FORMAL EQUIVALENCE VERIFICATION OF PHASE LOCKED LOOPS

Major Project
Submitted in Fulfilment of the Requirements
for the Degree of
Master of Technology
in
VLSI Design
by

Jigar Patel
11MECV12

Department of Electrical Engineering
Electronics & Communication Programme
Institute of Technology
Nirma University
Ahmedabad
May-2013

FORMAL EQUIVALENCE VERIFICATION OF PHASE LOCKED LOOPS

Major Project
Submitted in Fulfilment of the Requirements
for the Degree of
Master of Technology
in
VLSI Design
by

Jigar Patel
11MECV12

Dr. Usha Mehta
Internal Guide

Mr. Somesh Agarwal
Engineering Manager

Department of Electrical Engineering
Electronics & Communication Programme
Institute of Technology
Nirma University
Ahmedabad
May-2013

Declaration

This is to certify that

1. I, Jigar Patel, a student of Master of Technology in VLSI Design, Nirma University, Ahmedabad hereby declare that the project work "FORMAL EQUIVALENCE VERIFICATION OF PHASE LOCKED LOOPS" has been independently carried out by me under the guidance of Mrs. Shilpa Huddar , Design Engineer, Intel Technology India Private Limited, Bangalore and Dr. Usha Mehta, Associate Professor, Department of VLSI Design, Nirma University, Ahmedabad. I would also like to give a vote of thanks to Dr. N. M. Devashrayee, Program Co-ordinator, Department of VLSI Design, Nirma University, Ahmedabad for providing valuable support for project work. This Project has been submitted in fulfillment of the requirements for the award of degree Master of Technology(M.Tech.) in VLSI Design, Nirma University, Ahmedabad during the year 2012 - 2013.

2. I have not submitted this work in full or part to any other University or Institution for the award of any other degree.

Jigar Patel
11MECV12

Certificate

This is to certify that the Major Project entitled "FORMAL EQUIVALENCE VERIFICATION OF PHASE LOCKED LOOPS" submitted by Jigar Patel (11MECV12), towards the fulfilment of the requirements for the degree of Master of Technology in VLSI Design of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my 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 my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. Usha Mehta
Internal Project Guide
Institute of Technology,
Nirma University, Ahmedabad

Mr. Somesh Agarwal
Engineering Manager
Intel Technology India Pvt. Ltd.,
Bangalore

Dr. N. M. Devashrayee
Program Cordinator
Institute of Technology,
Nirma University, Ahmedabad

Dr. P. N. Tekwani
Head of EE Dept
Institute of Technology,
Nirma University, Ahmedabad

Date:

Place:Ahmedabad

Acknowledgement

First and foremost, sincere thanks to Mr. Somesh Agarwal, Engineering Manager, Intel Technology India Private Limited, Bangalore. He assigned me this project from which i learnt a lot. I owe him lots of gratitude for having a profound impact on this report.

I would like to thank my Mentor, Mrs. Shilpa Huddar, Design Engineer, Intel Technology India Private Limited, Bangalore for his valuable guidance. Throughout the training, She has given me much valuable advice on project work which I am very lucky to benefit from. Without her, this project work would never have been completed.

I would also like to thank Dr. K.R.Kotecha, Director, Institute of Technology, Nirma University, Ahmedabad for providing me an opportunity to get an internship at Intel Technology India Private Limited, Bangalore.

I would thank to my Project Co-ordinator, Dr. N.M. Devashrayee, Professor, VLSI Design, Institute of Technology, Nirma University, Ahmedabad for giving valuable support for project work.

I would also like to thank my internal guide Dr. Usha Mehta for her continuous guidance to make my project and presentation effective.

I would like to thank my all faculty members for providing encouragement, exchanging knowledge during my post-graduate program.

I also owe my colleagues in the Intel, special thanks for helping me on this path and for making project at Intel more enjoyable.

Jigar Patel
11MECV12

Abstract

Ever-growing complexity in VLSI is forcing logic design to move above the register transfer level (RTL). For example, functional specifications are being written in software. These specifications are written for clarity, and are not optimized or intended for synthesis. The Logic Synthesis as a process is prone to Bugs. There are too many transformations happening in logic synthesis which can alter the netlist in a wrong manner and make it infer functionality other than what was intended in the original RTL. These bugs are not intentional but happen by accident during synthesis tool development. So Formal Equivalence Verification (FEV) between the software specification and the implementation is needed.
In today's high-performance integrated circuits for clock & data recovery, modulation & demodulation, clock generation and frequency synthesis, PLLs are widely adopted. As the speed of systems increases, PLLs with higher operating frequency, faster lock acquisition and lower jitter are urgently in demand. The phase frequency detector (PFD), which helps PLLs achieve simultaneous phase and frequency error detection, is an indispensable functional block and plays an important role in improving the performance of the whole PLL system.

This report introduces the Formal Equivalence Verification approach for PLLs. It describes the method for formal equivalence verification. It also introduces the algorithm by which the FEV tool runs. This report imposes upon the importance of the Formal equivalence verification in VLSI design flow. Apart from FEV, this report also contains study work of various types of phase frequency detectors, its limitations and approaches to overcome those limitations and improve performance.

Brief Content

"Formal Equivalence Verification of Phase Locked Loops" is the definition of the project, this report is divided in to main two parts. first part contains two chapters in which, formal verification is presented. And second part contains three chapters which covers brief explanation about PLL and details about phase frequency detectors.

In first chapter, various methodologies of verification is described and more attention is given to formal equivalence verification (FEV) and algorithm behind it. It also covers importance of FEV. Second chapter shows power of tool required for formal equivalence verification. Third chapter gives brief knowledge about PLL and its operation. Some more attention is paid on design and performance of phase frequency detector (PFD) in forth and fifth chapters. They also focus on limitations of PFD and approaches to overcome those limitations. At the end, conclusion and references are presented.

Contents

List of Figures

Chapter 1

Formal Equivalence Verification

Pentium bug costs 4.75 million dollar. This proves importance of verification itself. Verification can be defined in several ways: A process used to demonstrate that the intent of design is preserved in it's implementation., Confirmation by examination and provision of objective evidence that specified requirements have been fulfilled.

1.1 Verification Methods

There are various methods of verification. Some of them are described below.

- Simulation Based Verification
  Testing and simulation are well-known and traditional techniques used to check that the software written is correct with respect to its functionality. Vigorous testing and code review techniques are available to test the software written in almost any programming language. These techniques are considered highly effective and very few coding bugs of any significance escape detection. However, very few defects in the final product are due to bugs contributed during coding.

- Formal Verification
  Formal verification is a technique that developers can use to construct systems that operate reliably. It is complementary to testing and should be used in conjunction with testing to increase reliability of the system being developed. It is worth noting that formal verification is not a way to ensure that the system being developed is 100% correct; it enhances reliability of the system by ensuring that it meets certain functional requirements, particularly at the earlier stages of design. Formal verification is based on formal methods which are mathematically based languages, techniques and tools for specifying and verifying systems. Specifying a system means writing down the requirements about the system in a mathematical language. Verification is the next step to specification and involves proving formally that the system meets its requirement. There are various tools that aid in specification and

verification. These tools provide notations and algorithms for a system developer to formally specify and/or verify a system.

- Specification
  Specification is the process of describing a system and its desired properties. Properties or requirements of a system being developed usually completely characterize the system and are generally written in a common language.
- Verification
  Formal verification is the next step to formal specification and involves checking/proving that a system satisfies a given property. Formal verification relies on building a mathematical model of the system and on formally specifying the requirements to be checked against the system. Verification tools, i.e. tools performing formal verification take two inputs: a system and its specification and check if the system satisfies the specification. Depending on how the modelling and the checking are done, there are two fundamental techniques in formal verification: model checking and theorem proving.
  1. Model Checking
  Model checking can be done only for systems that have a finite number of states. Model checkers, i.e., tools which perform model checking take two inputs: a finite state model of the system and a property specified formally. A model checker checks if the system satisfies the property and gives a 'yes' or 'no' answer. If the answer is 'no', i.e., if the system does not satisfy the property, model checkers also output a counter example, i.e., a run of the system which violates the property. The counter example can be analysed to discover bugs in the system design.
  2. Theorem Proving
  Theorem proving is a technique where both the system and its desired properties are expressed as formulas in some mathematical logic. The logic is presented by a formal system, which defines a set of axioms and a set of inference rules. Theorem proving is the process of finding a proof of a property from the axioms of the system by using the axioms and rules and also, derived definitions and intermediate lemmas. Theorem provers are tools that assist in theorem proving. They aid in constructing proofs of a system satisfying the property and are semi-automatic as against model checkers which are fully automatic. On the other hand, theorem proving can directly deal with infinite state spaces whereas model checking applies only to finite state systems.

- Functional Verification
  Functional verification is the task of verifying that the logic design conforms to specification. It requires set of test vectors to verify design. It is very difficult to get 100% coverage using this approach. There are three different approaches of

functional verification.

- Black Box Verification
  This approach can not look at or know about the inside of design. The functional verification must be performed without any knowledge of actual implementation of design. All verification must be accomplished through available interfaces without direct access to internal states of design and without knowledge of its internal structure and implementation.
- White Box Verification
  This approach has intimate knowledge and control of internals of design. This approach can ensure that implementation specific features behave properly.
- Grey Box Verification
  It is compromise between aloofness of black box approach and the dependence on implementation of white box verification. It increases coverage by adding some non functional units to increase visibility like software accessible register to observe internal states.

1.2 Introduction to Formal Verification

Formal Verification (FV) is a general set of methods that mathematically prove aspects of a design, in contrast to dynamic validation / simulation (which just checks specific values). Thus FV is:

- A way of creating evidence suggesting that a system either does or does not have some property
- By comparing a formal model of the system to a formal specification of the property
- Using logical inference to prove that the model does or doesn't meet the spec

Note that formal verification does not guarantee that a real system has an intended property. It only provides suggestive evidence about the real system the proof is about a formal model. The distinction is important because the evidence may be mistaken for several reasons: the formal model may not accurately describe the real system; the formal spec may not accurately describe the intended property; the logical inference system may not be sound; and the purported proof may not actually follow the rules of inference in the logic. All of these have happened in practice. Thus it's important to also test the real system.

Formal verification is different from dynamic validation (testing by simulation) in that formal verification is complete within some formally defined domain. Dynamic validation may use formal inference, but isn't complete within a well-defined domain. A hybrid verification method combines formal verification and simulation-based methods.

1.3 Formal Equivalence Verification

Formal equivalence verification (FEV) is the use of formal verification to show that two models are equivalent (under some set of assumptions). for example, that a circuit implementation works identically to the RTL it is implementing. It is a method of proving the equivalence of two different view of the same logic design. It uses mathematical technique to verify equivalence of a referenced design and a modified design. These tools can be used to verify the equivalence of RTL-RTL, RTL-gate, and gate-gate implementations. Since equivalence checking tools compare the target design with the reference design, It is critical that the reference design is functionally correct.

1.3.1 Why Formal Equivalence Verification is needed?

Logic Synthesis as a process is prone to Bugs. There are too many transformations happening in logic synthesis which can alter the netlist in a wrong manner and make it infer functionality other than what was intended in the original RTL. These bugs are not intentional but happen by accident during synthesis tool development. That can happen mostly during the elaboration stage as new verilog/VHDL constructs start showing up from time to time in new language standards. It takes time to build a robust Language analysis/elaboration engine as a front end to synthesis engine. Simulation is not a robust proof of correctness as it was never intended to be exhaustive. However we can run the same functional vectors and check our netlist to see if we are lucky to catch a problem in gates. This is never a full proof verification strategy. Equivalence checking comes to the rescue. And it is fool proof. EC mathematically proves that the RTL and Netlist are equivalent and does not need functional vectors.

Advantages of Formal Equivalence Verification:

- Verifies 100 %
- Faster than performing an exhaustive simulation

Limitation:

- It does not verify the timing of the design

1.4 DUV Partitioning

The DUV partitioning attempts to divide the DUV into smaller pieces for which the EC algorithms prove equivalence individually. The partition can be done vertically, horizontally or combination of both. The equivalence checkers take outputs from either model and compare only the logic that specifically drives these outputs. In this example, the comparison occurs between the pairs of logic partitions A1, A2, B1, B2 and C1, C2. The

equivalence checker successively picks a pair of inputs, one per DUV model for check. It iterates over all DUV outputs. For each iteration step, the tool eliminates all the logic that does not directly drive the given outputs. Horizontal partitioning works only if the equivalence checker indeed prove that the cut points are Boolean equivalent Vertical partitioning selects intermediate points inside the logic of either DUV model. This cuts the depth of the logic cones compared at each step as well as number of inputs and amount of logic covered by cone. The figure shows the illustration of this principle: C1, C2, C3 are the cut points for which the checker must prove equivalence for the partitioning to succeed. Either the user specifies the cut points manually or the EC tool has built in heuristics to find these automatically. Vertical partitioning using cut points C1, C2 and C3 allows the comparison process to prove equivalence for smaller sub cones with less number of inputs and logic. The XOR check now takes not only the model output to compare pair-wise but also the signals that represent the cut point in each model.

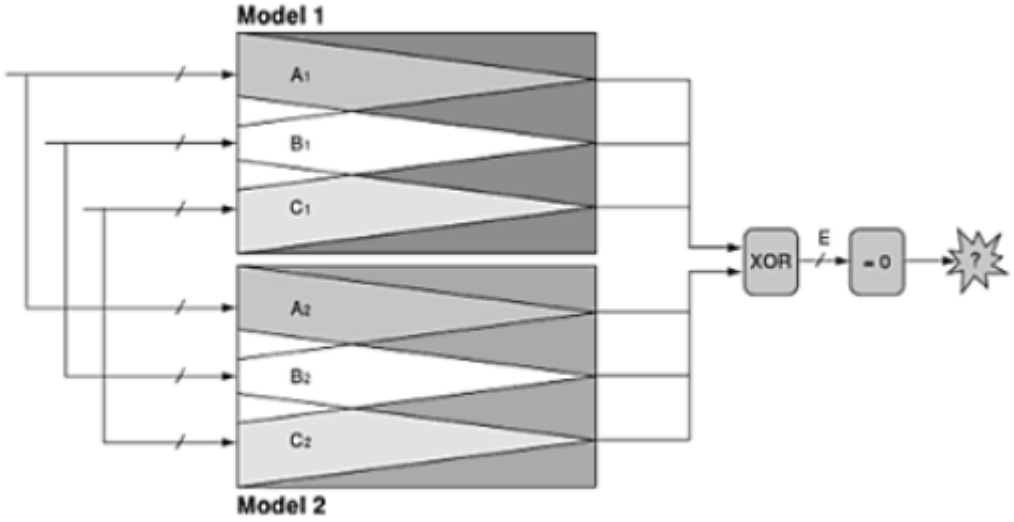

Figure 1.1: Horizontal Partitioning Approach

1.5 Mapping

Pairing correspond golden and revised key points. Key points are the points to map. So while doing FEV the key points of golden design should be mapped with the revised design.

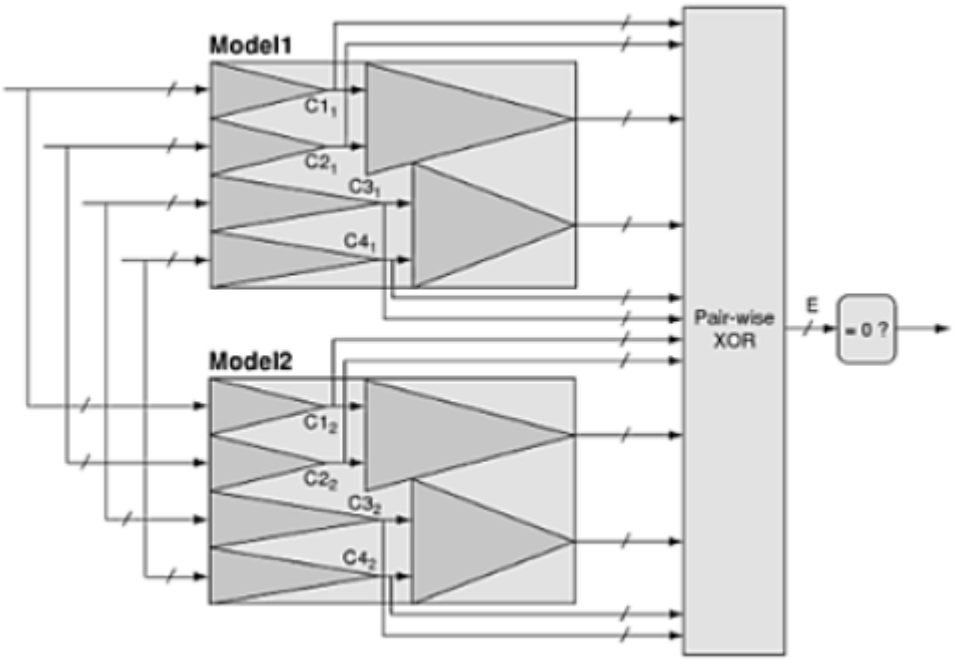

Figure 1.2: Vertical Partitioning Approach

1.6 State Matching FEV

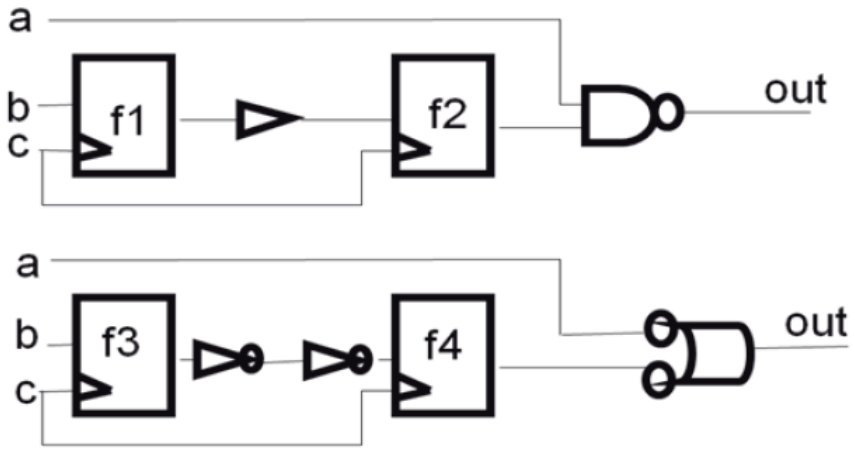

Figure 1.3: Example of Equivalent Design

FEV works on state matching algorithm. It does the partition of the whole circuit into many cones. And then compares the key points. Lets us take an example given fig 1.3.

- Question: Are the following two figures equivalent?

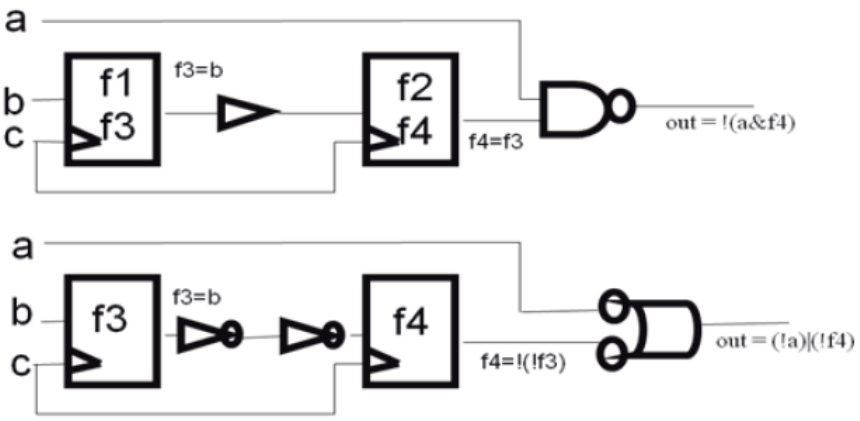

Figure 1.4: Building Equivalence of Design

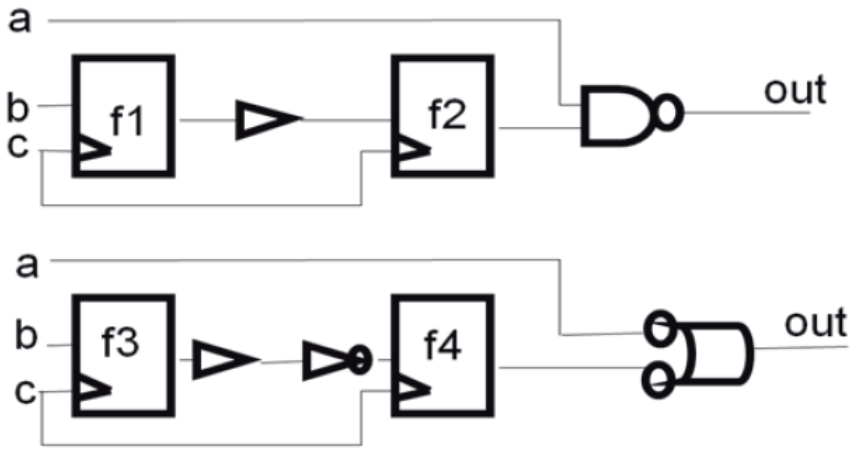

Figure 1.5: Example of Non-equivalence Design

f1=b	f3=b	f1=f3
f2=f1	f4=!f3	f4 !=f3
out =a & f2	out = (!a) & (!f4)	Both the outputs are not equal

Figure 1.6: Truth Table Non-Equivalence Design

- Answer: Yes, but again the question is how the tool will understand that both the figures are equivalent. So the tool does this comparison by using state matching algorithm. First of all it maps the key point. Once the key points are mapped then the comparison can be done. The design is converted into groups of cones. And the cones inputs and outputs are compared by mathematical equation. The steps are given below.
  Step 1: Map the key points.
  Check whether inputs of both the circuits are mapped
  Check whether outputs of both the circuits are mapped
  Check whether states of both the circuits are mapped. i.e. f1->f3, f2->f4
  Step 2: Build equations.
  Step 3: Now compare the equations.

So, FEV performs an important role in VLSI design flow. It is a complete mathematical process and does not require test patterns to verify designs.

Chapter 2

FEV Tool

There are various tools used for verification of design. FEV is a bundle of tools which performs various tasks. Whole process is divided in few steps.Each step generates log file which reports all errors, warnings, and reasons of failure.

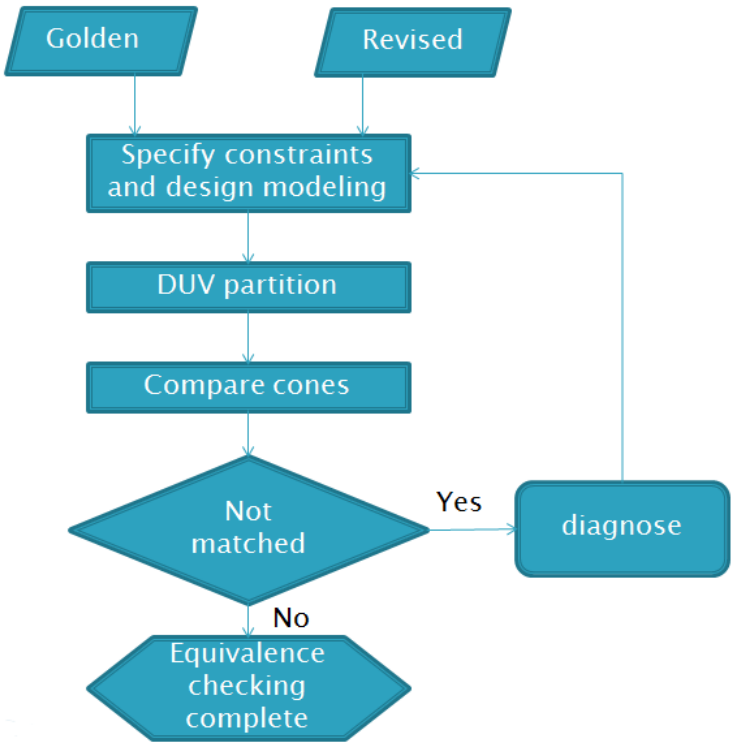

Figure 2.1: fev flow

- Set up
- Conversion to specified formate
- Interface mapping

- Comparison of designs
- Debugging of mismatches

2.1 Inputs for the FEV run

A. RTL
B. Schematic netlist
C. Black Box list
D. Map file

2.2 Set Up

In this step, the tool verifies that provided pointers of the input files are correct.

2.3 conversion to specified formate

Both the representations of design need to be converted to specific formate to make them comparable.

2.4 Interface Mapping

Tool must be aware of the points which needs to be compared. Auto mapper will map the interface signals which have the same names. Operator has to map remaining points in map file. all nodes and state elements (flops) should be mapped. Designs will be broken into cones based on mapped state elements.

2.5 Comparison of Designs

Equivalent cones of both representations are compared in this stage. Log file contains total number of matching cones, mismatching cones and problematic cones.

2.6 Debugging of Mismatches

There are different ways to analyze and debug the non-equivalent points.

- Diagnosis manager: It will tell the corresponding point.

- Schematic viewer: Viewing the schematic of both Golden and Revised design.
- Source code: By viewing the source code of Golden and Revised design.

2.7 Features of Tool

Mapping: By mapping state elements and equivalent nets, size of cone can be reduced. there are various cut attributes which makes debugging an easy task.
Debugger: There are some common causes of mismatches which debugger can directly detect.
A. Cones have inverted output.
B. Cones have different inputs.
What-if analysis: This feature helps to set any value to any net to make debugging easy.

2.8 Black Box

FEV is used to verify digital designs. There are some designs which contain analog blocks. These analog blocks are thoroughly verified by expert tools. Some digital blocks may contain zero delay loop (loop without state element). FEV tool can not verify such blocks. By defining them Black boxes, tool can be prevented from entering in these blocks. Tool just checks interfaces of black box and defines it as cut points.

2.9 Frequently Seen Issues

- Top level and black box pin name mismatches.
- Extra components in either schematic or RTL.
- Incorrect logic gates used in feeding compare points.
- Lots of cells not listed in initial black box lists can't be extracted and need to be manually black boxed during the run.

2.10 Debug Tips

- Check all black boxes are correctly mapped. and their interface pins are matching.

- Look for any unmapped points. All these points must be debugged or ignored before proceeding.
- Check cone cut points (inputs and outputs) are correctly mapped.
- If cone is big and complex, map internal matching nets to make them cut points.
- Check for common causes of mismatch.
- Use schematic viewer to compare both schematics.

In this way, FEV tool is very user friendly tool to debug mismatches. It is obvious from this chapter that automation has made the design and verification processes much easier and faster.

Chapter 3

Phase Locked Loops

3.1 Introduction

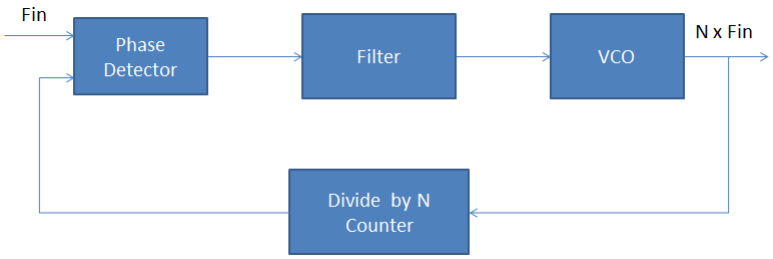

Figure 3.1: Block Diagram of Phase Locked Loop

Phase Locked Loop (PLL) circuits are used for frequency control. Figure 3.1 contains a block diagram of a basic PLL frequency multiplier. The operation of this circuit is typical of all phase locked loops. It is basically a feedback control system that controls the phase of a voltage controlled oscillator (VCO). The input signal is applied to one input of a phase detector. The other input is connected to the output of a divide by N counter. Normally the frequencies of both signals will be nearly the same. The output of the phase detector is a voltage proportional to the phase difference between the two inputs. This signal is applied to the loop filter. It is the loop filter that determines the dynamic characteristics of the PLL. The filtered signal controls the VCO. Note that the output of the VCO is at a frequency that is N times the input supplied to the frequency reference input. This output signal is sent back to the phase detector via the divide by N counter. Normally the loop filter is designed to match the characteristics required by the application of the PLL. If the PLL is to acquire and track a signal the bandwidth of the loop filter will be greater than if it expects a fixed input frequency. The frequency range which the PLL will accept and lock on is called the capture range. Once the PLL is locked and tracking a signal the range of frequencies that the PLL will follow is called the tracking range. Generally the tracking range is larger than the capture range. The loop

filter also determines how fast the signal frequency can change and still maintain lock. This is the maximum slewing rate. The narrower the loop filter bandwidth the smaller the achievable phase error. This comes at the expense of slower response and reduced capture range.

3.2 The Phase detector

The first component in our PLL is the phase detector. The two types of phase detectors, an XOR gate and a phase frequency detector (PFD), have significantly different characteristics. This makes understanding the limitations and performance very important. Selection of the type of phase detector is the first step in a PLL design.

3.2.1 The XOR Phase Detector

Figure 3.2: XOR phase detector

The XOR PD is simply an exclusive OR gate. When the output of the XOR is a pulse train with a 50 percent duty cycle (square wave), the PLL is said to be in lock; or in other words, the clock signal out of the PLL is synchronized to the incoming data, provided the conditions stated below are met. Consider the XOR PD shown in Fig 3.2. Let's begin by assuming that the incoming data are a string of zeros and that a divide by two is used in the feedback loop. The output of the phase detector is simply a replica of the dclock signal. Since the dclock signal has a 50 percent duty cycle, it would appear that the PLL is in lock. If a logic" 1" is suddenly applied, there is no way to know if the clock signal is synchronized (the clock rising edge coincides with the center of the data bit) to the data. This leads to the first characteristic of an XOR PD;

1. The incoming data must have a minimum number of transitions over a given time interval.

Now consider the situation when the output of the phase detector, with the data input being a string of zeros, is applied to a simple RC low-pass filter. If RC period of the clock signal, the output of the filter is simply VDD/2. This leads to the second characteristic of the XOR phase detector.

2. With no input data, the filtered output of the phase detector is VDD/2.

The voltage out of the loop filter is connected to the input of the VCO. Consider the typical characteristics of a VCO shown in Fig 3.3. The frequency of the square wave

output of the VCO is f_{center} when Vin ($= V_{center}$) is VDD/2 (typically). The other two frequencies of interest are the minimum and maximum oscillator frequencies, f_{min} and f_{max} possible, with input voltage V_{min} and V_{max} ,respectively. It is important that the VCO continue to oscillate with no input data. Normally, the VCO is designed so that the nominal data input rate and the VCO center frequency are the same. This minimizes the time it takes the PLL to lock.

Since the output of the PD is averaged, or more correctly integrated, noise injected into the data stream (a false bit) can be rejected. A fourth characteristic of this PD is:
3. The XOR DPLL has good noise rejection.

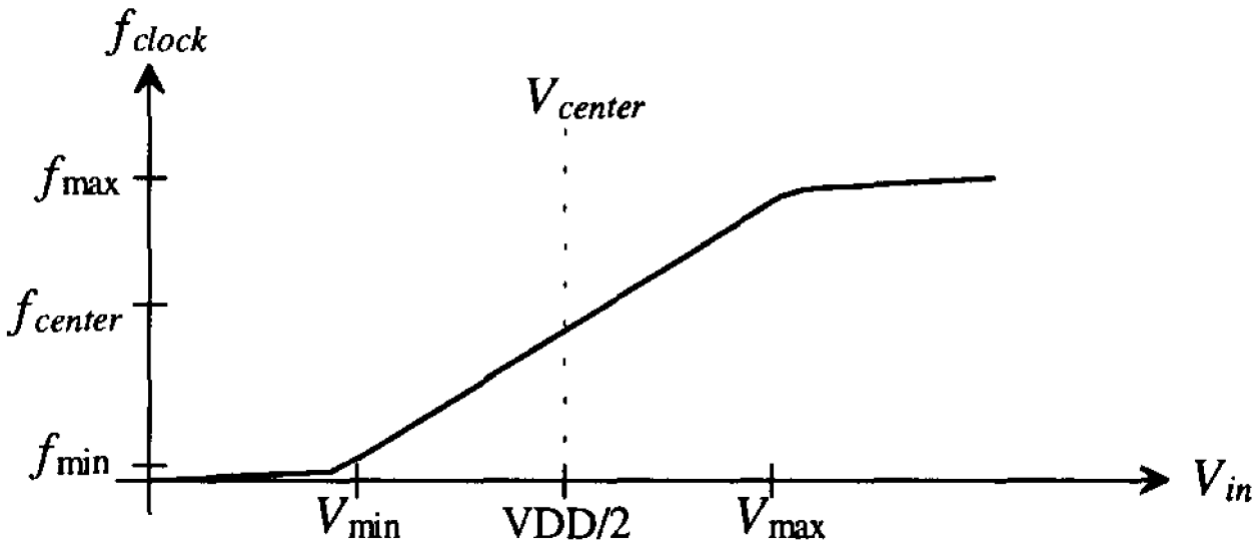

Figure 3.3: Output frequency of VCO vs input control voltage

The XOR PLL will lock on harmonics of the data.
4. The VCO operating frequency range should be limited to frequencies much less than $2^{*}f_{clock}$ and much greater than $O.5f_{clock}$ where f_{clock} is the nominal clock frequency for proper lock with a XOR PD.

The loop filter used with this type of PD is a simple RC low-pass filter. Since the output of the PD is oscillating, the output of the filter will show a ripple as well, even when the loop is locked. This modulates the clock frequency, an unwanted characteristic of a PLL using the XOR PD. This characteristic can be added to our list:
5. A ripple on the output of the loop filter with a frequency equal to the clock frequency will modulate the control voltage of the VCO.

When the loop is locked, the clock rising edge is centered on the data. Therefore, the

phase difference between dclock and the data, under locked conditions, may be written as $\triangle\Phi = \Pi/2$

3.2.2 The Phase Frequency Detector

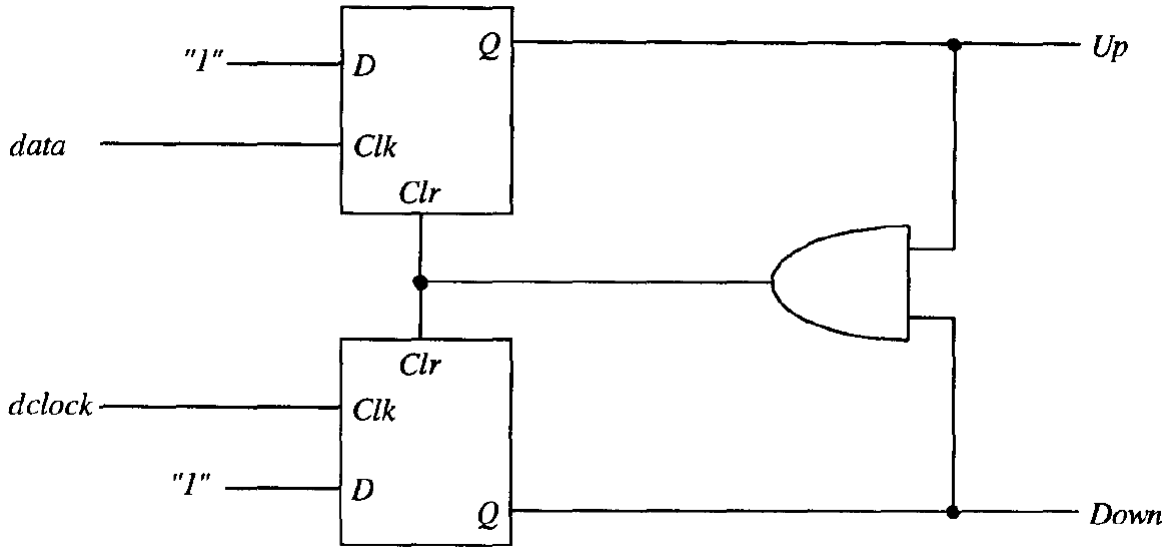

Figure 3.4: Phase frequency detector

A schematic diagram of the phase frequency detector is shown in Fig 3.4. The output of the PFD depends on both the phase and frequency of the inputs. This type of phase detector is also termed a sequential phase detector. It compares the leading edges of the data and dclock. The first thing we notice is that the data pulse width and the dclock pulse width do not matter. If the rising edge of the data leads the dclock rising edge, the "up" output of the phase detector goes high while the Down output remains low. This causes the dclock frequency to increase, having the effect of moving the edges closer together. When the dclock signal leads the data, Up remains low while the Down goes high a time equal to the phase difference between dclock and data. Notice that, unlike the XOR PD, the outputs remain low when the loop is locked. Again, several characteristics of the PFD can be described:

1. A rising edge from the dclock and data must be present when doing a phase comparison.
2. The width of the dclock and the data is irrelevant.
3. The PFD will not lock on a harmonic of the data.
4. The outputs (Up and Down) of the PFD are both logic low when the loop is in lock, eliminating ripple on the output of the loop filter.
5. This PFD has poor noise rejection; a false edge on either the data or the dclock inputs will drastically affect the output of the PFD.

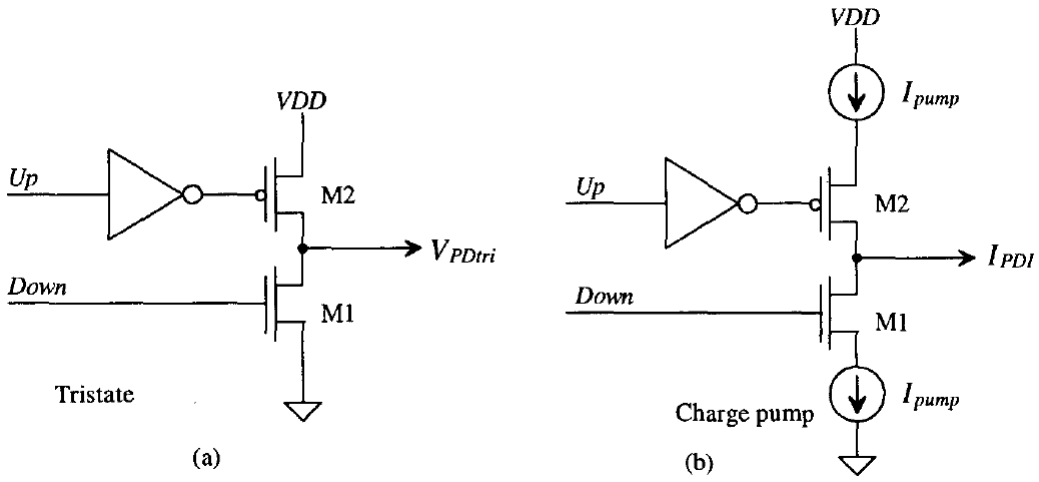

Figure 3.5: (a) Tri-state and (b) charge pump outputs of the PFD

The output of the PFD should be combined into a single output for driving the loop filter. There are two methods of doing this, both of which are shown in Fig 3.5. The first method is called a tri-state output. When both signals, Up and Down, are low, both MOSFETs are off and the output is in a high-impedance state. If the Up signal goes high, M2 turns on and pulls the output up to VDD while if the Down signal is high the output is pulled low through Ml. The main problem that exists with this configuration is that the power supply variations can significantly affect the output voltage when M2 is on. The effect is to modulate the VCO control voltage. This wasn't as big a problem when the XOR PFD was used because of the averaging taking place. The second configuration shown in this figure is the so-called charge pump. MOS current sources are placed in series with Ml and M2. When the PFD Up signal goes high, M2 turns on, connecting the current source to the loop filter. Since the current source can be made insensitive to supply variations, modulation of the VCO control voltage is absent.

3.3 Loop Filter

The output of charge pump is given to loop filter. The loop filter is the brain of the DPLL. If the loop-filter values are not selected correctly, it may take the loop too long to lock, or once locked small variations in the input data may cause the loop to unlock. The pull-in range, $\triangle w_P$, is defined as the range of input frequencies which the DPLL will lock to. The time it takes the loop to lock is labeled T; and may be a very long time. If the center frequency of the DPLL is 10MHz and the pull-in range is 1 MHz, the DPLL will lock on an input frequency from 9 to 11 MHz in a time T; (assuming N =1). The lock range, $\triangle w_L$, is the range of frequencies in which the DPLL locks within one single beat note between of the divided down output (dclock) and input (data) of the DPLL. The operating frequency of the DPLL should be limited to the lock range for normal

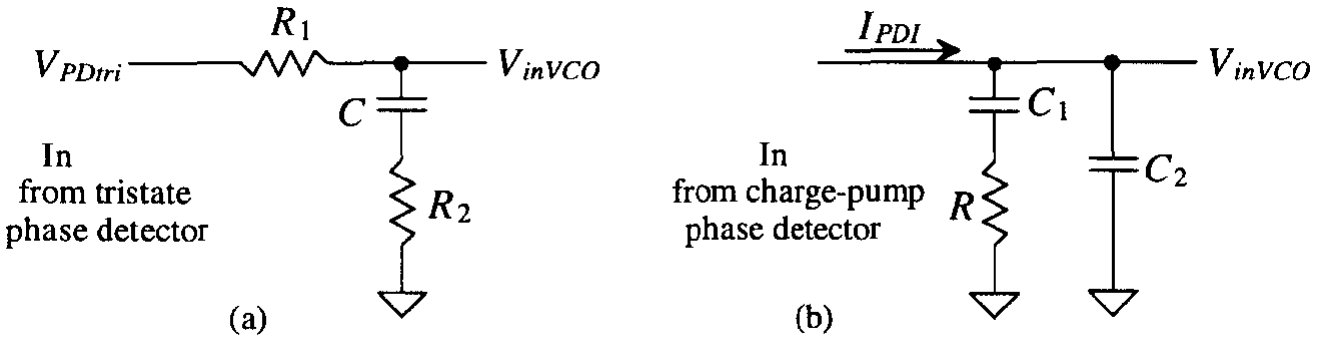

Figure 3.6: Loop filters for (a) tristate output and (b) charge-pump output

operation. Once the DPLL is locked, it will remain locked as long as abrupt frequency changes, $\triangle$w, in the input frequency (input frequency steps) over a time interval t are much smaller than the natural frequency of the system squared, that is, $\triangle \mathrm{w}/\mathrm{t} < \triangle {w_n}^2$.

3.4 Voltage Controlled Oscillator

The loop filter generates control voltage that is used to bias the VCO. Based on the control voltage, the VCO oscillates at a higher or lower frequency, which affects the phase and frequency of the feedback clock. If the PFD produces an up signal, then the VCO frequency increases. A down signal decreases the VCO frequency. The VCO stabilizes once the reference clock and the feedback clock have the same phase and frequency.

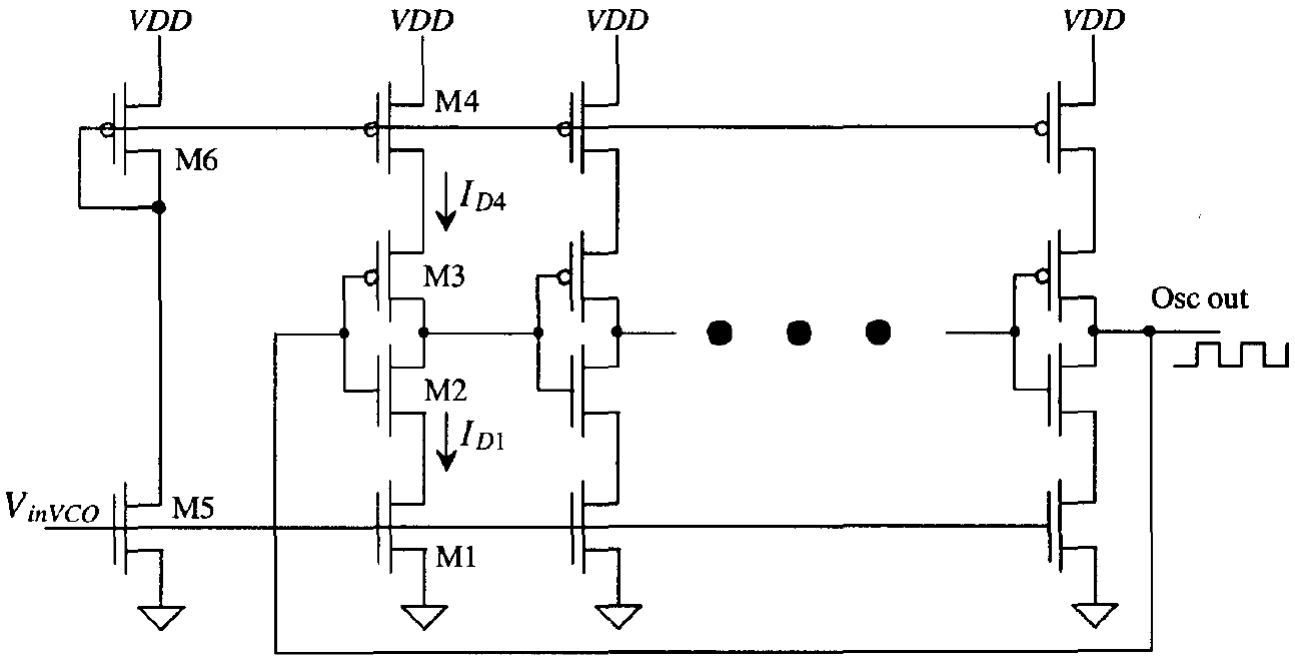

Figure 3.7: Ring type VCO

Its operation is similar to the ring oscillator. MOSFETs M2 and M3 operate as an inverter, while MOSFETs Ml and M4 operate as current sources. The current sources, Ml and M4, limit the current available to the inverter, M2 and M3; in other words, the inverter is starved for current. The MOSFETs M5 and M6 drain currents are the same and are set by the input control voltage. The currents in M5 and M6 are mirrored in each inverter/current source stage.
The oscillation frequency of the current-starved VCO for N stages is

$$f_{osc} = I_D/(N * C_{tot} * VDD), \qquad C_{tot} = drain\ \ capacitance\ \ of\ \ M2\ \ and\ \ M3 \quad (3.1)$$

PLLs are used in various applications such as clock/data recovery, frequency synthesizer and so on. All blocks of PLL are very important to characterise various operational parameters. These parameters varies according to requirements of application. In this way, PLL plays significant role in signal processing world.

Chapter 4

Performance Parameters for a PLL

Phase Locked Loops (PLLs) have many characteristics,such as the lock time, the phase noise, the reference spur, the dead zone, and the comparison frequency, and those characteristics are related to one another. If we improve some performances, others may deteriorate. Many techniques exist for achieving a PFD with no dead zone and many methods exist for increasing the operating frequency of the PFD. In this chapter, a simple relationship between the dead zone and the operating frequency of the PFD is described, and a useful scheme for obtaining both characteristics is proposed. No dead zone is important for accurate frequency generation, low phase noise in frequency synthesizer PLLs, and low timing jitter in clock generator PLLs. In some applications, such as radio tunes, an appropriate dead zone is needed. If the input frequency of the PFD is increased, the lock time can be shortened by using a PLL with a fixed loop bandwidth and the reference spur can be reduced in many PLL applications, especially in a fractional-N frequency synthesizer. In many PLL applications, an appropriate delay is added in the PFD reset path to avoid the dead zone problem. Unfortunately, because of this delay, there will be short pulses on both the UP and the DOWN signals, even in the locked state. Thus, the charge-pump current will switch on and off,and current spikes will appear on the charge-pump output at the reference frequency. This will cause reference spurs to appear in the PLL output spectrum at a frequency offset from the carrier equal to this reference frequency. Thus, too long a delay in the PFD for no dead zone is the cause of reference spur problems. A careful design needs to consider the delay length of the PFD. Aside from the characteristics of the reference spur and the dead zone, the reset delay length of the PFD is important in many PLL characteristics, such as the maximum operating frequency of the PFD, the phase error, and the lock time.

4.1 Dead Zone in the PLL

The dead zone, crossover distortion, of the PLL is the region where the charge-pump currents can not flow proportionally to the phase error. The main cause of the dead zone is the relationship between the propagation delay of the internal gates for the reset of the PFD and the switching time of the charge-pump currents. A conventional technique

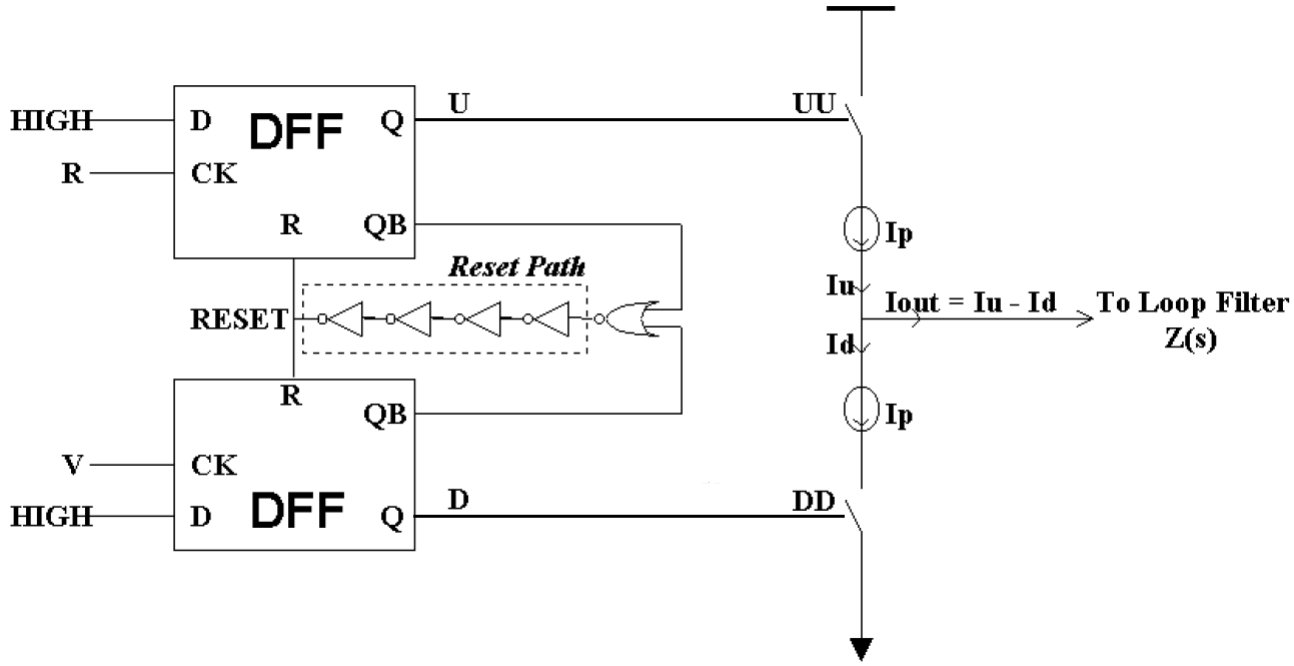

Figure 4.1: Circuits of PFD and charge pump

to avoid the dead zone problem is to make the delay in the PFD reset path longer than the switching time of the charge-pump currents. This switching time of the charge-pump currents is a function of the charge-pump currents, the load capacitance of the charge-pump MOS switch, and the drivability of the buffer.

4.2 Operating Frequency of the PFD

As the comparison frequency of the PLL increases, some characteristics, such as the lock time, the phase noise, and the reference spur, can be improved. The maximum operating frequency of the PFD is

$$f_{PFDmax} = 1/(2 * \triangle T_R) \tag{4.1}$$

As the reset delay of the PFD, decreases, the maximum operating frequency of the PFD increases.

4.3 Reference Spur in the PLL

The reference spur is due to periodic charge-pump current spikes at the reference frequency, even in the lock state.

4.4 Proposed scheme for optimal T_R in design of the pfd

As seen from the above, the amount of delay in the PFD reset path is the key parameter common to the dead zone, the operating frequency of the PFD, and the reference spur.

Therefore, if these three characteristics are to be improved the delay of the PFD reset path must be determined carefully in many PLL applications. Figure 4.1 presents the PFD circuit with D flip-flops. A timing diagram of the signals for the PFD in Fig. 4.1 is shown in Fig. 4.2, where T_R is the delay in the PFD reset path, Te is a given phase error, Vth is the threshold voltage of the charge-pump current switch, and Tth is the time for the input voltage of the charge-pump switch from zero to Vth. Tth is also the switching time of the charge-pump currents. Vth and Tth are determined for a given PLL application. The difference between Fig. 4.2 (a) and (b) is the size of $\triangle T_R$. If

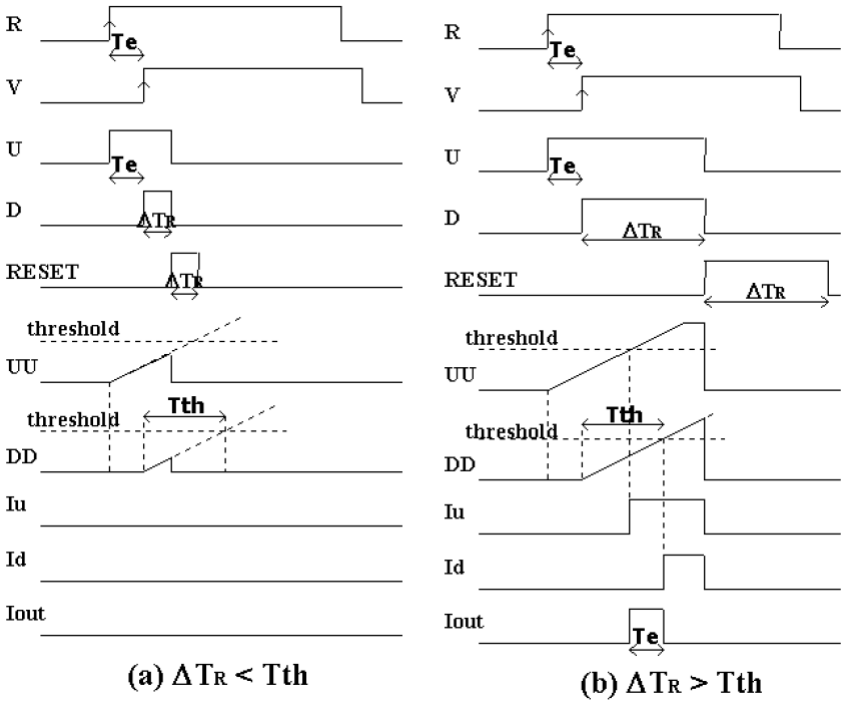

Figure 4.2: Timing diagrams of PFD and charge-pump: (a) $\triangle T_R <$ Tth and (b) $\triangle T_R >$ Tth, (courtesy of ref[7])

$\triangle T_R <$ Tth, as in Fig. (a), the charge-pump switch can not be closed, Iout can not flow during the time Te, and there is a dead zone. However, if $\triangle T_R >$ Tth, as shown in Fig. (b), charge-pump switch can be closed, Iout can flow during the time Te, and there is no dead zone. Therefore, to avoid the dead zone problem, we must design the delay of PFD reset path ($\triangle T_R$) to be longer than the charge-pump current switching time (Tth). The minimum delay in the PFD reset path ($\triangle T_{Rmin}$) in order to avoid a dead zone is related to T_{th} as $\triangle T_{Rmin} = T_{th}$. The maximum delay in the PFD reset path ($\triangle T_{Rmax}$) is the maximum operating frequency of the PFD (f_{PFDmax}), which is derived by

$$\triangle T_{Rmax} = 1/(2 * f_{PFDmax}) \tag{4.2}$$

In addition, to reduce the reference spur, we made the delay in the PFD reset path ($\triangle T_R$) as short as possible. Therefore, from the above relations, a useful scheme for

optimal T_R in PFD design is as follows: First, for a particular application, Tth is calculated, and $\triangle T_{Rmin}$ is determined. Second, f_{PFDmax} and T_{Rmax} are determined for the application. Last, T_R is made to be a little longer than $\triangle T_{Rmin}$ in order to reduce the reference spur.

$$\triangle T_{Rmin} < \triangle T_R < \triangle T_{Rmax} \tag{4.3}$$

At the end, it is proved that length of reset path delay should be in a range that removes dead zone but does not affect maximum operating frequency of PLL and creates minimum reference spur which can be filtered out by loop filter.

Chapter 5

Non linear Phase Frequency Detectors

Phase-locked loops (PLL) are widely adopted in today's high-performance integrated circuits for clock & data recovery, modulation & demodulation, clock generation and frequency synthesis. As the speed of their host systems increases, PLLs with higher operating frequency, faster lock acquisition and lower jitter are urgently in demand. The PFD, described in last chapter is linear in tracking the phase of reference signal. To increase the speed of PLL locking, PFD with non linear in behaviour is adopted. In this chapter, the design and operation of non linear PFD are covered in detail.

5.1 Non linear PFD for Fast-Lock Phase-Locked Loops

The phase frequency detector (PFD), which helps PLLs achieve simultaneous phase and frequency error detection, is an indispensable functional block and plays an important role in improving the performance of the whole PLL system. To obtain PLLs that can accomplish fast lock while at the same time maintain the optimum jitter suppression at lock-in, non-linear PFDs are a popular choice.

In this chapter, a new nonlinear PFD is presented which uses the same amount of transistors as the conventional PFD does, while simultaneously accomplishes self-adjusting π nonlinearity and improves the lock speed by about 20%.

5.1.1 Design Principle

The logic schematics of the widely-used conventional tristate PFD (PFDcon) and the proposed non-linear PFD (PFDnew) are demonstrated in Fig 5.1. The only difference between them is that two AND gates are inserted into the Reset signal path of the proposed PFD to gate-control the Reset signal. As the timing diagram of Fig 5.2 shows, for the proposed PFD, if Fref leads Fvco by not less than π (i.e. $\triangle\phi >= \pi$), the Reset signal only passes the AND gate controlled by Fvco and so UPnew keeps at logic "1" until $\triangle\phi < \pi$, when the Reset signal can pass both of the AND gates and resets both of the DFFs. when the input phase error is larger than π or smaller than -π, the PFD

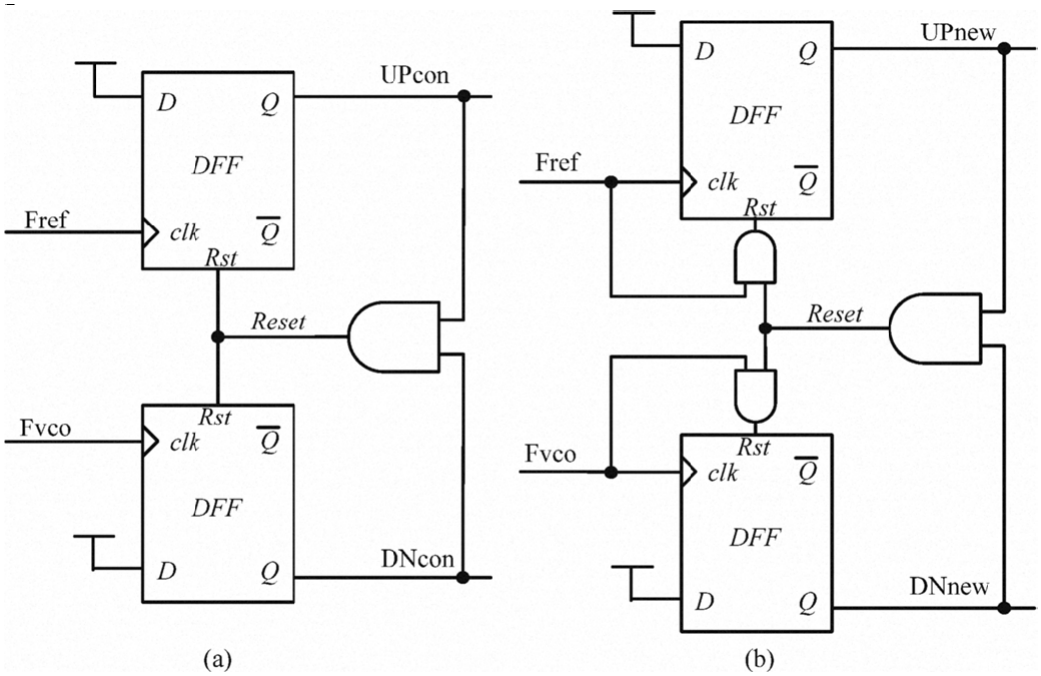

Figure 5.1: Logic schematics of (a) the conventional tristate PFD and (b) the proposed nonlinear PFD, (courtesy of ref[11])

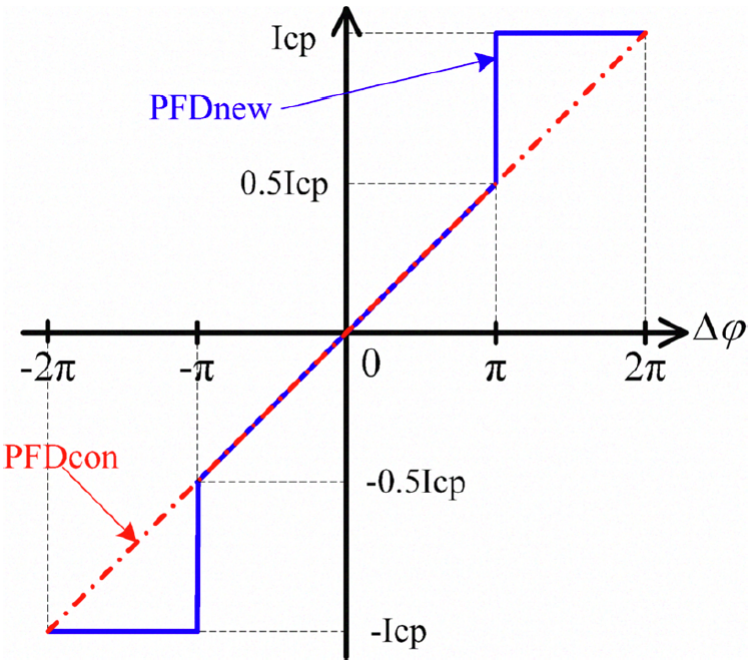

Figure 5.2: The transfer characteristic of the nonlinear PFD, (courtesy of ref[11])

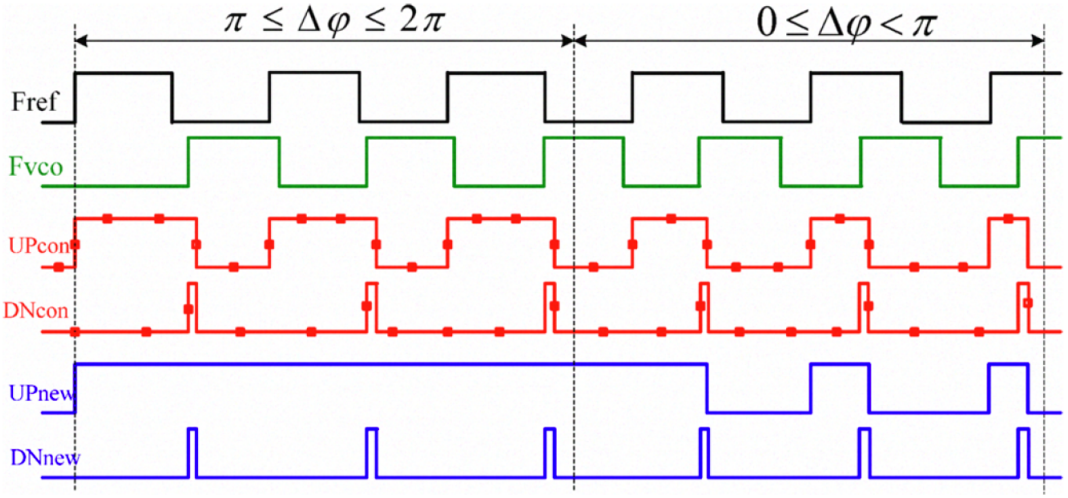

Figure 5.3: Timing diagram of the conventional PFD and the proposed PFD, (courtesy of ref[11])

works in the constant output region and as a result the lock process can be accelerated; when the PLL is in lock, the PFD works in the linear gain region and so a proper loop bandwidth can be maintained for low output jitter.

5.1.2 Difference in CMOS Circuit

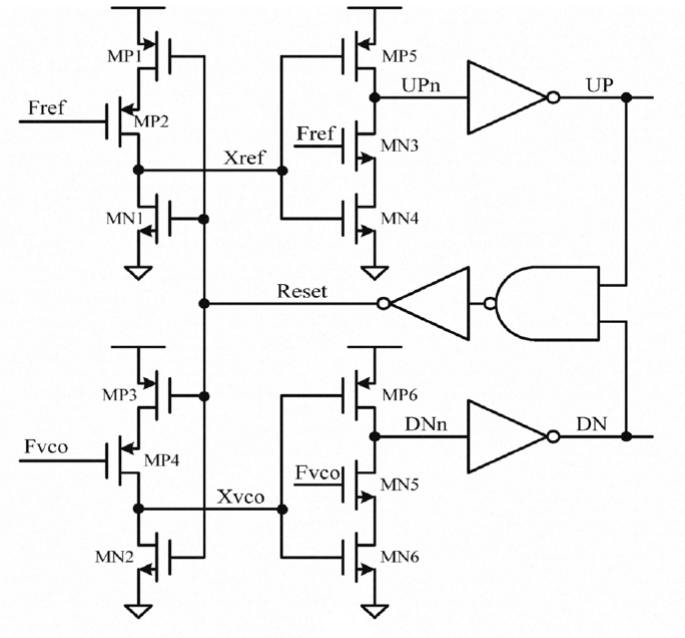

Figure 5.4: Linear PFD, (courtesy of ref[11])

A type of circuit embodiments of the conventional PFD and the proposed PFD is shown in Fig 5.4 and fig 5.5. Here number of transistor in both the circuits are same. The circuit of the proposed PFD works as follows. At the beginning, UPn=DNn=l,

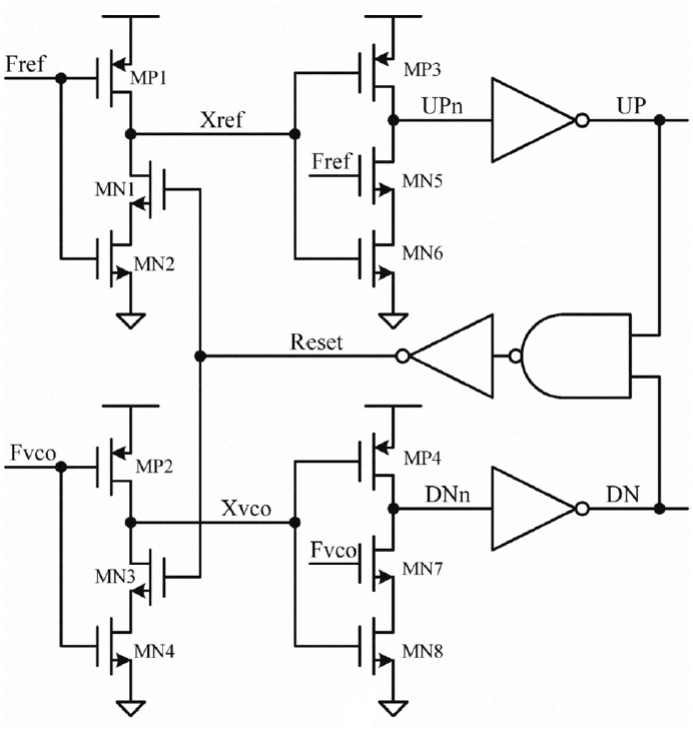

Figure 5.5: Nonlinear PFD, (courtesy of ref[11])

UP=DN=O and so Reset=O, which shuts off MN1 and MN3. When Fref=Fvco=O, Xref and Xvco are precharged to 1; meanwhile, MN5 and MN7 are shut off, so UPn and DNn are not affected and Reset keeps 0. When Fref (Fvco) rises up to 1, MN5 (MN7) turns on and so UPn (DNn) discharges to 0, which results in UP=l (DN=l). If Fref leads Fvco by less than π (i.e. $\triangle\phi < \pi$), Reset signal will become 1 when Fref and Fvco are both 1, which makes Xref and Xvco go to 0 simultaneously. Subsequently, UP and DN will be reset to 0 at the same time. Apparently, in this condition ($|\triangle\phi| < \pi$), the proposed PFD circuit works like the conventional PFD, so it has a linear gain of I_{cp}. Otherwise, if Fref leads Fvco by not less than π (Le. $|\triangle\phi| > \pi$), Reset signal will become 1 when Fref=O and Fvco= 1, so only Xvco is discharged to 0, which means only DN will be reset to 0 while UP will keep 1. Resultantly, in this condition ($|\triangle\phi| >= \pi$) the proposed PFD circuit has a constant output of I_{cp}.

5.2 A Non linear PFD with Zero Blind Zone

In Previous chapter, It is described that PFD is bothered by the dead zone problem when the input phase error between Fref and Fvco approaches zero. By inserting a deliberate delay td into the reset signal path, the dead zone problem can be alleviated effectively. Therefore, the conventional PFD should ideally have a linear transfer characteristic as shown by Fig 5.6(a). However, while alleviating the dead zone problem, the inserted delay td brings about a detrimental side effect called blind zone. The blind zone phenomenon of the conventional PFD occurs when the input phase error approaches 2π, as illustrated

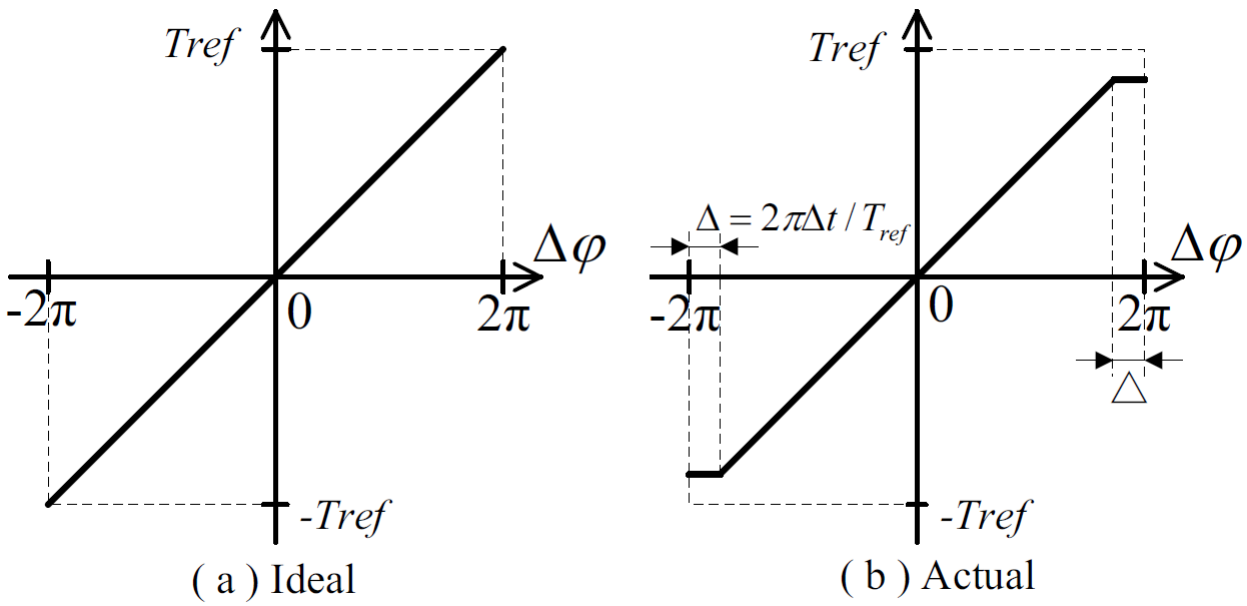

Figure 5.6: The transfer characteristics of the conventional PFD

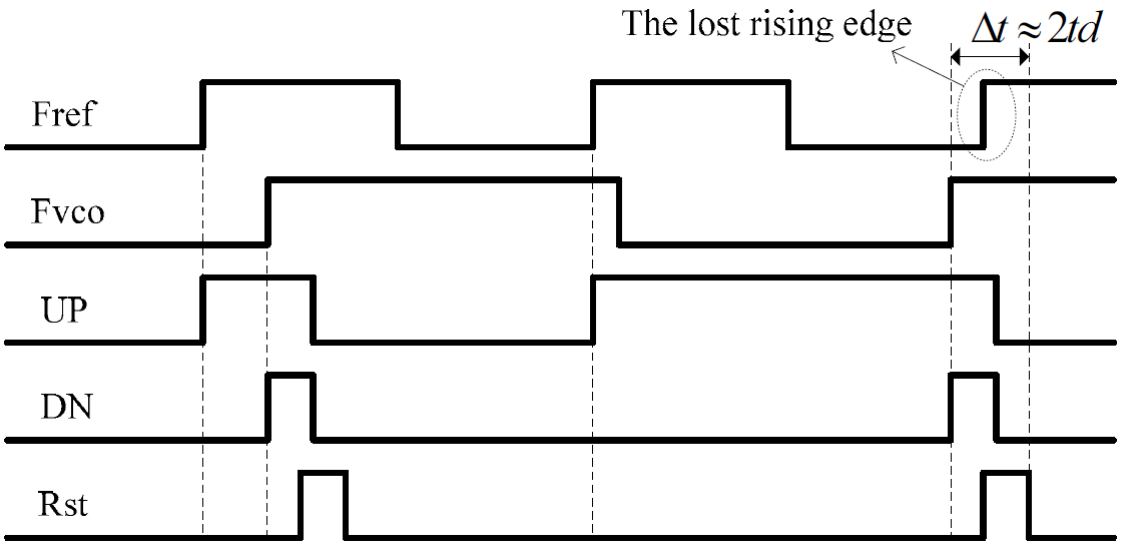

Figure 5.7: The timing diagram of the conventional PFD, (courtesy of ref[10])

by its timing diagram in Fig 5.7. When the input phase error is near 2π, the third rising edge of Fref closely follows the second rising edge of Fvco and it arrives before the Rst signal is deasserted, which makes it undetected by the PFD. Apparently, if the third rising edge arrives during the time slot of $\triangle$t , it will be lost. According to our analysis, the blind zone problem causes the output of the conventional PFD to saturate prematurely when the input phase error approaches 2π, and its actual transfer characteristic is as Fig 5.6(b) shows.

Because of the existence of the blind zone, the actual effective output of the conventional PFD is smaller than expected, which directly slows down the locking process of the PLL. This problem deteriorates when the input frequency Fref gets higher, because now the blind zone takes a larger portion. An intuitive way to alleviate the above problem is to reduce the inserted reset path delay. however, as the minimum delay value is usually determined by the switching speed of the charge pump, this solution can not be always effective. In order to solve the blind zone problem thoroughly, a new PFD is adopted that does not generate the reset signal Rst when the input phase error is beyond the range of [-π, π], resulting in a beneficial non-linear transfer characteristic, which effectively accelerates the locking process of the PLL.

5.2.1 Design principle

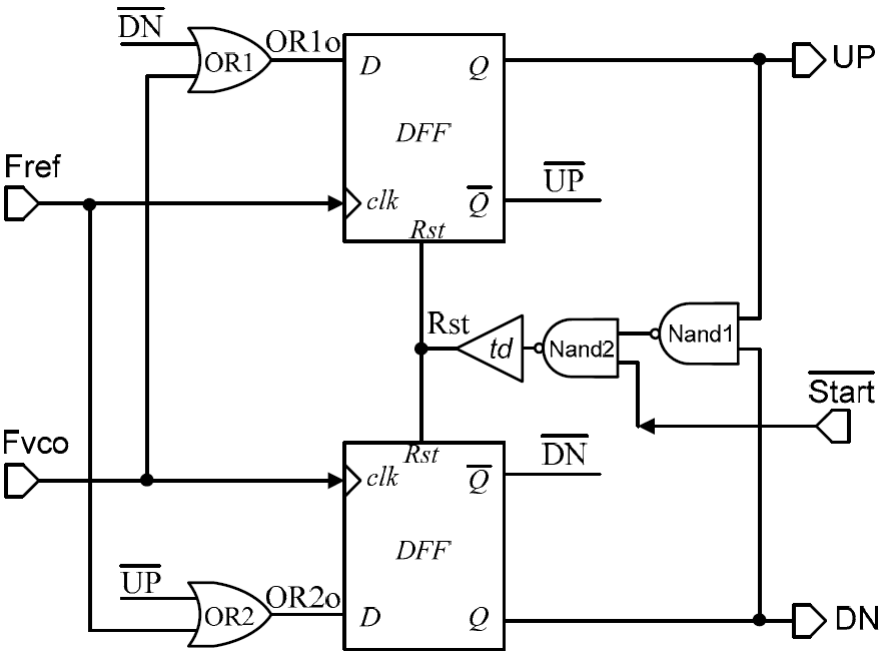

Figure 5.8: The logic schematic of the proposed PFD, (courtesy of ref[10])

The logic schematic of the proposed PFD is presented in Fig 5.8. Obviously, it is an improvement version of the conventional PFD, with two more OR gates and an additional Start signal. In Fig 5.9, the timing diagram of the proposed PFD illustrates how it works

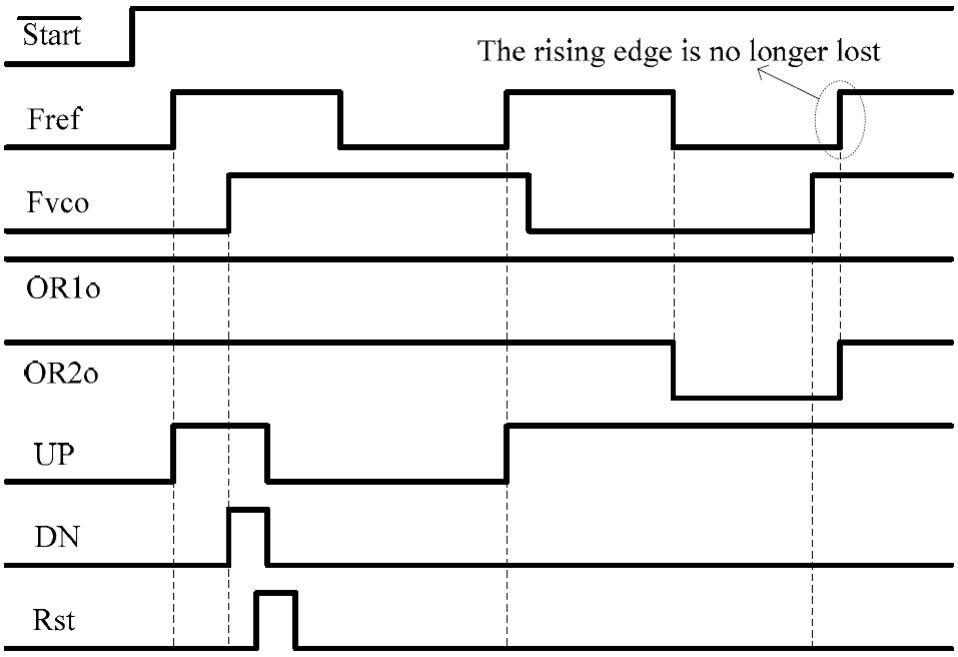

Figure 5.9: The timing diagram of the proposed PFD, (courtesy of ref[10])

to eliminate the blind zone. In the beginning, such as power on, the Start signal is set to logic zero to make the PFD starts from the proper initial condition, which means UP=DN=0, $\bar{UP} = \bar{DN} = 1$ and so OR1o=OR2o=1. When the input phase error is smaller than π, the lagging rising edge of Fvco arrives while Fref=1, and at the same time OR1o=OR2o=1. So in this condition the proposed PFD works in the same way as the conventional PFD. Therefore, it has the same transfer characteristic as the conventional PFD when the input phase error is in the range of [-π, π]. When the input phase error is larger than π, the lagging rising edge of Fvco arrives while Fref=0 and UP=1, which means OR2=0. So in this condition no DN pulse will be generated and consequently no Rst signal will be generated. The blind zone now is eliminated because the rising edge of the Fref is no longer lost by the PFD. Moreover, as there is no DN pulse or Rst signal, the UP pulse keeps logic one until the input phase error decreases to smaller than π. As a result, the proposed PFD maintains a constant output of Tref when the input phase error is out of the range of [-π, π]. In all, it possesses a nonlinear transfer characteristic as shown by Fig.7. The non-linear transfer characteristic of the proposed PFD is beneficial and is often intentionally desired. If a PLL uses the proposed PFD, when the input phase error is out of the range of [-π, π], the PFD works in the constant output region and as a result the locking process can be accelerated; when the PLL is in lock, the PFD works in the linear transfer characteristic region and so a proper loop bandwidth can be maintained for low output jitter.

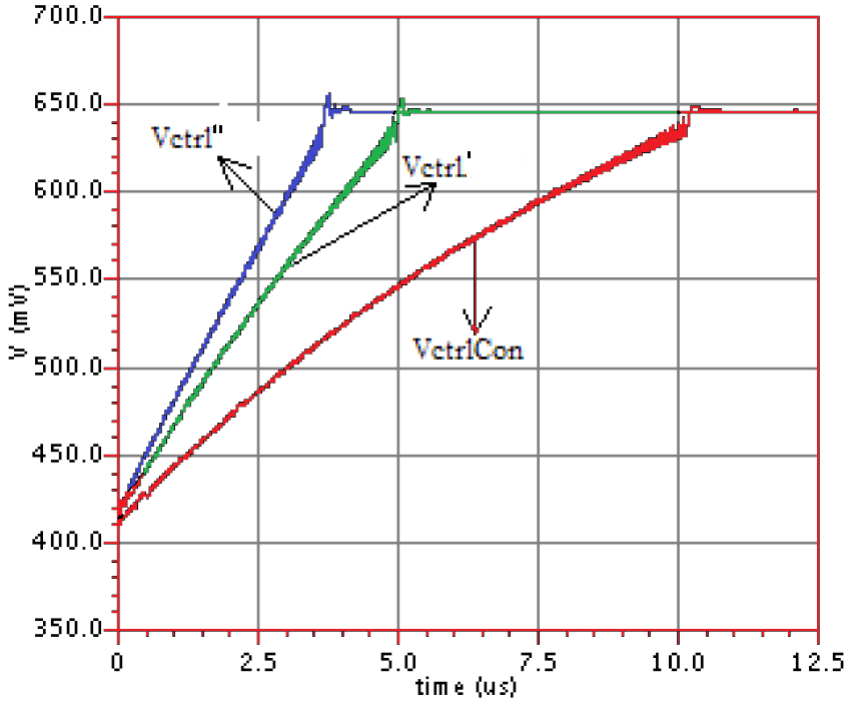

Figure 5.10: The transient response of Vctrl of the test bench PLL when the input reference frequency Fref changes from 600MHz to 800MHz, (courtesy of ref[10])

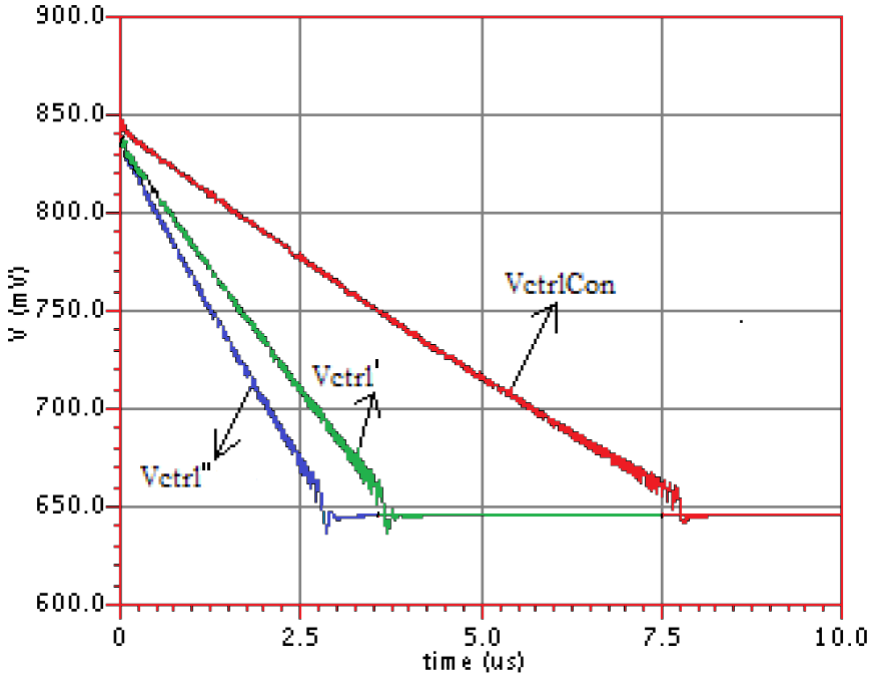

Figure 5.11: The transient response of Vctrl of the test bench PLL when the input reference frequency Fref changes from 1GHz to 800MHz, (courtesy of ref[10])

5.3 Performance Evaluation

In order to evaluate the performance of the proposed PFD, a test bench PLL is set up. Its parameters are: Icp=100μA, Kvco=1GHz/V, R1=400ohms, C1=1nF, C2=80pF, the loop bandwidth of the PLL is about 4MHz.
In fig 5.10 and 5.11, VctrlCon, Vctrl' and Vctrl" respectively indicates the VCO control voltage of the test bench PLL when it uses the conventional PFD, the nonlinear PFD in fig 5.1(b) and the new PFD in fig 5.5. As expected, the non linear PFDs accelerate the locking process of the test bench PLL remarkably, because of their non linear transfer characteristics. It should be noted that the PFD in fig 5.5 is a little more effective than the PFD in fig 5.1(b) in accelerating the locking process of the test bench PLL. The reason for this performance improvement is that when the input phase error is larger than π, the PFD in fig 5.5 does not generate any DN pulse, whereas the PFD in fig 5.1(b) does generate a DN pulse, which offsets some effect of the constant UP pulse.

In Summary, it can be said that non linear PFD which eliminates blind zone can be locked faster than other PFDs. So using this PFD, speed of PLL operation can be increased.

Chapter 6

Non Sequential Phase Frequency Detector

The Phase detectors, described in previous chapters are designed using sequential components. But with existing circuit implementations, the phase detector is often the bottleneck that limits the data rates that can be achieved by the PLL which are speed-limited by sequential logic circuits. The phase detector we present here is specifically designed to operate at higher data rates than what is achievable with the sequential logic circuits. The structure of this new PD is partly determined by the number of stages in the ring oscillator VCO.

6.1 Structure of Non-Sequential Phase Detector used with a VCO with odd number of stages

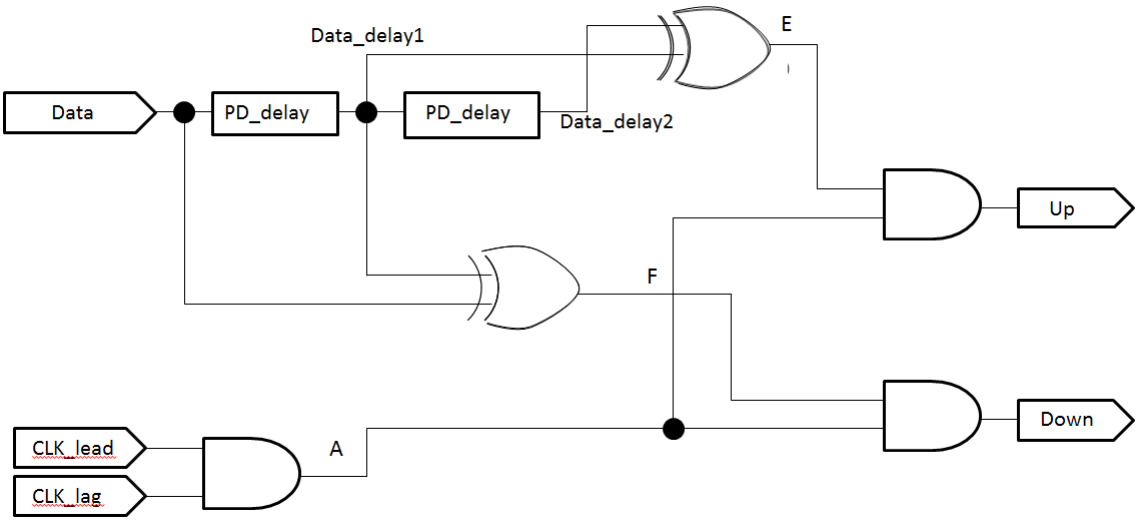

Figure 6.1: Phase Detector based on VCO with odd number of stages, (courtesy of ref[9])

The structure of the PD that is used with an odd number of VCO stages is shown in

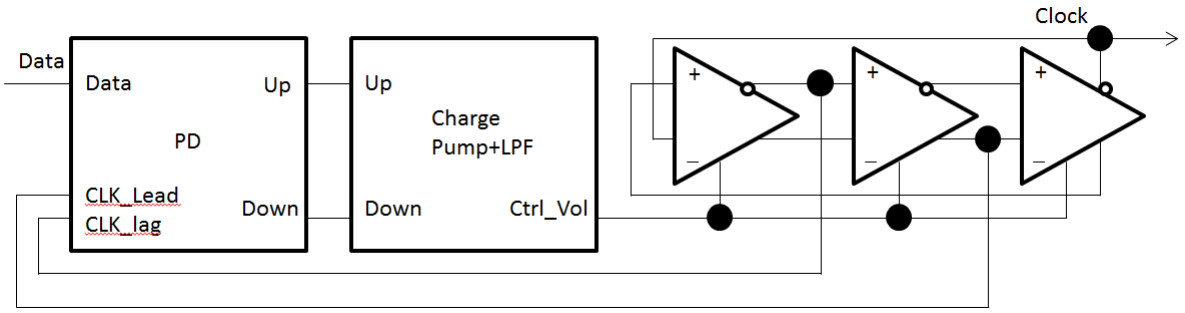

Figure 6.2: The structure of whole PLL, (courtesy of ref[9])

Fig 6.1. The PLL structure that uses this PD is shown in Fig 6.2 for the case where there are 3 delay stages in the VCO. If there are more than 3 delay stages in the VCO, CLK-lead signal can come from the non-inverting output of the stage immediately preceding the clock output stage and CLK-lag signal can come from the inverting output of the immediately following stage. Other signals can also be used for CLK-lead and CLK-lag when there are more than 3 delay stages depending on the PLL design. This PD uses two signals extracted from the VCO. They are CLK-lead and KLK-lag, the leading and inverting lagging signals of the Clock (CLK) signal. We use two delay cells and two XOR gates to detect the edges of the input random data. The Up and Down signals that are required to drive the charge pump are generated from signals A, E and F. Fig 6.3 shows a timing diagram for the situation when the PLL is in lock. The input data is random in ths figure. Signal A is generated from CLK-lead and /CLK-lag signals. It has a fixed

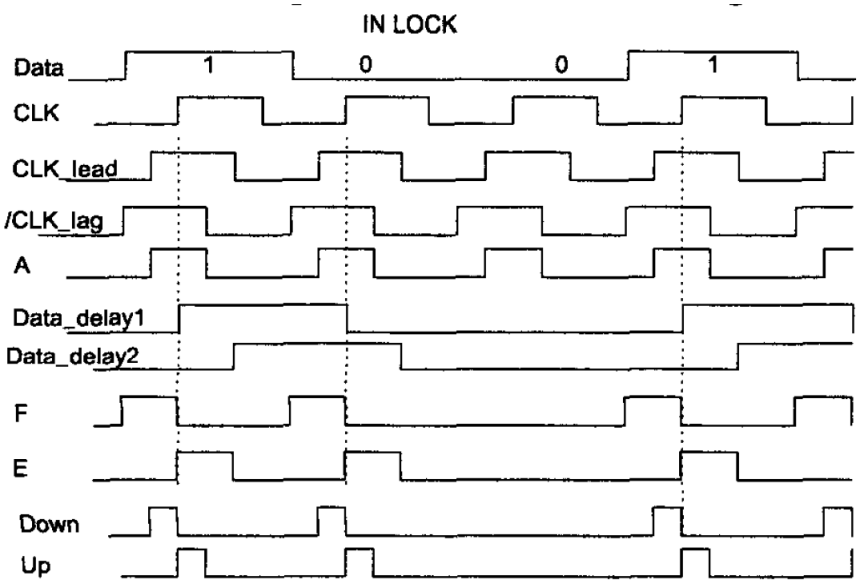

Figure 6.3: The operations of the combinational phase detector, (courtesy of ref[9])

relationship with the regenerated clock, i.e. the rising edges of the CLK are at the middle of signal A. Signals E and F are generated from the signals Data, Data-delay1 and Data-delay2. The falling edges of F and the rising edges of E are aligned at the dotted line,

which, when the PLL is in lock, is right at the middle of the signal A. When the PLL is in lock, Up and Down signals are generated by using E and F to cut signal A in half. Therefore, Up and Down signal have the same duty cycles and hence the loop filter which filters the difference in the duty cycles of the Up and Down signals will not be driven up or down when in lock. From the preceding description, we note that when PLL is in lock, CLK is not locked to data but data_delay1 instead. However this should not affect operation of PLL. The scenarios with leading and lagging Data signals are shown in Fig

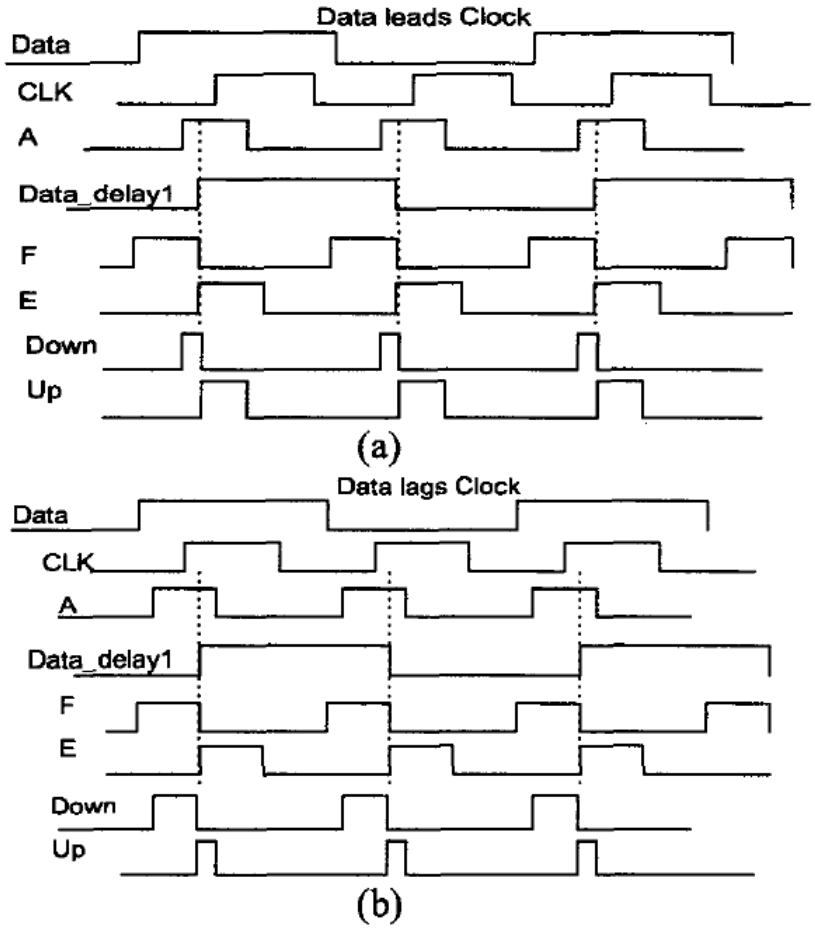

Figure 6.4: (a) Data leads Clock; (b) Data lags Clock, (courtesy of ref[9])

6.4. For simplicity purposes, emphasis is only on how the Up and Down signals relate to the Data signal. When Data is leading the CLK, the dotted line in Fig 6.3 will move to the left relative to signal A. Therefore, the width of the Down pulse will decrease and the width of the Up pulse will increase. Conversely, when Data is lagging the CLK, the dotted line will move to the right relative to signal A. Therefore, the width of the Down pulse will increase and the width of the Up pulse will decrease. The changes in the duty cycles of the Up and Down signals will, in turn, decrease or increase the frequency of the CLK signal through the PLL. In this phase detector, the Up and Down signals are only generated whenever there are transitions on the incoming data stream. This property guarantees its ability to handle random NRZ data. It is well known that many phase

detectors are plagued by dead zone which causes jitter in the PLL. In order to achieve correct operation of the phase detector and eliminate the dead zone in the proposed phase detector, the following constraint on the phase detectors delay stage must be satisfied:

$$max(T_0/2, (T - T_0)/2) < T_{delay} < (T - T_0/2) \tag{6.1}$$

Where T is the period of the signal CLK, To is the pulse width of the signal A, and T_{delay} is the delay of the phase detector's delay stage. Typically, the A signal has a 50% duty cycle.

6.2 Structure of Non-Sequential Phase Detector used with a VCO with even number of stages

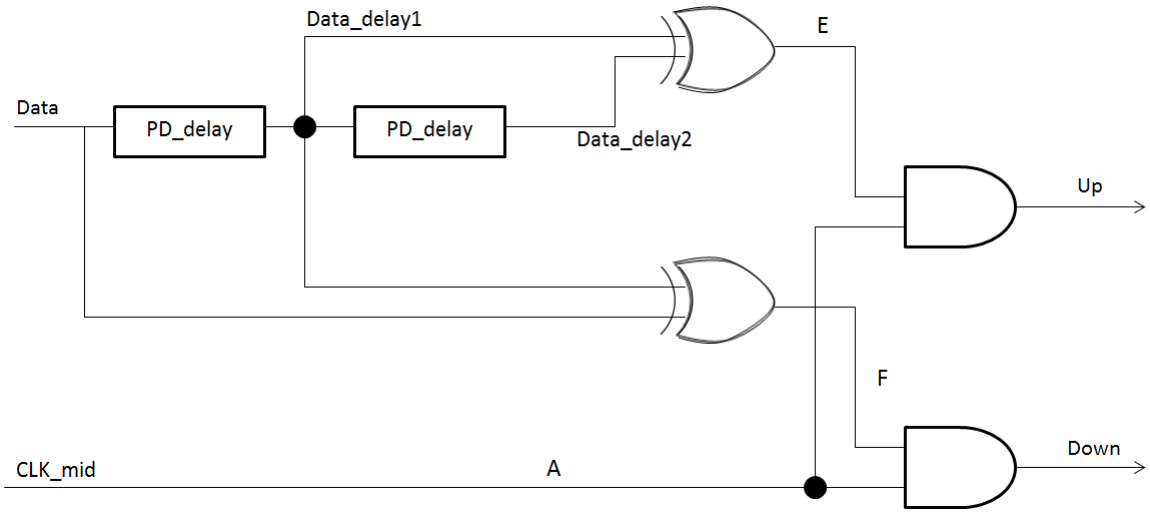

Figure 6.5: Phase Detector based on VCO with an even number of stage, (courtesy of ref[9])

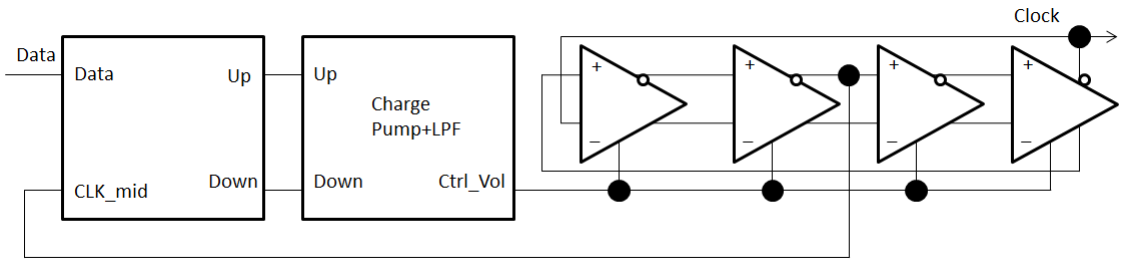

Figure 6.6: The structure of the whole PLL, (courtesy of ref[9])

In where the VCO has an even number of stages, the PD structure becomes even simpler. In this case, we can eliminate the AND gate that is used to generated the signal A in Fig 6.5. Since the signal A can be directly extracted from the VCO. The PD structure is shown in Fig 6.5. A PLL structure that uses this PD for the case where the VCO

has 4 delay stages is shown in Fig 6.6. For more than 4 delay stages in the VCO, the CLK-mid must come from the inverting output of the stage which preceding or following half number of stages to the CLK output stage. The operation principle of this kind of PD is basically the same as described previously when used with a VCO with an odd number of stages.

Finally, it can be concluded that combinational phase detector can be used where PLL speed is limited by its sequential phase detector. It can be operated up to 2 GHz. It also takes care of dead zone. The only condition here is that PLL must use ring VCO.

Chapter 7

Conclusion

7.1 Contributions

Addressing the formal equivalence Verification of specification vs. implementations of PLL, I have contributed for making FEV clean for some of the modules in our project.

7.2 Conclusion

Formal Equivalence Verification is very useful and important for VLSI design flow. It can detect every logic difference between two representation of design. It is also useful to check top level connectivity by FCFEV (Full Chip FEV).
Based on study work of phase frequency detector, it can be concluded that PFD is a very important block of PLL and it determines many characteristics of PLL. Non linear PFD offers several advantages over linear PFD.

Bibliography

[1] Janick Bergeron, "*Functional Verification of HDL Models*" Springer Publication

[2] Junji Kitamichi, Hiroyuki Kageyama and Nobuo Funabiki, "*Formal Verification Method for Combinatorial Circuits at High Level Design*", IEEE Publication, January 1999, vol. 1, pp. 319-322

[3] Jianhua Wang and Yingmei Li, "*Survey on Formal Verification Methods for Digital IC*", IEEE Publication, December 2009, pp. 164-168

[4] Bogdan J. Falkowski, "*Equivalence Checking for Digital Circuits*", IEEE Publication, April 2004, vol. 23, pp. 21-23

[5] B. Meenakshi, "*Formal Verification*", General article, May 2005

[6] J. A. Abraham, "*Verification of Digital Systems*" The University of Texas at Austin, Ch 5

[7] Han-il Lee, Tae-won Ahn, Duck-young Jung and Byeong-ha Park, "*Scheme for No Dead Zone, Fast PFD Design*", Journal of the Korean Physical Society, Vol. 40, No. 4, April 2002, pp. 543-545

[8] Abishek Mann, Amit Karalkar, Lili He, and Morris Jones, "*The Design of A Low-Power Low-Noise Phase Lock Loop*", IEEE Publication, 2010, pp. 528-531

[9] Yonghui Tang, Randall L. Geiger, "*A Non-sequential Phase Detector for PLL-based High-Speed Data/Clock Recovery*", IEEE Publication, 2000, Vol. 1, pp. 428-431

[10] Jinbao Lan, Yuxin Wang, Lintao Liu and Ruzhang Li, "*A Nonlinear Phase Frequency Detector with Zero Blind Zone for Fast-Locking Phase-Locked Loops*", IEEE Publication, July 2010, pp. 41-44

[11] Jinbao Lan, Fengchang Lai, Zhiqiang Gao, Hua Ma and Jianwei Zhang, "*A Nonlinear Phase Frequency Detector for Fast-Lock Phase-Locked Loops*", IEEE Publication, October 2009, pp. 1117-1120

[12] R. Jacob Baker, Harry W. Li and David E. Boyce, "*CMOS Circuit Design, Layout, and Simulation*" IEEE Press Series on Microelectronic Systems, Ch 19