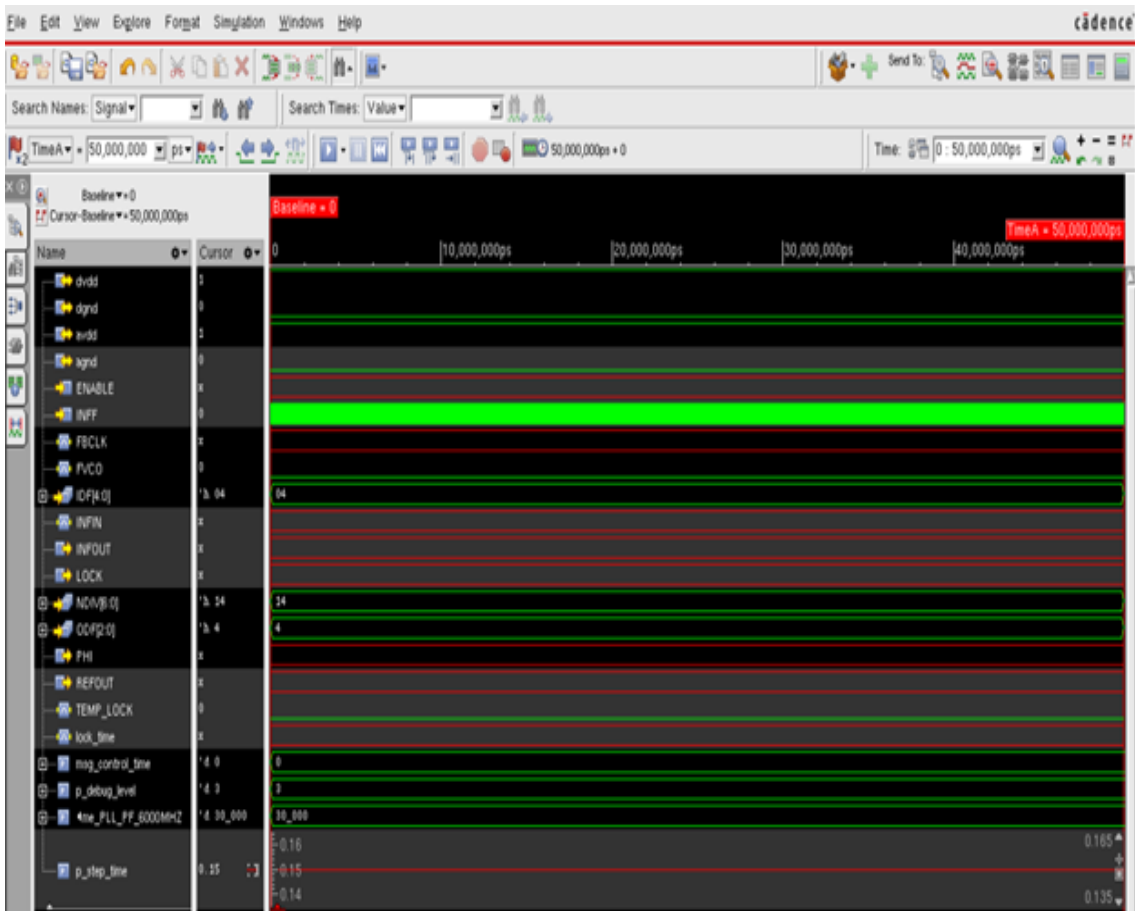


Figure 3.2: Power down test case is applied to PLL

When PLL has a proper supply but the enable signal is corrupted then outputs must be corrupted and can be recovered only when enable signal goes low with proper supply.

3.2.4 Input Clock Test Cases

These test cases are driven for the verification of input clock cases. For the normal functionality of the PLL, input clock must have a frequency which ranges between

CHAPTER 3. GENERIC TEST BENCH FOR BEHAVIOURAL MODELS OF PLL19

minimum and maximum frequency range. Also with that there should not any change in the frequency of the PLL when PLL is in lock state.

- INFF = X: Invalid input at INFF in power down, capture and normal Mode.
- INFF frequency changes in power down, capture and normal mode.
- INFF frequency is in between 3.96 MHz and 353.3 MHz
- Jitter is greater than specified limits.
- Duty Cycle out of range.
- INFF is stuck at 0/1.

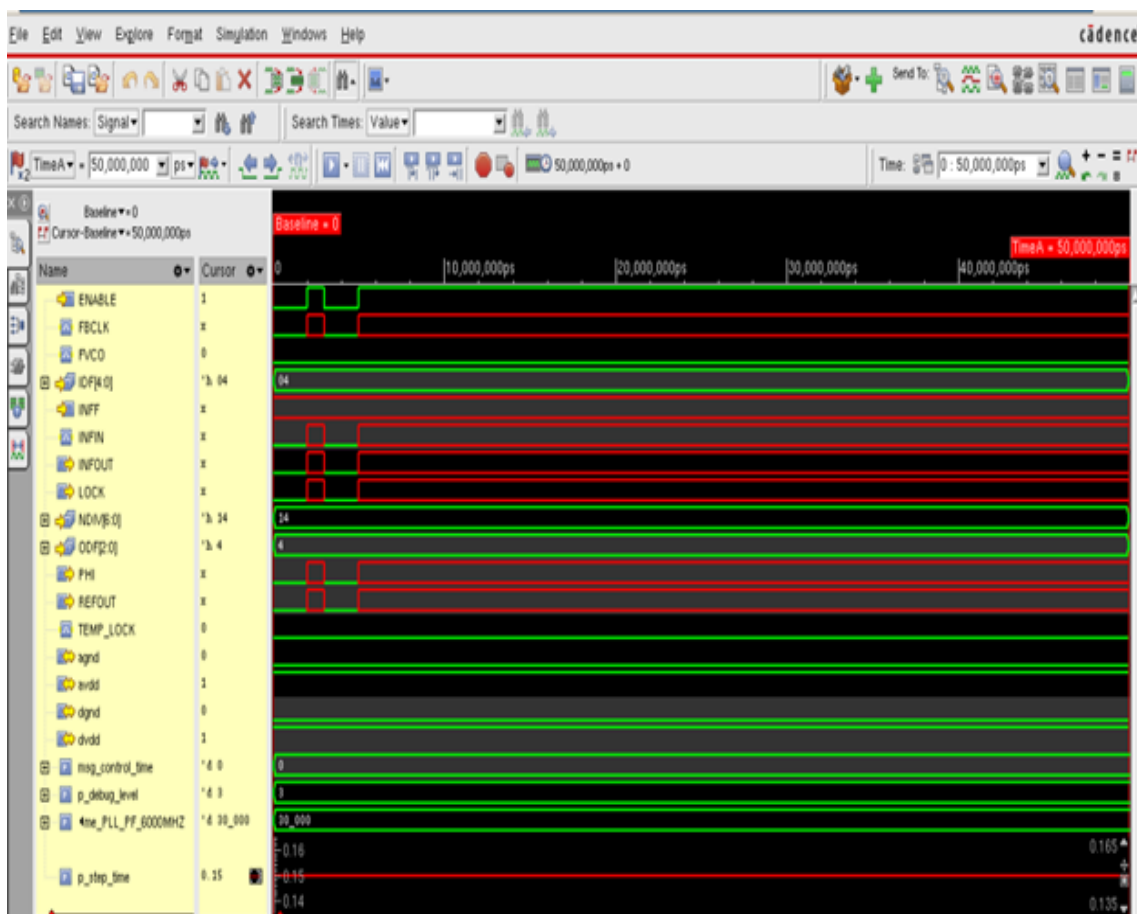


Figure 3.3: INFF test case applied to PLL

When proper supply is given to PLL and enable signal goes from low to high then the input clock should not be corrupted. As shown in figure 3.3, when enable goes from low to high then input signal that is INFF is corrupted, as a result all the outputs are corrupted. This can be recovered only with proper down pulse.

3.2.5 Input Divider Test Cases

The input divider is required for generating the input for the phase/frequency detector. The input divider value is taken such that the frequency of INFIN, which is the input of phase/frequency detector, must be in range. When PLL is in lock state or in normal mode, input divider bits must not toggled.

- IDF = X: Invalid input at IDF.
- IDF = X to valid input: changes from invalid to valid input when ENABLE = H.
- IDF min value and max value
- INFF is in range but IDF is such that INFIN is out of range

When PLL goes out from power down mode with proper supply and input frequency is correct then input divider bits divides the input frequency INFF and gives output INFIN. If IDF is corrupted then all the outputs should be corrupted as shown in figure 3.4. This can be recovered when IDF bits changes from invalid states to valid states in power down mode.

3.2.6 Loop Divider Test Cases

The loop divider is used for generating the output frequency which is multiplication of input frequency. The loop divider value is taken such that the frequency of FVCO, which is the output of voltage controlled oscillator, must be in range. When PLL is in lock state or in normal mode, loop divider bits must not toggled.

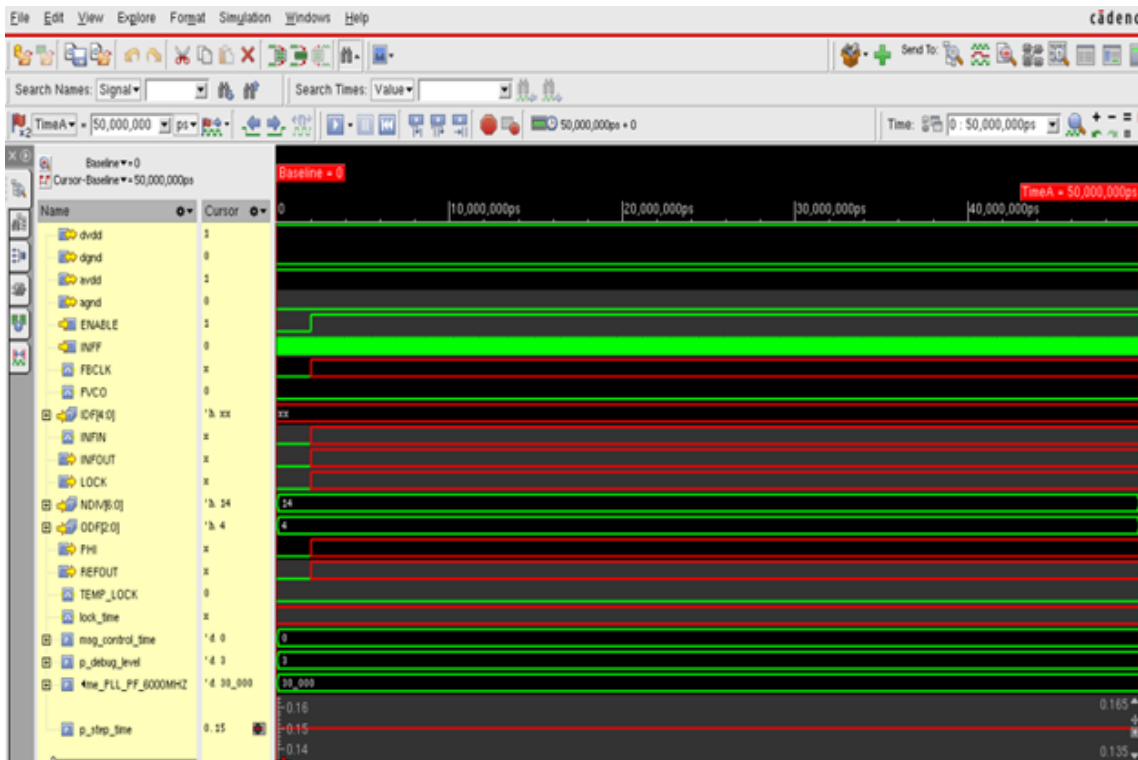


Figure 3.4: IDF test cases applied to PLL

- $\text{NDIV} = \text{X}$: Invalid input at NDIV
- NDIV changes in power down, normal and capture mode
- $\text{NDIV} = \text{X}$ to L/H : changes from invalid to valid input when $\text{ENABLE} = \text{H}$.
- $\text{NDIV} = \text{L/H}$ to X: changes from invalid to valid input when $\text{ENABLE} = \text{H}$.
- INFIN is in range but NDIV is such that FVCO is out of range

When PLL goes out from power down mode with proper supply and input frequency is correct then input divider bits divides the input frequency INFF and gives output INFIN. If loop divider bits NDIV are changed in after the lock, when frequency and phase are same, then all the outputs should be corrupted except INFOUT as shown in figure 3.5. This can be recovered when NDIV bits changes from invalid states to valid states in power down mode.

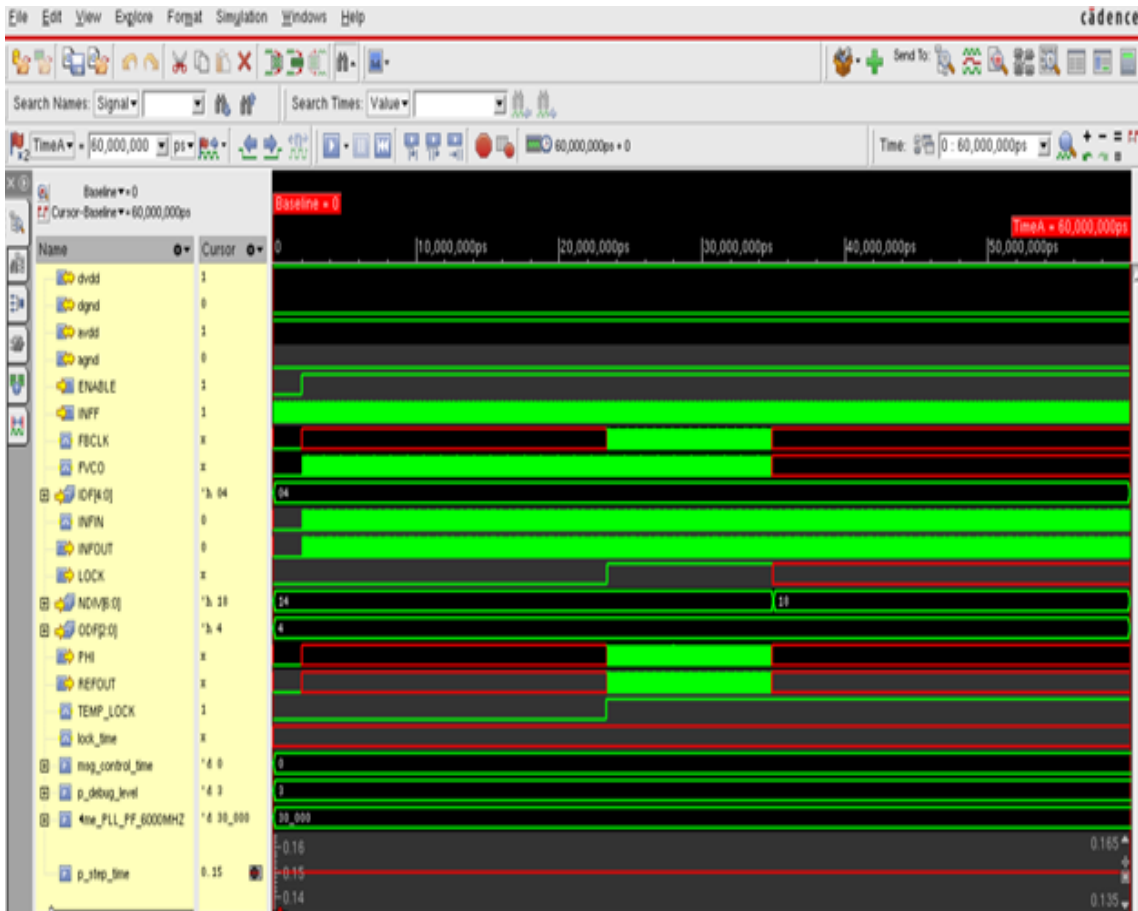


Figure 3.5: NDIV test case applied to PLL

3.2.7 Output Divider Test Cases

The output divider is used for generating the range of output phi frequency which is division of input fvco frequency. When PLL is in lock state or in normal mode, loop divider bits can be toggled.

- $ODF = X$: Invalid input at NDIV
- ODF changes in power down, normal and capture mode
- $ODF = X$ to L/H : changes from invalid to valid input when $ENABLE = H$.
- $ODF =$ valid input to X : changes from invalid to valid input when $ENABLE = H$ if Settling time is given then after that PHI should be $FVCO/ODF$ else dont

wait FVCO/ODF.

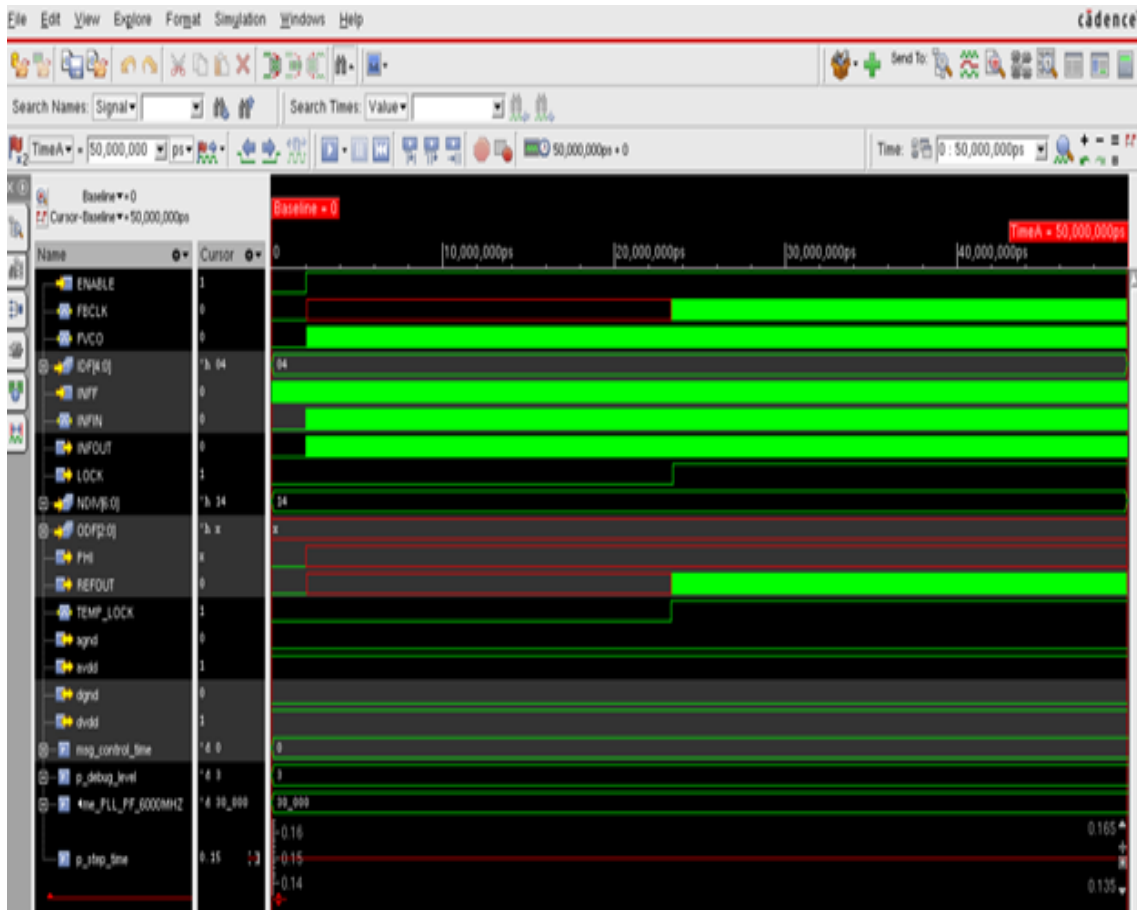


Figure 3.6: ODF test case applied to PLL

PLL is in normal mode where phase and frequency of the output are same as the input phase and frequency. Now if output divider bits corrupts then only PHI output should be corrupted because there is no role of ODF bits in the feedback loop. As long as ODF bits are corrupted, PHI should be corrupted and if ODF changed to a valid state then PHI output will start to give an output frequency after some settling time.

3.2.8 Fractional Mode Test Cases

This mode is generally supported for generating the large no of frequency ranges. When this mode is on then the loop divider bits toggles after a fix numbers of cycles such that it does not change the lock state of PLL. This test case is driven to check that proper toggling of loop divider bits is performing or not. The output of pll is then compared with the expected behaviour of pll.

3.2.9 SSCG Mode Test Cases

When this SSCG mode is active then the output frequency changes according to the control bits of sscg mode. test bench driver drives the sscg control bits to dut and verify that spreading is proper or not. When PLL is in normal mode then no input is allowed to toggling, if any of the input is toggled, outuputs should be corrupted.

3.2.10 Clock Insertion Test Cases

Clock insertion mode is not supported by all the PLL's. Verification of this mode is a complex task because generation of an external divider is dependent on user that how it is implemented. When this mode is active, there is a delay in the input frequency and output frequency. If this delay is out of range then PLL should corrupt all of its outputs. This mode is used for reduction in clock tree sythesis at the later part.

3.2.11 Free Running Test Cases

When the input clock is stuck to 0/1, then the controlled voltage of voltage controlled oscillator should go down to 0 as per the design but there will be a constant voltage of voltage controlled oscillator which generates a random output frequency. Test bench drives an input clock which is stuck to 0/1 in normal mode, then output frequency is compared with the expected behaviour of the PLL.

3.3 Verification Checkers

For every IP verification, checkers are the most important parts. In this test bench total 5 checkers are used which checks the outputs continuously.

- Frequency Check
- Duty Cycle Check
- Status check
- Stuck Check
- Lock Protocol Check

3.3.1 Frequency Check

At every edge of the output clock for everyone INFOUT, REFOUT, FVCO, PHI, it calculates the time period and check the period is in range. This frequency is changed according to the running mode that can be Integer, Fractional, SSCG, Clock insertion or Free Running.

3.3.2 Duty Cycle Check

It will also checks the duty cycle at the very edge of the clock. Duty cycle can be changed according to the output divider bits.

3.3.3 Status Check

At every 100 ps it will check the output status that 0, 1 or X. when driver drive the values such that expected should be corrupted it will toggle a flag and according to that the status is checked. Expected status is changed through a flag.

3.3.4 Stuck Check

At every posedge and negedge of the output clock is checked that the output is stuck or not. It will wait the next edge of clock and also wait for the time period of clock, if the edge is not available till the time period of clock it will give an error that output is stuck at this particular time.

3.3.5 Lock Protocol Check

It will check the continuous checking of the status of the lock when lock is high. after maximum lock time lock is H or not if not test bench will through an error. There can be maximum three lock in a PLL. This lock protocol checks the status of each lock at every 10 ps.

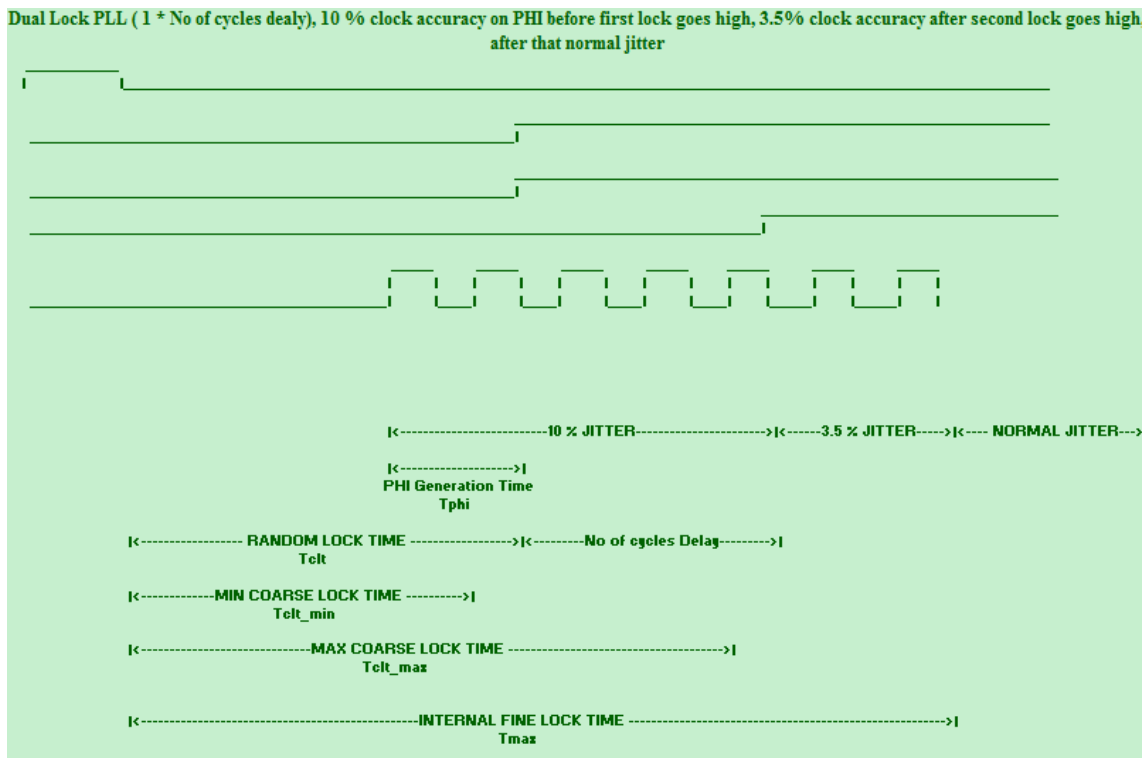
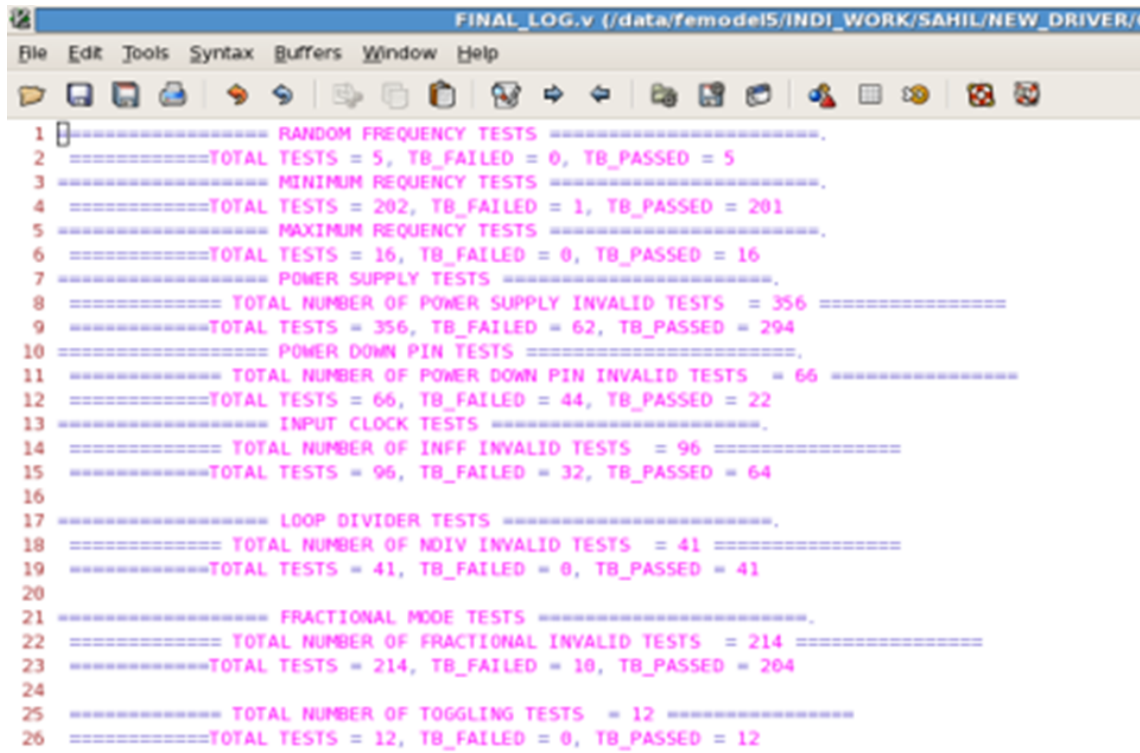


Figure 3.7: Lock protocol for dual lock

3.4 Final Report

When all the test cases are driven to the dut and verification process is done, then a final report is generated by the script which is shown below..



```

1 ===== RANDOM FREQUENCY TESTS =====.
2 =====TOTAL TESTS = 5, TB_FAILED = 0, TB_PASSED = 5
3 ===== MINIMUM FREQUENCY TESTS =====.
4 =====TOTAL TESTS = 202, TB_FAILED = 1, TB_PASSED = 201
5 ===== MAXIMUM FREQUENCY TESTS =====.
6 =====TOTAL TESTS = 16, TB_FAILED = 0, TB_PASSED = 16
7 ===== POWER SUPPLY TESTS =====.
8 ===== TOTAL NUMBER OF POWER SUPPLY INVALID TESTS = 356 =====
9 =====TOTAL TESTS = 356, TB_FAILED = 62, TB_PASSED = 294
10 ===== POWER DOWN PIN TESTS =====.
11 ===== TOTAL NUMBER OF POWER DOWN PIN INVALID TESTS = 66 =====
12 =====TOTAL TESTS = 66, TB_FAILED = 44, TB_PASSED = 22
13 ===== INPUT CLOCK TESTS =====.
14 ===== TOTAL NUMBER OF INFF INVALID TESTS = 96 =====
15 =====TOTAL TESTS = 96, TB_FAILED = 32, TB_PASSED = 64
16
17 ===== LOOP DIVIDER TESTS =====.
18 ===== TOTAL NUMBER OF NOIV INVALID TESTS = 41 =====
19 =====TOTAL TESTS = 41, TB_FAILED = 0, TB_PASSED = 41
20
21 ===== FRACTIONAL MODE TESTS =====.
22 ===== TOTAL NUMBER OF FRACTIONAL INVALID TESTS = 214 =====
23 =====TOTAL TESTS = 214, TB_FAILED = 10, TB_PASSED = 204
24
25 ===== TOTAL NUMBER OF TOGGLING TESTS = 12 =====
26 =====TOTAL TESTS = 12, TB_FAILED = 0, TB_PASSED = 12

```

Figure 3.8: Final Report

Chapter 4

Verification Environment for PLL

This chapter includes the information about verification environment for PLL, which will test PLL with all the test cases in one click. Flow diagram of verification environment for PLL is shown below.

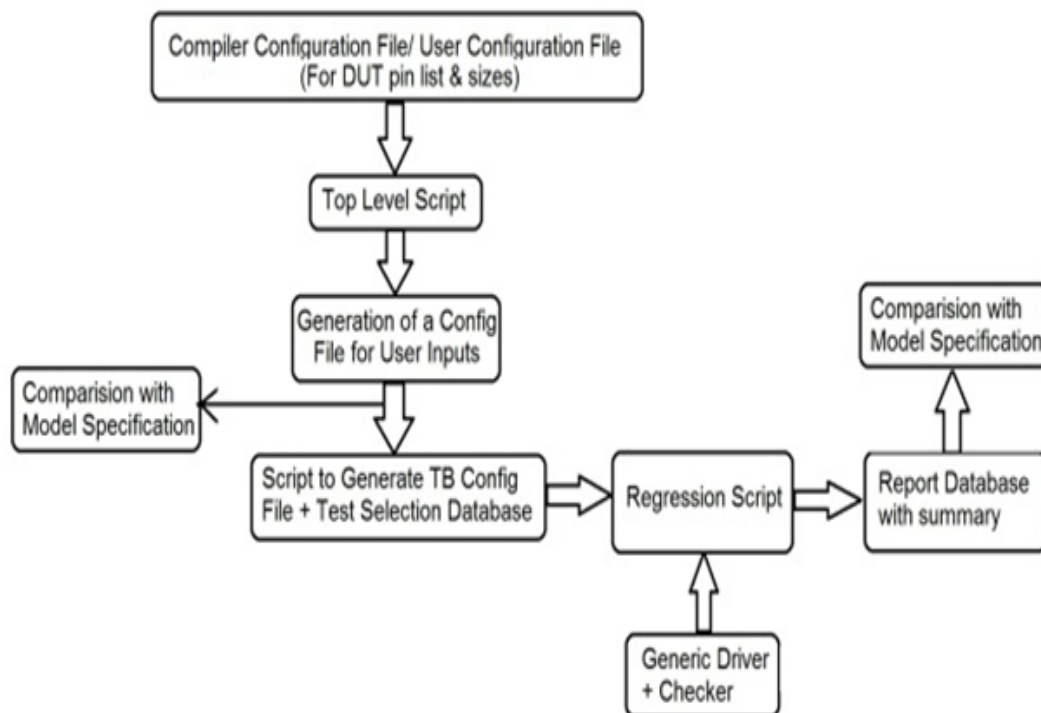


Figure 4.1: Flow diagram of verification for PLL

4.1 Compiler Configuration File/ User Input Configuration File

- This file is used for extracting pin list with sizes.
- This file can be configured in two forms..
 - Compiler configuration file or
 - User Input configuration file
- It provides the information about the test modes i.e. which test cases have to run from the generic test bench with the help of script.

4.2 Generation of Configuration File for User Inputs

- A top level script will run on the compiler configuration file or user configuration file, which will generate a configuration file for user inputs
- Benefits of this file compared to the previous one will be..
 - Only those inputs have to write which are supported by DUT functionality.
 - No input will be provided for user which is not supported by DUT functionality

4.3 Generation of Test Bench Configuration File + Test Selection Database

- A script will run on user input config file to generate test bench configuration file.
- These parameters will be generated with some calculation from user input config file such that user has to provide minimum number of inputs.

- The parameters required for Generic test bench will be provided from this file.

4.4 Generic Driver and Checker

- In this file all the test cases will be written which have to run on DUT.
- It will support..
 - Only those inputs have to write which are supported by DUT functionality.
 - No input will be provided for user which is not supported by DUT functionality

4.5 Regression Script

- This script will be used for running all possible test cases for all supported modes.
- It will generate a report based on tests which will also compared with the specifications

4.6 Algorithm for Generic Verification for all types of PLL

The testing of PLL will be done in verilog language so a TB-TOP module will be the top of test bench. A parameter file will be included with 'include in verilog. This file will contain the information about the PLL minimum input frequency, maximum input frequency, total time for lock, minimum width of power down pulse, minimum duty cycle of input frequency, maximum allowable jitter, maximum duty cycle of input frequency, minimum output frequency, maximum output frequency, minimum

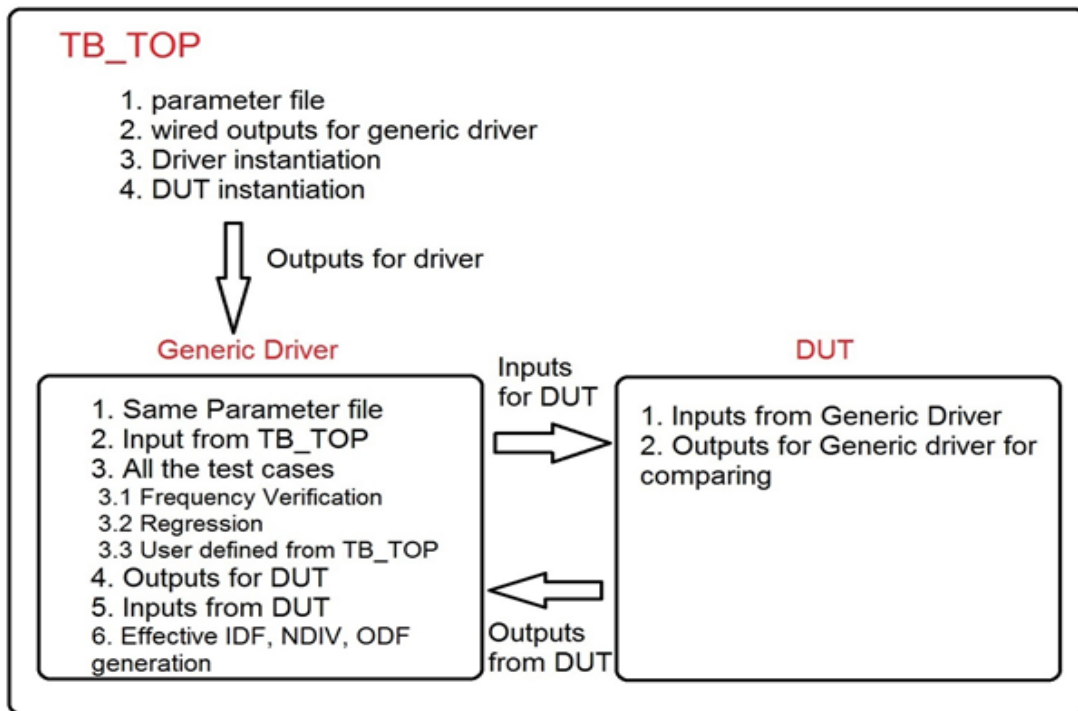


Figure 4.2: Overview of Generic Test Bench

duty cycle of output frequency, maximum duty cycle of output frequency and settling time for PHI etc.

Two modules generic driver and checker and PLL DUT will be instantiated. These two module ports will be connected through wires which are created in TB-TOP module. Generic driver will have all the test cases written for PLL and these test will be applied to PLL DUT. The result will be checked in the checker part and compared with the expected result. If the DUT outputs and expected outputs are matched then PLL has no error and if they are mismatched then checker will show the errors in the simulation result file.

Some of the test cases are specific to PLL functionality which can to be generic for all. These test cases can also be written in TB-TOP module. These test cases are called user defined test cases. A script will run for all the test cases at once and the report will be generated from the log files of simulations.

Chapter 5

Result

This behavioural PLL IP is used for complete SoC verification. SPICE manual checking for analog PLL takes huge time for verification and provide higher perfection. The behavioural verification environment for analog PLL gives reduction in verification time but also with that degrades accuracy of the verification. The analog functionality verification is not done by the behavioural verification environment. Verification environment for the behavioural models of PLL provides flexibility in verifying the PLL. All types of supported functionality can be checked for different modes of all PLL with this approach. An automated script is used which runs all the test cases supported by the behavioural PLL. It takes almost 2 to 3 hours which is a huge reduction compare to the SPICE checking which takes around 6 months to test the PLL. There is no need of verifying on the waveforms as in SPICE verification. All the results are provided in the simulation log files. The final report contains the information about total test cases applied to PLL and the total numbers of tested test case with the number of failing tests. Two separate columns will be provided for test passed and test failed. In this way an automated verification environment helps to reduce the time for verification of any PLL.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Verification of analog IP is very time taking such that it will create a large time to market. Use of behavioral models of analog IP is very time consuming. It provides less time for modeling and testing also the accuracy degrades as compare to analog simulations. PLL is an IP which is used to generate clock, which is an essential IP for any type of SoC.

Verification of behavioral model of PLL can be made simpler as the basic functionality of any kind of PLL is the multiplication of frequency with the same phase. A number of test cases are applied to PLL DUT with the help of script reduces very much amount for testing PLL. Analog testing of PLL for all test cases takes almost an year. But with the help of this verification environment gives a testing PLL in only few hours. Also result can also be seen in a simpler way.

6.2 Future Work

- Automation script phase 2 for running all the test cases.

References

- [1] <http://www.embedded.com/design/system-integration/4419244/Phase-locked-loops-in-an-IC-based-clock-distribution-system>
- [2] <http://en.wikipedia.org/wiki/Phase-locked-loop>
- [3] <http://www.analog.com/library/analogDialogue/archives/33-03/phase>
- [4] Scott E. Meninger and Michael H. Perrott, A Fractional-N Frequency Synthesizer Architecture Utilizing a Mismatch Compensated PFD/DAC Structure for Reduced Quantization-Induced Phase Noise, IEEE transactions on circuits and systems ii: analog and digital signal processing, vol. 50, no. 11, november 2003.
- [5] Ken Kundert, Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers, The Designers Guide, Copyright August 2006, Kenneth S. Kundert - Designers Guide Consulting, Inc.
- [6] Xiaojian Mao, Huazhong, Yang Hui Wang, Behavioral Modeling and Simulation of Jitter and Phase Noise in Fractional-N PLL Frequency Synthesizer.