5-2004

# Design and Implementation of an All Digital Phase Locked Loop using a Pulse Output Direct Digital Frequency Synthesizer

Akila Gothandaraman
*University of Tennessee - Knoxville*

To the Graduate Council:

I am submitting herewith a thesis written by Akila Gothandaraman entitled "Design and Implementation of an All Digital Phase Locked Loop using a Pulse Output Direct Digital Frequency Synthesizer." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Syed Kamrul Islam, Major Professor

We have read this thesis and recommend its acceptance:

Michael J. Roberts, Donald W. Bouldin

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Akila Gothandaraman entitled "Design and Implementation of an All Digital Phase Locked Loop using a Pulse Output Direct Digital Frequency Synthesizer." I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Syed Kamrul Islam, Major Professor

We have read this thesis
and recommend its acceptance:

Donald W. Bouldin

Accepted for the Council:

Vice Chancellor and Dean of Graduate Studies

# DESIGN AND IMPLEMENTATION OF AN ALL DIGITAL PHASE LOCKED LOOP USING A PULSE OUTPUT DIRECT DIGITAL FREQUENCY SYNTHESIZER

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Akila Gothandaraman

May 2004

# Acknowledgements

I sincerely thank Dr. Syed Kamrul Islam for providing me an opportunity to pursue my Master's degree in this university and for serving as my advisor as well as the head of this committee. I would also like to thank my thesis committee members, Dr. Donald W. Bouldin and Dr. Michael J. Roberts. The concepts that I learnt in Transform Methods course with Dr. Michael J. Roberts helped me a lot during my thesis work. The courses on ASIC and CAD with Dr. Donald W. Bouldin, where I got exposed to the different CAD tools helped me through out this research. Special thanks to Dr. Benjamin J. Blalock for allocating his time and providing me valuable suggestions. I would also like to thank Dr. Chandra Tan who helped me overcome the problems I encountered in the usage of CAD tools and Stephen Terry for answering my thesis related questions. I would also like to thank Michael Tousek for helping me build the test board for my thesis.

I would also like to thank Analog Devices Inc., where I interned from May 2002-August 2002 in the Signal Synthesis Group. I am thankful to my supervisor, Ken Gentile and all the engineers I worked with at Analog Devices. I would also like to thank Analog Devices Inc., for awarding me the ADI Fellowship which enabled me to go through all levels of design including fabrication and testing.

Finally, I would like to thank my parents, Mrs. Rani Gothandaraman and Mr. Gothandaraman, my sister, Annapoorani Gothandaraman and my fiancé, Balajee Kannan for their continuous support and encouragement.

# Abstract

Phase Locked Loops (PLLs) are widely used in clock recovery and frequency synthesis. Fully Digital PLLs are more suitable for the monolithic implementation with other circuits compared to the traditional implementations of the PLLs. The All Digital PLLs are also independent of process variations and can be easily ported to different technologies.

This thesis presents the design of an All Digital Phase Locked Loop (ADPLL) using a pulse output Direct Digital Frequency Synthesizer (DDFS) and an All Digital Phase Frequency Detector (ADPFD). General design criteria are summarized for the all digital implementation in comparison to the traditional approaches and analog implementations. The design has been fabricated using 0.6-μm CMOS technology. The ADPLL has 16-bit binary control and can operate in the frequency range between 1 MHz and 500 MHz. The ADPLL has 50-cycles lock time and a duty cycle distortion of less than 2%. The simulation and test results of the ADPLL are also presented to verify its operation.

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| APLL | Analog Phase Locked Loop |
| ADPFD | All Digital Phase Frequency Detector |
| ADPLL | All Digital Phase Locked Loop |
| APR | Automatic Place and Route |
| ASIC | Application Specific Integrated Circuit |
| CCO | Current Controlled Oscillator |
| CLK | Clock |
| CMOS | Complementary Metal Oxide Semiconductor |
| CPPLL | Charge Pump Phase Locked Loop |
| CU | Control Unit |
| DAC | Digital to Analog Converter |
| DCD | Duty Cycle Distortion |
| DCO | Digitally Controlled Oscillator |
| DCPLL | Digitally Controlled Phase Locked Loop |
| DDFS | Direct Digital Frequency Synthesizer |
| DDS | Direct Digital Synthesis |
| DPLL | Digital Phase Locked Loop |
| DUT | Device Under Test |
| EXOR | Exclusive OR |
| FF | Flip Flop |
| FIFO | First In First Out |
| FTW | Frequency Tuning Word |
| GCD | Greatest Common Divisor |

| | |
|---|---|
| IC | Integrated Circuit |
| ID | Increment Decrement |
| LPF | Low Pass Filter |
| LPLL | Linear Phase Locked Loop |
| LSB | Least Significant Bit |
| MSB | Most Significant Bit |
| NCO | Numerically Controlled Oscillator |
| NMOS | N Type Metal Oxide Semiconductor |
| NRPD | Nyquist Rate Phase Detector |
| PA | Phase Accumulator |
| PADDFS | Phase Accumulator DDFS |
| PD | Phase Detector |
| PFD | Phase Frequency Detector |
| PLL | Phase Locked Loop |
| PMOS | P Type Metal Oxide Semiconductor |
| ROM | Read only Memory |
| RST | Reset |
| RTL | Register Transfer Level |
| SPLL | Software Phase Locked Loop |
| SRFF | Set Reset Flip Flop |
| STC | Single Time Constant |
| SYNC | Synchronization |
| VCO | Voltage Controlled Oscillator |
| VLSI | Very Large Scale Integration |

# List of Symbols

| | |
|---|---|
| $C_G$ | Gate to source capacitance of the NMOS and PMOS transistors |
| $f_{clk}$ | System clock frequency of Direct Digital Frequency Synthesizer |
| $f_{DCO}$ | Frequency of the Digitally Controlled Oscillator |
| $f_0$ | Free running frequency of the Voltage Controlled Oscillator |
| $f_{osc}$ | Oscillation frequency of the Ring Oscillator |
| $f_s$ | Frequency of the reference or input signal |
| $\omega_2(t)$ | Frequency of the output of the Voltage Controlled Oscillator |
| $I_{PD}$ | Output current of the Charge Pump Phase Frequency Detector |
| $I_{pump}$ | Current of the Charge Pump |
| $K$ | Overall gain of the Phase Locked Loop |
| $K_a$ | Gain of the Active Lag Lead filter |
| $K_M$ | Gain of the Four Quadrant Multiplier Phase Detector |
| $K_{PD}$ | Gain factor of the Phase Detector |
| $K_V$ | Gain of the Voltage Controlled Oscillator |
| $N$ | Register size, Number of delay stages in the Ring Oscillator |
| $S(n)$ | Output of the Phase Accumulator at the nth clock tick |
| $T_d$ | Delay of each stage of the Ring Oscillator |
| $\overline{v_d}$ | Average output of the EXOR Phase Detector |
| $\overline{v_c}$ | Output of the Multiplier Phase Detector |

| | |
|---|---|
| $V_0$ | Output sinusoid of the Voltage Controlled Oscillator |
| $V_{OSC}$ | Oscillation amplitude of each stage in the Ring Oscillator |
| $V_{PD}$ | Output voltage of the Phase Detector |
| $V_s$ | Input sinusoid to the Multiplier Phase Detector |
| $\theta_e$ | Phase shift between the input signal and the output of the VCO |
| $\theta_i$ | Phase of the input signal of the Phase Locked Loop |
| $\theta_0$ | Phase of the output signal of the Phase Locked Loop |
| $v_c(t)$ | Control voltage input of the Voltage Controlled Oscillator |
| $v_0(t)$ | Output signal of the basic Phase Locked Loop |
| $v_s(t)$ | Input signal to the basic Phase Locked Loop |
| $\theta_e(s)$ | Laplace transform of the phase error |
| $\theta_i(s)$ | Laplace transform of the phase of the input |
| $\theta_0(s)$ | Laplace transform of the phase of the output |
| $F(s)$ | Transfer function of the Loop Filter |
| $H(s)$ | Phase transfer Function of the Phase Locked Loop |
| $H_e(s)$ | Error transfer Function of the Phase Locked Loop |
| $V_c(s)$ | Laplace transform of the control voltage of the VCO |
| $V_e(s)$ | Laplace transform of the error signal of the Phase Detector |
| $\tau$ | Time constant of the Resistor-Capacitor in the Loop Filter |

# Chapter 1

# INTRODUCTION AND OVERVIEW

## 1.1 Introduction

Phase Locked Loops (PLLs) are feedback systems that generate signals that are phase locked with external input signals. PLLs are used to generate an output signal whose frequency is a programmable, rational multiple of a fixed input frequency. In other words, PLLs are used to lock or track input signals in frequency and phase. When the phase and frequency of the input signals are synchronized, the PLL is said to be in the locked condition. The phase difference between the output signal and the reference is a known value when the loop is locked.

Phase Locked Loops are used for applications such as clock recovery, motor speed control and frequency control of communications equipment. PLLs are widely used in modern communication systems and high performance microprocessors because of their remarkable versatility. PLLs are also used in frequency modulation, demodulation as well as to regenerate the carrier from an input signal in which the carrier has been suppressed [1]. In communication applications, PLLs are used to make local clocks synchronous with other signals.

Consider a scenario in which two systems with no external handshaking signals are synchronized to the same clock [2]. The two systems may be out of synchronization if the local clocks deviate from each other. Figure 1.1 shows two systems, System A and System B synchronized to a global timing reference, $f_{global}$.

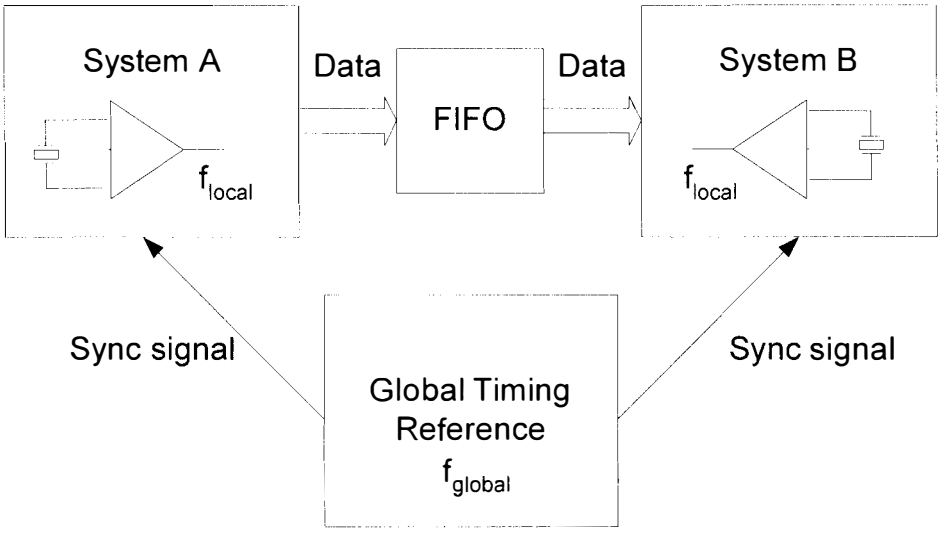



**Figure 1.1:** Two systems synchronized to the same clock

The jitter between the two systems A and B operating at the frequency, $f_{local}$ makes the First in First out (FIFO) buffer fluctuate when transmitting data between the two systems.

Therefore in a complex system made up of many synchronous devices, it is often necessary to distribute the clock to one or more devices. This leads to problems such as clock jitter, synchronization and clock skew. It is probable that the individual devices also operate at different clock frequencies. Thus the Phase Locked Loops find a variety of applications including clock skew/jitter suppression, clock distribution and clock synchronization.

Phase Locked Loops are classified into four different categories based on the implementations; Linear PLL (LPLL), Classical Digital PLL (DPLL), All Digital PLL (ADPLL) and the Software PLL (SPLL) [3].

Moore's law states that the density of integrated circuits doubles every eighteen months [4]. Due to the advances in Very Large Scale Integration (VLSI), Phase Locked Loops often operate in a noisy environment. Currently, PLL design techniques are trying to achieve the following using state of the art design methodologies:

1. Faster lock time
2. Improved jitter performance
3. Robust performance and enhanced stability

## 1.2 Motivation

The conventional PLLs use analog techniques and the classical digital PLLs use a combination of analog and digital techniques to achieve robust operation. However, integrating an analog PLL in a digital noisy environment is difficult. The digital switching noise coupled through the power supply and substrate causes considerable noise in the noise-sensitive analog circuits. The Analog PLL (APLL) is also sensitive to process variations and must be redesigned for each new process. Advances in all digital implementations of the phase locked loops offer enhanced functionality and flexibility.

The All Digital Phase Locked Loop (ADPLL) has better portability and can be used in system on a chip applications and is independent of process variations. It also minimizes the design cost and time. Further, with the use of a Direct Digital Frequency Synthesizer (DDFS), the output frequency, phase and amplitude can be precisely manipulated under digital processor control [13].

This thesis will focus on the design of a PLL using digital methodologies. As the feature size of the VLSI technology continues to shrink, fully integrated digital techniques will be more scalable and portable in many applications than their analog counterparts.

## 1.3 Scope of Thesis

### 1.3.1 ADPLL Requirements

The purpose of this work is to design an All Digital PLL that replaces the blocks of a traditional PLL by digital blocks that can achieve faster lock time, and operate at frequencies of 1 MHz to 500 MHz.

### 1.3.2 Literature Review

An extensive literature search has been performed for papers dealing with the design of Phase Locked Loops. Charge Pump PLLs (CPPLLs) [5] use a Phase Frequency Detector to provide attractive tracking and locking performance. Traditional analog approaches [6]-[9] are proposed to improve the jitter performance of the PLLs however they lead to long lock time and increased design complexity.

Recently, All Digital Phase Locked Loops have gained attention as they yield better testability, programmability, stability and portability [10]-[12] over different processes and can reduce the system turnaround time. The ADPLL proposed in [10] features better resolution, fast lock time, enhanced stability and low-voltage operation; however, the use of a Digitally Controlled Oscillator (DCO) requires it to be custom designed for each new process. The Digitally Controlled PLL (DCPLL) proposed in [11] achieves fast lock time but due to the low sensitivity of the frequency detector and resolution

limitation of the Digital to Analog Converter (DAC), its jitter performance is worse than analog designs. A complete cell-based ADPLL is proposed in [12] to improve the DCO's resolution and output clock jitter effectively; however the delay matrix architecture occupies a large silicon area and has high power consumption.

Table 1.1 presents a summary of the performance characteristics of some analog and all digital phase locked loops that were reported in the literature during the time period of 1995-2002. The PLLs reported use analog, digital or semi digital techniques for the realization of the various building blocks. These PLLs use the Voltage Controlled Oscillator (VCO) or the Digitally Controlled Oscillator (DCO) to realize the oscillator block of the phase locked loop.

**Table 1.1:** Summary of characteristics of reported PLLs

| Performance characteristic | [6] | [10] | [11] | [12] |
|---|---|---|---|---|
| CMOS Process | 0.25-μm | 0.5-μm | 0.6-μm | 0.6-μm |
| Area | 0.09 $mm^2$ | 0.71 $mm^2$ | 0.83 $mm^2$ | 2.75 $mm^2$ |
| Methodology | Analog | All Digital | Semi Digital | All Digital |
| Lock time | < 720 cycles | < 50 cycles | < 16 cycles | < 25 cycles |
| Minimum frequency | 8.5 MHz | 50 MHz | 300 MHz | 360 MHz |
| Maximum frequency | 660 MHz | 550 MHz | 800 MHz | 800 MHz |
| Supply voltage | 1.9 V | 3.3 V | 3.3 V | 3.3 V |

## 1.4 Contributions of Current Work

This research concentrated on designing an All Digital Phase Locked Loop with a pulse output Direct Digital Frequency Synthesizer. The goal of this work was to design a PLL to eliminate the use of analog components and fully digitize its components. The PLL also aims at achieving a faster lock time. After testing is performed, conclusions will be arrived at to determine the performance of the ADPLL and to determine what changes and improvements could be made for performance enhancement. This thesis provides a good starting point for a designer who aims at fully digitizing the components of a PLL using a Direct Digital Frequency Synthesizer.

The design was initially simulated at the behavioral level for testing the functionality of the design. The design was synthesized to obtain the gate-level equivalent that was simulated for testing the timing constraints. Finally, the transistor-level design was also simulated for obtaining the transient response. The design was also fabricated for the 0.6-$\mu$m CMOS technology. In addition to relying on SPICE simulations for testing the working of the design, hardware testing was also given importance in order to validate the effectiveness of the design.

## 1.5 Organization of Thesis

Chapter 2 presents a review of the Phase Locked Loop fundamentals. An overview of the basic building blocks of a Linear PLL is presented. A second order model of a Linear PLL is also presented. Existing approaches such as conventional Charge Pump PLLs and Digitally Controlled Oscillator based approaches are also discussed.

Chapter 3 presents an in-depth look at the architecture, design and implementation of the current research on implementing a fully digital PLL with a pulse output Direct Digital Frequency Synthesizer.

Chapter 4 presents the simulation and measured results of the All Digital Phase Locked Loop that verify some of the robust features of the fully digital implementation.

Chapter 5 presents a summary of the contributions of the research and the conclusions for this work. The objectives accomplished in this thesis will be reviewed and suggestions for future work are also investigated in this chapter.

# Chapter 2

# PHASE LOCKED LOOPS – BACKGROUND

## 2.1 Introduction

Chapter 2 presents a review of the fundamental concepts of Phase Locked Loops (PLLs) and an overview of the All Digital Phase Locked Loops (ADPLLs). Section 2.2 discusses the implementation techniques of the different building blocks of Linear PLLs (LPLLs). Section 2.3 presents the system analysis of a second-order PLL. Section 2.4 presents a brief description of Digital Phase Locked Loops (DPLLs). Section 2.5 presents an overview of All Digital PLLs (ADPLLs) and concludes with a discussion on the implementation methods currently used for the realization of ADPLLs.

Figure 2.1 shows the block diagram of a basic Phase Locked Loop (PLL). The complex nonlinear PLL feedback system is made up of the following blocks: Phase Detector (PD), Low Pass Filter (LPF) and Voltage Controlled Oscillator (VCO). A Divide by N Counter is inserted between the VCO and the PD in frequency synthesis applications. When the counter is used, the VCO generates a frequency that is N times the reference frequency.

Figure 2.2 shows the input and output waveforms when the PLL is locked with zero phase error in the absence of a frequency divider (N = 1). Figure 2.2 also shows the waveforms in the presence of a Divide by 2 Counter in the feedback loop (N = 2). In this case, the frequency of the PLL output ($v_o(t)$) is twice that of the input signal ($v_s(t)$).

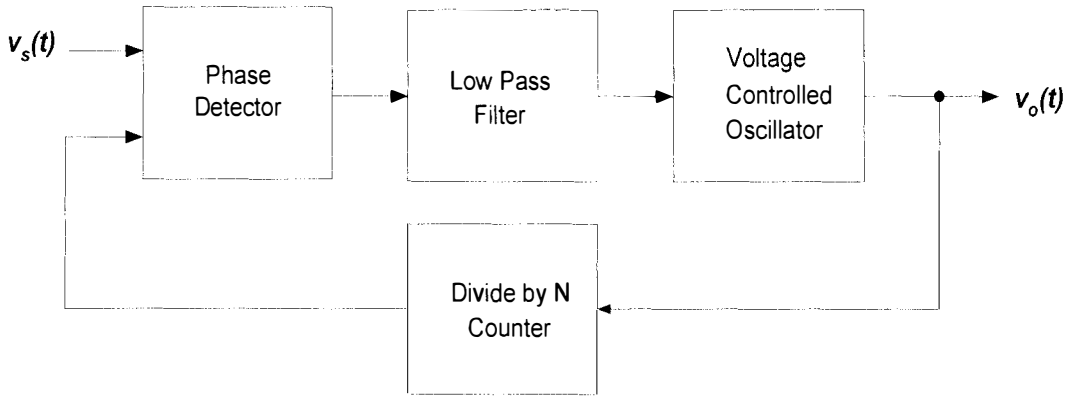



**Figure 2.1:** Block diagram of a basic Phase Locked Loop (PLL)

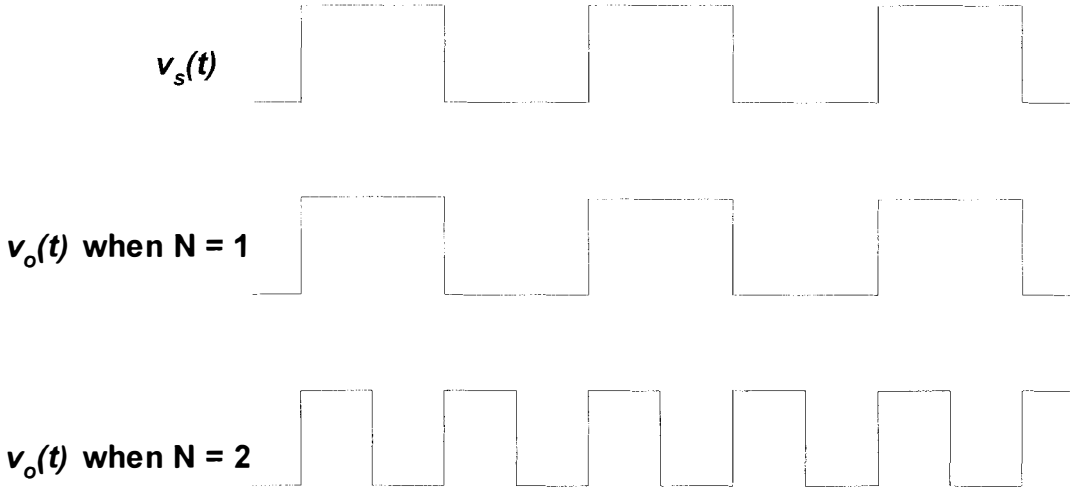



**Figure 2.2:** PLL in locked state

The working of a basic PLL can be summarized as follows; the Phase Detector (PD) compares the frequencies and phases of the input signal ($v_s(t)$) and the output of the Voltage Controlled Oscillator (VCO) ($v_o(t)$). The PD generates an error signal that is proportional to the phase difference between the two signals. The Low Pass Filter (LPF) is used to suppress the sum or the high-frequency component of the error signal. The VCO is a free-running multivibrator and operates at a frequency called the free running frequency ($f_o$). It can be shifted to either side by applying a dc control voltage to the VCO. The low-frequency component from the LPF is applied as the control signal to the VCO. The VCO generates an output signal whose frequency is a linear function of the control voltage. This signal is then fed back to the Phase Detector to achieve zero or constant phase difference between the input signal and the output of the VCO. Once locked, the PLL tracks the frequency changes of the input signal. Some of the commonly used terminologies in a PLL are

1. The *Hold range* is defined as the range of frequencies over which the PLL can maintain lock with the input signal.
2. The *Pull-in range* or *Capture range* of a PLL is the range of frequencies over which the PLL can acquire lock with the input signal.
3. *Pull-in time* or *Lock time* is the total time taken by the PLL to establish the lock.

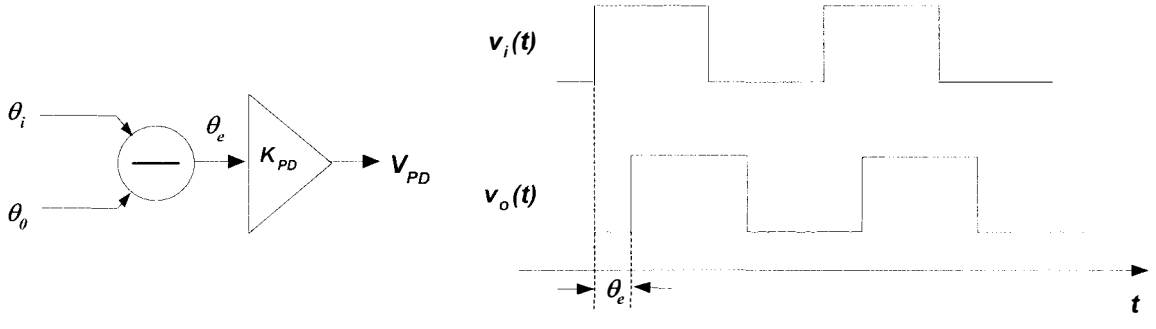
## 2.2 Building Blocks of a PLL

The basic PLL negative feedback system has three blocks, namely Phase Detector, Low Pass Filter and Voltage Controlled Oscillator. The features and implementation methods of these blocks are investigated in this section.

### 2.2.1 Phase Detector

Phase Detectors fall into two broad categories: The first category consists of detectors that are memoryless; their outputs are independent of the previous inputs. These detectors are based on multiplier circuits such as Exclusive OR gate and the Diode Ring Mixer. The second category consists of detectors that require memory because the output is determined using the previous inputs. These detectors are based on sequential circuits constructed from flip-flops and logic gates. The sequential phase detector provides many advantages over the memoryless detectors: (1) Extended Phase Detector range (2) Improved frequency acquisition (3) Insensitivity to input signal levels.

A simple model of a Phase Detector (PD) is shown in Figure 2.3 (a). The PD converts the difference between the phase of the input ($\theta_i$) and the phase of the VCO output ($\theta_0$) into a voltage ($V_{PD}$) with a gain factor ($K_{PD}$). Let $\theta_e$ represent the difference between the input phase and the VCO phase as illustrated in Figure 2.3 (b). The PD produces an output voltage ($V_{PD}$) that is proportional to the phase error ($\theta_e$).




(a) (b)

**Figure 2.3:** Phase Detector **(a)** Model **(b)** Waveform

Equation 2.1 gives the phase difference between the input and output.

$$\theta_e = \theta_i - \theta_0 \tag{2.1}$$

In the linear region, the PD can be modeled by Equation 2.2.

$$V_{PD} = K_{PD}\theta_e \tag{2.2}$$

The range of values of $\theta_e$ for which the linear model is valid is called the Range of the PD. The phase detectors can be analog or digital.

#### 2.2.1.1 Four Quadrant Multiplier Phase Detector

Figure 2.4 shows the block diagram of a Four Quadrant Multiplier PD. The Four Quadrant Multiplier performs the multiplication of the input signal by the output of the VCO. Let the input signal be given by

$$v_s = V_s \sin(\omega_s t) \tag{2.3}$$

and the VCO signal be

$$v_o = V_o \sin(\omega_o t + \theta_e) \tag{2.4}$$

The output of the multiplier is

$$\overline{v_e} = K_M v_s v_o = K_M V_s V_o \sin(\omega_s t)\sin(\omega_o t + \theta_e) \tag{2.5}$$

where $K_M$ is the multiplier constant or the PD gain and $\theta_e$ is the phase shift between the input signal and the output of the VCO.

Equation (2.5) can be simplified to,

$$\overline{v_e} = 0.5K_M V_s V_o[\cos(\omega_s t - \omega_o t - \theta_e) - \cos(\omega_s t + \omega_o t + \theta_e)] \tag{2.6}$$

When the PLL is in the locked condition, $\omega_0 = \omega_s$

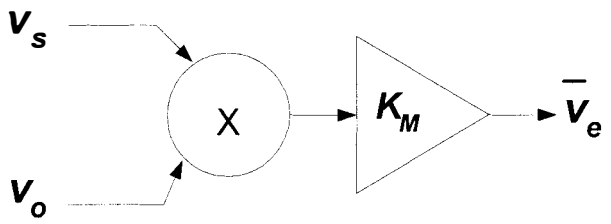



**Figure 2.4:** Block diagram of a Four Quadrant Multiplier PD

Thus,

$$\overline{v_e} = 0.5K_M V_s V_o[\cos(-\theta_e) - \cos(2\omega_o t + \theta_e)] \tag{2.7}$$

The phase comparator output contains a dc term ($0.5KV_sV_o$) and a double frequency term. The second term is filtered using a low pass filter and the dc term is applied as the control input to the VCO. Thus in the locked state ($\omega_s = \omega_o$), the phase difference between the input signal and the VCO output should be 90° ($\theta_e$ = 90°). Equation 2.8 shows that the PD gain for small values of $\theta_e$ is

$$K_{PD} = 0.5K_M V_s V_o \theta_e \tag{2.8}$$

#### 2.2.1.2 Exclusive OR Phase Detector

Figure 2.5 (a) shows the symbol of the Exclusive OR (EXOR) Phase Detector. The EXOR PD is the simplest type of phase detector that uses digital input waveforms. The output of the gate is logic high only when one input goes high and is low for all other inputs. When the input signals are out of phase by exactly 90°, the phase error is zero. When the input signals are symmetrical square waves, the output is a square wave with 50% duty cycle. This condition is depicted in Figure 2.5(b).

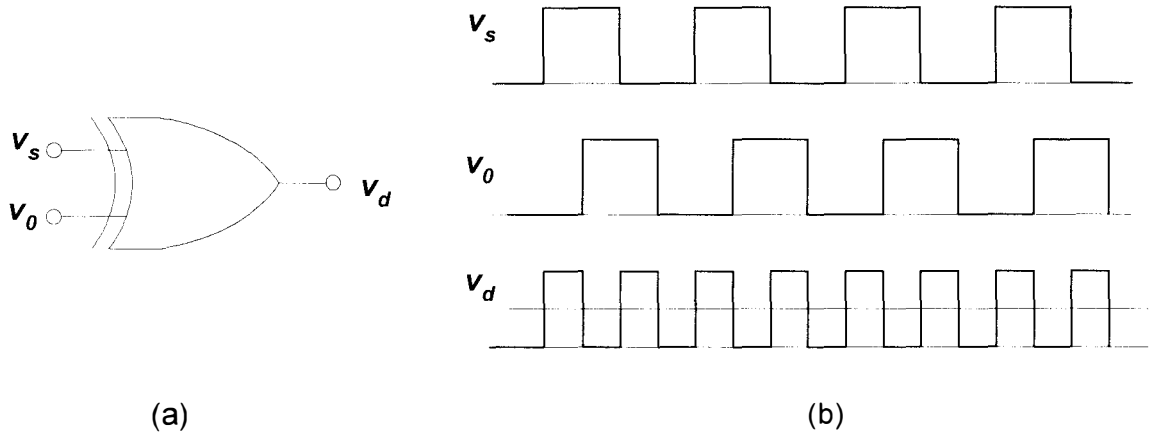



**Figure 2.5:** EXOR PD **(a)** Symbol **(b)** Waveform when $\theta_e = 0$

The output of the Four Quadrant Multiplier varies with the sine of the phase error; the average output of the EXOR is a triangular function of the phase error as shown in Figure 2.6. When $-\pi/2 < \theta_e < \pi/2$, the average output signal is given by Equation 2.9,

$$\overline{v_d} = K_{PD}\theta_e \tag{2.9}$$

When the high and low logic levels are $V_{DD}$ and 0, the phase detector gain ($K_{PD}$) is given by,

$$K_{PD} = \frac{V_{DD}}{\pi} \tag{2.10}$$

### 2.2.1.3 Phase Frequency Detector (PFD)

The Phase Frequency Detectors (PFDs) can be used for both phase and frequency detection. Figure 2.7 shows the schematic of a PFD constructed using two D Flip Flops (FFs) and an AND gate. Table 2.1 shows the state assignment of the PFD. Figure 2.8 shows the state diagram of the PFD.

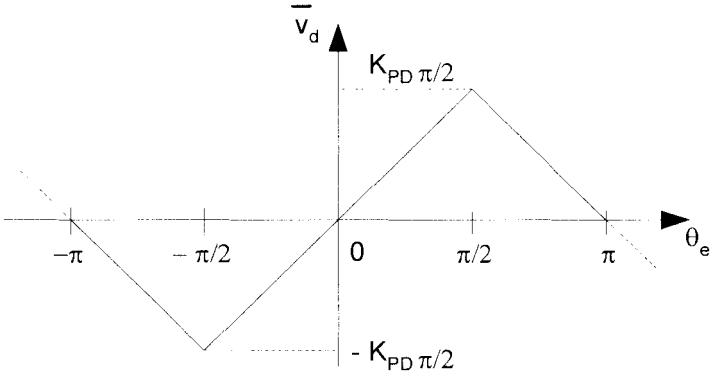
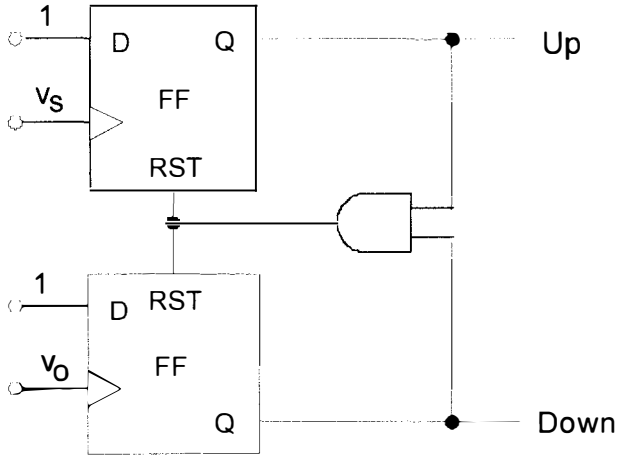



**Figure 2.6:** Characteristics of EXOR PD

**Table 2.1:** State assignment of the PFD




| Up | Down | State |
|---|---|---|
| 0 | 0 | S0 |
| 0 | 1 | S1 |
| 1 | 0 | S2 |

**Figure 2.7:** Block diagram of the PFD

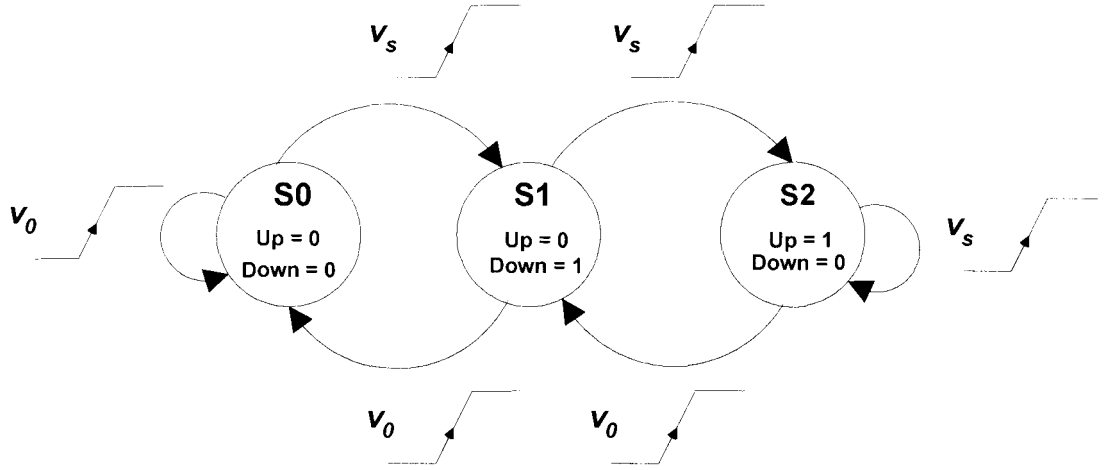



**Figure 2.8:** State diagram of the PFD

The outputs of the PFD are designated as *Up* and *Down*. The PFD can be in one of the four states, *Up* = 0, *Down* = 0; *Up* = 1; *Down* = 0; *Up* = 0, *Down* = 1; *Up* = 1, *Down* = 1. However, the fourth state is prevented using an AND gate which resets the flip-flops. Thus the three allowed states of the PFD are designated as *S0, S1,* and *S2*.

Upon Reset, the PFD enters state, *S0*. The state of the PFD is determined by the positive transitions of the input signals ($v_s$ and $v_0$). The PFD generates the appropriate *Up/Down* signals depending on the current and the previous states. The *Up/Down* pulses are generated proportional to the phase error when the input leads/lags the output. If the loop is in lock, short pulses are generated on the *Up/Down* outputs. The PFD also functions as a frequency detector. Depending on the frequencies of the input and the output, more pulses are generated on the *Up/Down* outputs. When $2\pi < \theta_e < 2\pi$, the average output signal is given by Equation 2.11,

$$v_d = K_{PD}\theta_e \tag{2.11}$$

When the high and low logic levels are $V_{DD}$ and 0, the Phase Detector gain ($K_{PD}$) is given by,

$$K_{PD} = \frac{V_{DD}}{4\pi} \tag{2.12}$$

The characteristics of different phase detectors are investigated and compared in the following section. In an EXOR Phase Detector, the output is averaged or integrated, hence it has good noise rejection. However a drawback of the EXOR Phase Detector is that it may lock to a multiple of the clock frequency. An EXOR DPLL locks on harmonics of the input. If any of the clock inputs is replaced with twice or one half of the frequency, the average of the waveforms will still remain the same. Thus, the EXOR PD declares a "*false lock*". This restricts the operating frequency of the VCO between $2f_s$ and $0.5f_s$ where $f_s$ is the frequency of the reference or the input signal.

The other phase detector discussed was the Phase Frequency Detector (PFD). The PFD overcomes the problem of locking on a harmonic of the data encountered in EXOR PDs. However, the PFD has poor noise rejection. Also, the outputs of the PFD namely *Up* and *Down* have to be converted to a single output which drives the loop filter. Figure 2.9 shows the two ways this can be achieved. Figure 2.9 (a) uses the tri-state output and the configuration in Figure 2.9 (b) is called the Charge Pump. The latter is preferred over the former configuration due to its better immunity to power supply variations. When the signals, *Up* and *Down* are low, the transistors M1 and M2 are off and the output is in the high impedance state. If *Down* signal is high, the output is pulled low through M2. If *Up* signal is high, the output is pulled to *VDD* as M1 is on.

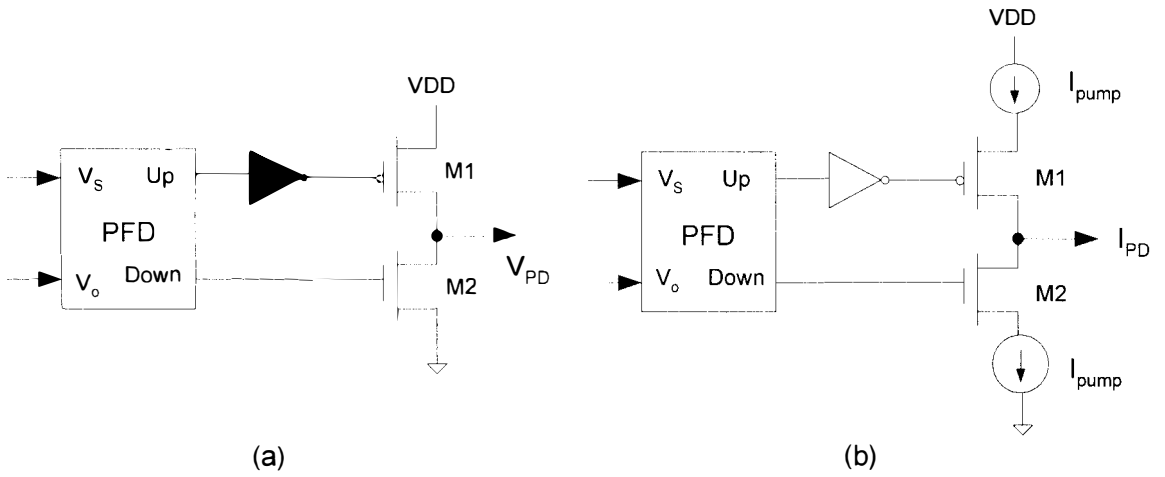



**Figure 2.9:** PFD **(a)** Tri-State Configuration and **(b)** Charge Pump Configuration

In this configuration, the power supply variations can adversely affect the output voltage when M1 is on and thus modulating the VCO control voltage. This can be overcome by using current sources in series with M1 and M2 and making them insensitive to power supply variations [14].

Multiplier type Phase detectors have an advantage when it comes to the frequency of operation. These phase detectors provide good performance when the applied signals are noisy. On the other hand, Sequential Phase detectors are used when the input signals are free from noise and have well-defined level transitions. The PFD will not lock on a harmonic of the clock frequency. Hence the VCO clock duty cycle is irrelevant with the PFD as the PFD performs edge-triggered comparison. However, in communication applications, an EXOR PD might be used due to its good noise rejection.

Table 2.2 summarizes the characteristics of different phase detectors.

**Table 2.2:** Comparison of characteristics of different phase detectors

| Characteristic | Multiplier | XOR | PFD |
|---|---|---|---|
| Phase Detection | Yes | Yes | Yes |
| Frequency Detection | No | No | Yes |
| Sequential (With Memory) | No | No | Yes |
| Detection Range | Small | $\pi$ | $4\pi$ |
| Sensitivity to missing edges | No | No | Yes |
| Locking Phase | $\pi/2$ | $\pi/2$ | 0 |
| Duty Cycle Sensitive | Yes | Yes | No |

### 2.2.2 Loop Filter

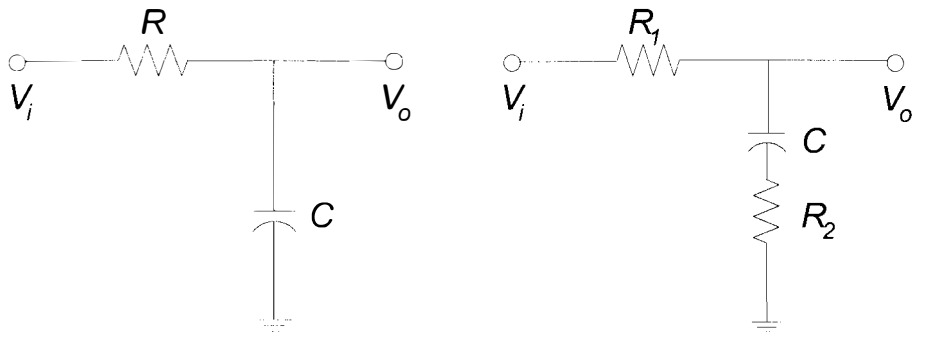
The higher-order frequencies out of the Phase Detector must be filtered out, so the Loop Filter must be of low-pass type. The Loop Filters can be of active type based on operational amplifier technology or passive type using resistor-capacitor networks.

#### 2.2.2.1 Passive Lag Lead Loop Filter

The simplest Low Pass Filter (LPF) can be constructed using RC Single Time Constant network (STC) as shown in Figure 2.10 (a). The RC filter has the transfer function of the form,

$$F(s) = \frac{1}{1 + s\tau} \tag{2.13}$$

where, $\tau = RC$




(a) (b)

**Figure 2.10:** Passive Loop Filter **(a)** First Order RC LPF **(b)** Lag Filter with a zero

A Passive Lag Lead Loop Filter with another resistor $R_2$ in series with the capacitor C can be constructed as shown in Figure 2.10 (b). This filter has one pole and one zero and does not have an integrating function. The filter has a low gain since it is made up of passive elements. The transfer function *F(s)* of the Passive Loop Filter is given by,

$$F(s) = \frac{1 + s\tau_2}{1 + s(\tau_1 + \tau_2)} \tag{2.14}$$

where $\tau_1 = R_1 C \quad and \quad \tau_2 = R_2 C$.

### 2.2.2.2 Active Lag Lead Loop Filter

The Active Filters are made using passive elements as part of the feedback network for a high-gain amplifier. Fig. 2.11 (a) shows an Active Lag Lead filter. Its transfer function contains a gain term, $K_a$. The transfer function *F(s)* of this filter is given by

$$F(s) = K_a \frac{1 + s\tau_2}{1 + s\tau_1} \tag{2.15}$$

where $\tau_1 = R_1 C_1$ , $\tau_2 = R_2 C_2 \quad and \; K_a = -C_1 / C_2$

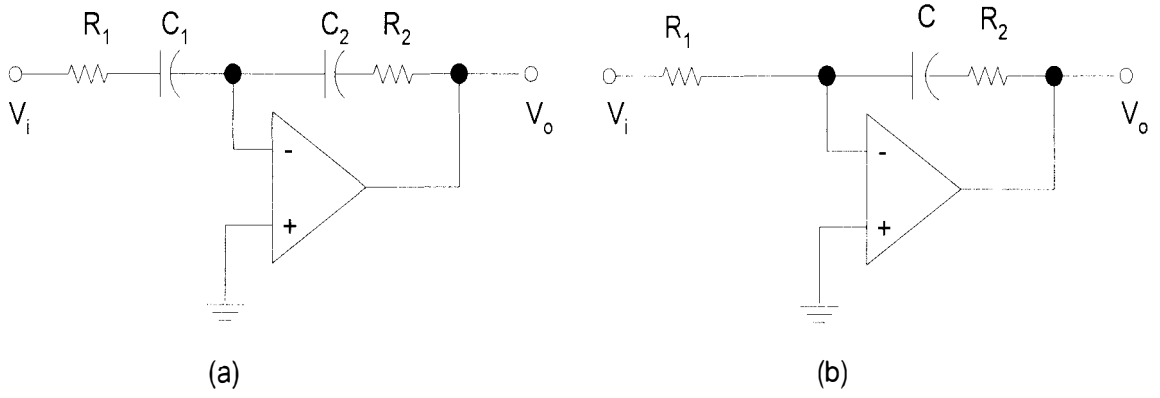



**Figure 2.11:** Active Loop Filter **(a)** Active Lag Filter **(b)** Active PI Filter

Fig. 2.11 (b) shows an active "PI" low pass filter. Equation 2.16 gives the transfer function of the active "PI" filter.

$$F(s) = \frac{1 + s\tau_2}{s\tau_1} \tag{2.16}$$

where $\tau_1 = R_1 C \quad and \quad \tau_2 = R_2 C$

The PI filter has a pole at s = 0 and behaves like an integrator at low frequencies.

### 2.2.3 Voltage Controlled Oscillator (VCO)

VCOs are electronically tunable oscillators in which the output frequency is dependent on the control voltage. A Voltage Controlled Oscillator uses a voltage input to control its frequency in contrast to a Current Controlled Oscillator (CCO) that uses a current input to control its frequency. The choice of a VCO circuit configuration depends on the application. If the VCO frequency ($\omega_2$), is a linear function of the control voltage ($v_c(t)$), it can be given by

$$\omega_2(t) = \omega_0 + \Delta\omega_2(t) = \omega_0 + K_v v_c(t) \tag{2.17}$$

where $\omega_0$ is the free running frequency of the VCO. $K_v$ is the gain factor of the VCO. Figure 2.12 illustrates the frequency versus control voltage characteristics of the VCO.

The VCO is the critical component of a conventional PLL that affects the jitter and phase noise performance. A resonant VCO circuit with an LC tank as the resonant element is known to have excellent jitter performance. However this type of VCO configuration is not suitable for monolithic implementations and requires some off-chip components. On-chip implementations of inductors have been reported in [15] but these have a low quality factor (Q) and are bulky. Ring oscillators are attractive from integration and cost point of view and are being used in jitter sensitive applications [7] [15] [16] [17].

A Ring Oscillator is made up of a number of delay stages. If a cascade of N gain stages with an odd number of inversions is placed in a feedback loop, the circuit oscillates with a period equal to $2NT_d$ where $T_d$ is the delay for each stage with a fanout of one. The schematic of a Ring Oscillator is shown in Figure 2.13. Equation 2.18 gives the frequency of oscillation.

$$f_{OSC} = \frac{1}{2NT_d} \tag{2.18}$$

The factor of 2 in the above equation is due to both rising and falling edges of each delay stage contributing to $f_{OSC}$. The above equation suggests that there are two ways to change the frequency of oscillation: (1) by varying the number of delay stages, $N$ (2) by varying the delay per stage, $T_d$.

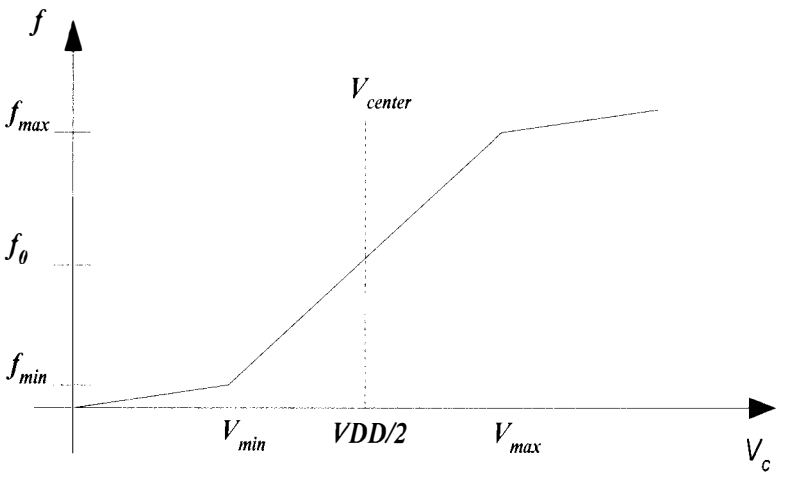



**Figure 2.12:** Characteristics of the VCO

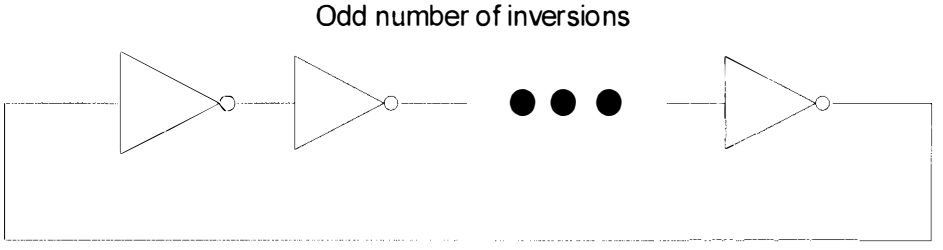



**Figure 2.13:** Ring Oscillator General Schematic

A Conventional Voltage Controlled Ring Oscillator and its delay approximation are shown in Figure 2.14 [23]. Here, N is an odd number of inversions. Assume that the gate to source parasitic capacitances, $C_G$ of the NMOS and PMOS transistors are equal. The frequency of oscillation is given by

$$f_{osc} = \frac{1}{2N\tau} \tag{2.19}$$

where $\tau$ is the delay of one inverter stage. The oscillation amplitude of each inverter stage is given by Equation 2.20,

$$V_{osc} = \int \frac{I_{ctrl}}{C_G} dt \tag{2.20}$$

From the above equation, the delay of the inverter stage can be given by Equation 2.21

$$\tau = \frac{V_{osc} C_G}{I_{ctrl}} \tag{2.21}$$

where $V_{osc}$ is the oscillation amplitude.

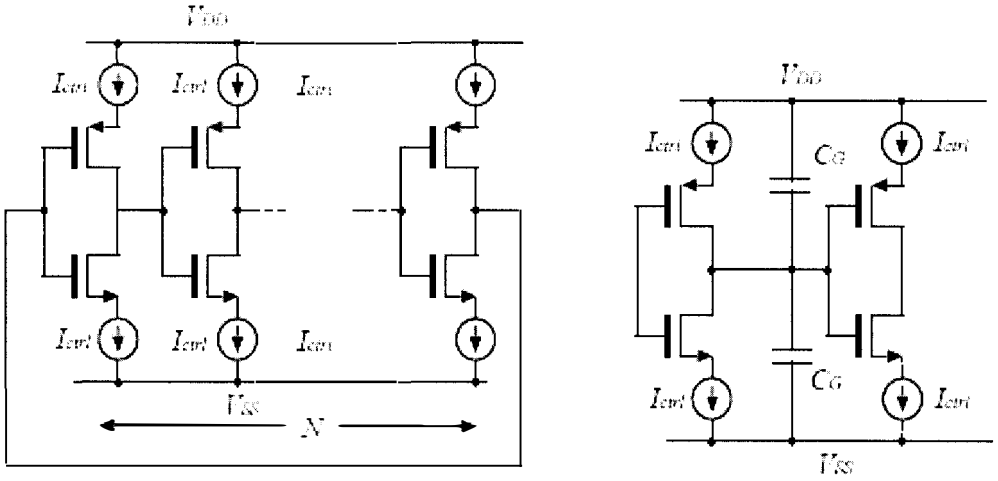



**Figure 2.14:** A Conventional Voltage Controlled Ring Oscillator

The expression for oscillation frequency, $f_{osc}$ can now be given by

$$f_{osc} = \frac{I_{ctrl}}{2NV_{osc}C_G} \tag{2.22}$$

The oscillation frequency depends on the current ($I_{ctrl}$), the number of stages ($N$), the parasitic capacitance ($C_G$) and the oscillation amplitude. Thus, the oscillation frequency can be varied by varying control current or amplitude, as $C_G$ is a fixed parameter. The current is usually the controlling factor. The oscillation frequency can be tuned for a wide range by changing the value of control current. However, it is difficult to keep the matching between the upper and lower control currents for very small currents.

A modified Voltage Controlled Oscillator topology with wide tuning range and fast voltage swing is reported in [23]. The design implements a low frequency ring oscillator with relatively smaller devices and fewer stages.

## 2.3 System Analysis of a Second-Order PLL

The three building blocks namely the Phase Detector, Low Pass Filter and VCO can be put together and a linear mathematical model for the system can be developed. The mathematical model will be used to derive the phase transfer function ($H(s)$) which relates the phase of the input signal $(\theta_1)$ to the phase of the output signal $(\theta_2)$:

$$H(s) = \frac{\theta_2(s)}{\theta_1(s)} = \frac{KF(s)}{s + KF(s)} \tag{2.23}$$

where $\theta_2(s)$ and $\theta_1(s)$ are the Laplace Transforms of the phases of the output and the input respectively. $F(s)$ is the transfer function of the Loop Filter. $K$ is the loop gain of

the PLL and is given by $K = K_{PD} K_v$. Figure 2.15 shows the linear model of the PLL. The output signal from the Four Quadrant Multiplier Phase Detector is given by

$$v_e(t) = K_{PD} \sin\theta_e \tag{2.24}$$

When the phase error is small,

$$v_e(t) \cong K_{PD}\,\theta_e \tag{2.25}$$

$K_{PD}$ represents the Phase Detector gain. Equation 2.25 represents the linear model of the Phase Detector. Thus a Four Quadrant Multiplier used in a Linear PLL, which is in the locked state, represents a zero-order block of gain $K_{PD}$. Laplace Transform of Equation 2.25 yields,

$$V_e(s) = K_{PD}\theta_e(s) \tag{2.26}$$

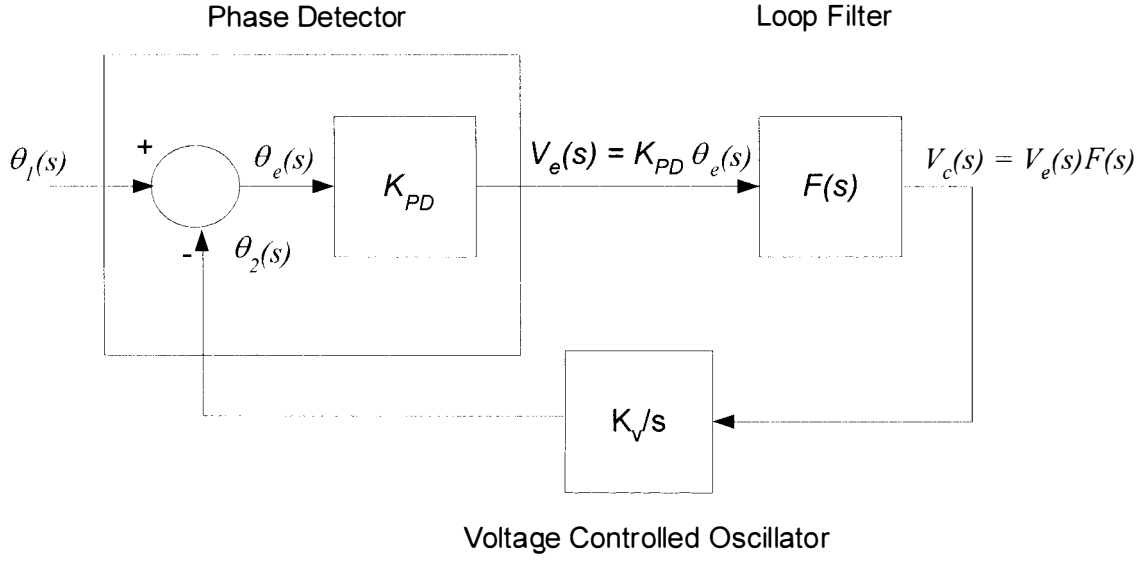



**Figure 2.15:** Linear model of the Phase Locked Loop

Table 2.3 presents a summary of the Transfer Functions, $F(s)$ for the different loop filters discussed in Section 2.2.2.

The output of the Low Pass Filter is applied as the control input to the VCO. The control voltage, $v_c(t)$ shifts the VCO frequency from its free running frequency, $\omega_0$ to a frequency $\omega_2$ given by,

$$\omega_2(t) = \omega_0 + \Delta\omega_2(t) = \omega_0 + K_v v_c(t) \tag{2.27}$$

The phase of the VCO, $\theta_2$ is given by the integral over the frequency variation, $\Delta\omega_2(t)$ as shown in Equation 2.28,

$$\theta_2(t) = \int \Delta\omega_2(t)dt = K_v \int v_c(t)dt \tag{2.28}$$

**Table 2.3:** Summary of the Transfer Functions of the Loop Filters

| Type of Loop Filter | Transfer Function, $F(s)$ |
|---|---|
| Passive Lag Filter | $F(s) = \frac{1}{1+s\tau}, \tau = RC$ |
| Passive Lag Filter with a zero | $F(s) = \frac{1+s\tau_2}{1+s(\tau_1+\tau_2)}, \tau_1 = R_1C \text{ and } \tau_2 = R_2C$ |
| Active Lag Lead Filter | $F(s) = K_a \frac{1+s\tau_2}{1+s\tau_1}, \tau_1 = R_1C_1, \tau_2 = R_2C_2$ and $K_a = -\frac{C_2}{C_1}$ |
| Active PI Filter | $F(s) = \frac{1+s\tau_2}{s\tau_1}, \tau_1 = R_1C \text{ and } \tau_2 = R_2C$ |

Laplace Transform of an integral over time is same as division by s, so the Laplace transform of Equation 2.28 yields

$$\theta_2(s) = \frac{K_v}{s} V_c(s) \tag{2.29}$$

The Transfer Function of the VCO is given by Equation 2.30.

$$\frac{\theta_2(s)}{V_c(s)} = \frac{K_v}{s} \tag{2.30}$$

The Laplace Transform of the error signal, $\theta_e(s)$ is given by Equation 2.31.

$$\theta_e(s) = \theta_1(s) - \theta_2(s) \tag{2.31}$$

The Phase Transfer Function, $H(s)$ is given by

$$H(s) = \frac{\theta_2(s)}{\theta_1(s)} = \frac{\theta_2(s)}{V_c(s)} * \frac{V_c(s)}{V_e(s)} * \frac{V_e(s)}{\theta_e(s)} * \frac{\theta_e(s)}{\theta_1(s)} \tag{2.32}$$

The Phase Transfer Function can therefore be expressed as

$$H(s) = \frac{K_v K_{PD} F(s)}{s + K_v K_{PD} F(s)} \tag{2.33}$$

In addition to the Phase Transfer Function, an Error Transfer Function, $H_e(s)$ can also be defined which is given by

$$H_e(s) = \frac{\theta_e(s)}{\theta_1(s)} = \frac{s}{s + K_v K_{PD} F(s)} \tag{2.34}$$

## 2.4 Digital Phase Locked Loop (DPLL)

The block diagram of a DPLL is shown in Figure 2.16. The difference between the DPLL and LPLL lies in the implementation of the Phase Detector. The Phase Detector in a DPLL can be implemented using the Digital Phase Detectors namely the EXOR gate, JK Flip Flop or using a Phase Frequency Detector (PFD).

## 2.5 All Digital Phase Locked Loops

### 2.5.1 Introduction

The All Digital Phase Locked Loop (ADPLL) aims at eliminating all the analog circuitry. In contrast to the Digital Phase Locked Loop (DPLL), it is "fully digital". The all digital phase locked loops offer better scalability and portability. Existing standard cell implementations can be used for the all digital PLL blocks. The term "digital" in an ADPLL signifies two things: Firstly, the components of an ADPLL are entirely digital.

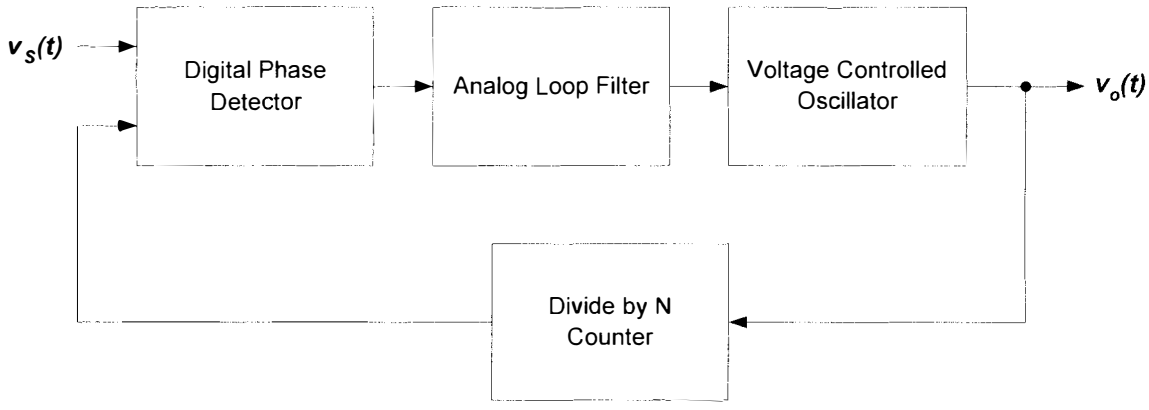



**Figure 2.16:** Block Diagram of a Digital Phase Locked Loop (DPLL)

Secondly, the signals associated with the different blocks are also digital and could be binary or bit signals or word signals such as the digital code word output of a data register or n-bit output word of a counter [3].

ADPLLs offer some advantages compared to Analog or Charge Pump Phase Locked Loops with respect to simplicity and flexibility of the design. But these loops also have some limitations when it comes to characteristics such as jitter performance or locking range. The goal of this work was to design a fully Digital Phase Locked Loop that can match or exceed the performance of the Analog PLLs. Some ADPLLs may also use a Digitally Controlled Oscillator (DCO) in contrast to its analog counterpart: the Voltage Controlled Oscillator (VCO). The following section explores some of the possible digitized phase detectors, loop filters and DCO implementations used in ADPLLs.

### 2.5.2 Basic Building Blocks of an ADPLL

#### 2.5.2.1 All Digital Phase Detector

The Digital Phase Detectors discussed in Section 2.2.1 can also be extended for ADPLLs when digital word signals instead of bit signals are used. A Flip Flop (FF) counter based PFD uses an edge triggered SR Flip Flop in combination with a counter [3]. The output of the FF is used to gate the high frequency clock signal into the counter. The content of the counter is proportional to the phase error ($\theta_e$).

Another phase detector is the Nyquist Rate Phase Detector (NRPD), which is used if the input is an analog signal [3]. The analog signal is converted into a digital signal using an Analog to Digital Converter (ADC) sampled at a rate greater than the Nyquist rate. The name stems from the Nyquist theorem, which states that a sampled signal

can be reconstructed only when the sampling rate is more than twice the highest frequency component of the signal. Other kinds of PD include the Hilbert Transform Phase Detector and the Digital Averaging Phase Detector discussed in [3]. These are suitable for Software PLL (SPLL) implementations.

#### 2.5.2.2 Digital Loop Filter

The all digital implementations of the loop filter do not have passive elements such as resistors or capacitors. The UP/DOWN Counter is the simplest type of loop filter, which is used in conjunction with the Phase Frequency Detector (PFD) [3]. The PFD delivers *Up/Down* pulses based on the comparison between frequencies of the input and the output. These pulses are used to increment or decrement a counter. The *Up/Down* pulses do not contain information about the magnitude of the phase error. The *Up/Down* pulses are a result of the relative comparison of the two signals.

A K Counter Loop filter [3] [18] used in 74HC297 uses two counters, UP and DOWN Counters to generate *Carry* and *Borrow* Signals. The $\overline{Up} / Down$ signal controls the operation of the two counters. If the $\overline{Up} / Down$ signal is active "high", the DOWN counter is active and if the signal is active "low", the UP counter is active.

Another digital loop filter is the N-before-M counter [3]. It operates in conjunction with a Phase Detector generating *Up/Down* pulses such as a Phase Frequency Detector. The N-before-M counter uses two counters, the ÷M counter and the ÷N counter. The *Carry* pulse is generated when a majority of N pulses have been *Up* pulses. Similarly, the

*Borrow* pulse is generated when a majority of N pulses have been *Down* pulses. (M > N always).

### 2.5.2.3 Digitally Controlled Oscillator (DCO)

The DCO is the digital counterpart of the Voltage Controlled Oscillator (VCO) in fully Digital Phase Locked Loops. The ÷N counter is used to scale down the signal generated by a fixed high frequency oscillator ($f_{REF}$). The N-bit output of the Digital Loop filter controls the operation of the DCO. The output of the DCO is given by Equation 2.35.

$$f_{DCO} = \frac{f_{REF}}{N} \tag{2.35}$$

Another popular DCO mechanism called the Increment-Decrement (ID) counter DCO is proposed in [18] which is a modified form of the DCO used in 74HC97 ADPLL. The DCO works in conjunction with a K-Counter Loop Filter or an N-before-M filter that generates the *Carry* or *Borrow* signals.

Several DCO approaches have also been proposed in [19] [24] [10]. The first type of DCO is the "Path Delay Oscillator". The Path Delay Oscillator proposed in [19] contains logic gates to form a closed ring oscillator with an odd number of inversions. The PLL achieves lock by using two tuning mechanisms namely the Coarse and the Fine-Tuning mechanisms. The Coarse Tuning mechanism operates by activating paths such that the frequency of the DCO just exceeds the frequency of the reference clock. When this occurs, the Fine-Tuning mechanism takes control of the lock process to achieve

fine resolution. However, the frequency of the Path Delay Oscillator is very limited and is not suitable for high frequency requirements.

The second type of the DCO is the "Schmitt-Trigger Based Current-Driven Oscillator" [24]. A Schmitt-Trigger inverter together with a big capacitor is used to realize the oscillator. The capacitor is usually connected off-chip due to its large value. The external capacitance may make the system complicated and degrade the performance.

The third category is the "Current-Starved Ring Oscillator" to design the DCO [10]. This DCO has good linearity and implemented by using different MOS switches to get different frequencies. However, the DCO is large and needs a lot of hardware. Another approach similar to the current starved technique has been proposed in [20] that uses less hardware and saves half the layout area of the conventional DCO in [10].

#### 2.5.2.4 Design Criteria

Using more bits for the DCO provides a wider range; however this comes at the cost of increased area, more power dissipation. The lock time of the ADPLLs depends on the implementation. Full Custom design is not required for the ADPLLs. The synthesis tools can be used to simplify the design process. The Integrated Circuit (IC) layout and simulation are also simpler compared to Analog Loops. The design details of the ADPLL using the Direct Digital Synthesis (DDS) techniques will be discussed in detail in the following chapter.

# Chapter 3

# All Digital Phase Locked Loops

## 3.1 Introduction

All Digital Phase Locked Loops (ADPLLs) are more attractive than the Analog PLLs since they offer better flexibility and portability. For portable or mobile applications, lock time is very important since the PLL must support fast exit and entry from power management techniques. The ADPLLs have the characteristic of fast frequency locking, full digitization and good stability. Chapter 2 discussed some of the implementations of the ADPLLs reported in [10] [21] [22] [23] using Digitally Controlled Oscillator (DCO) techniques. This chapter describes an All Digital Phase Locked Loop using a pulse output Direct Digital Frequency Synthesizer (DDFS). Section 3.2 presents the building blocks of the ADPLL. Some target specifications for this ADPLL include

1. Fast Lock Time
2. Frequency and Phase Locking and Wide Operating Range
3. Reduced Silicon Area

## 3.2 Building Blocks of the ADPLL

Figure 3.1 depicts the block diagram of an All Digital Phase Locked Loop. The ADPLL consists of the following blocks: Phase Frequency Detector (PFD), Control Unit (CU) and the pulse output Direct Digital Frequency Synthesizer. The PFD compares the phases and frequencies of the reference signal (*REF_Clk*) and the Most Significant Bit

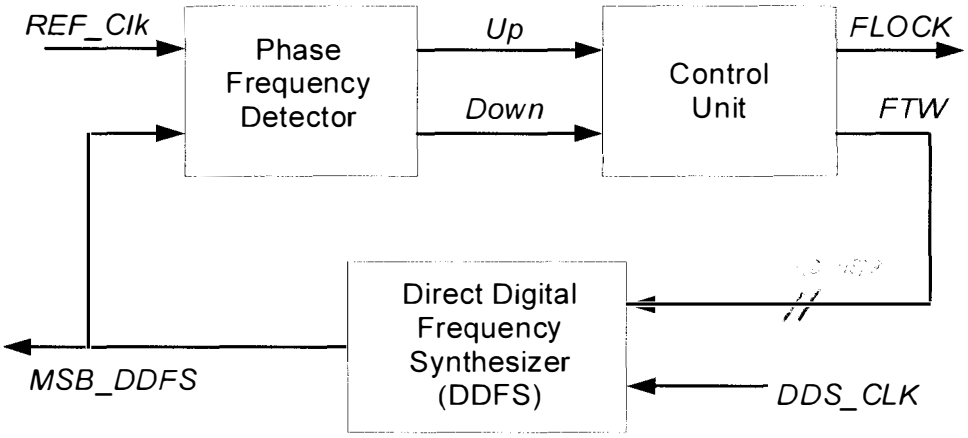



**Figure 3.1:** Block Diagram of an All Digital Phase Locked Loop (ADPLL)

(MSB) output of the Direct Digital Frequency Synthesizer (*MSB_DDFS)*. The Phase Frequency Detector uses a search algorithm to sweep the frequency of the reference signal to match it with that of the output of the DDFS. One of the two signals, *Up* or *Down* is asserted after each phase - frequency comparison. The Control Unit changes the Frequency Tuning Word (*FTW*) based on the output of the PFD.

The pulse output Direct Digital Frequency Synthesizer forms the core of the ADPLL. The output frequency of the DDFS depends on the Frequency Tuning Word and the System Clock, *DDS_CLK*. When the *REF_Clk* and *MSB_DDFS* signals are locked, the Control Unit asserts the *FLOCK* signal.

The output of the Phase Frequency Detector determines whether the Frequency Tuning word has to be incremented or decremented. Arithmetically incrementing or decrementing the Tuning Word modulates the DDFS Frequency. Two Gain registers, *ADD_GAIN* and *SUB_GAIN* registers are also used in conjunction with the Tuning Word register to determine the add/subtract operation on the tuning word.

### 3.2.1 All Digital Phase Frequency Detector

The All Digital Phase Frequency Detector (ADPFD) compares the phases and frequencies of the *REF_Clk* and *MSB_DDFS* respectively. The block diagram of the Phase Frequency Detector is shown in Figure 3.2. The ADPFD consist of two counters; The REF CLOCK Counter counts the positive edges of the reference clock. The MSB of the DDFS is stored as a binary signal which changes to "1" or "0" at each zero crossing of the DDFS output. The DDFS Counter counts the positive edges of this DDFS binary output. The ADPFD compares the counts over a fixed time window. Based on the relative comparison of the contents of the two counters, the ADPFD asserts the *Up* or *Down* Signal. The Control Unit uses an algorithm that sweeps the DDFS frequency in progressively smaller increments to match it with that of the reference clock. The algorithm begins by initializing the *FTW* register, *ADD_GAIN* and *SUB_GAIN* registers. The algorithm changes the tuning word based on the output of the phase frequency detector. The values held in the gain registers determine the magnitude of the changes.

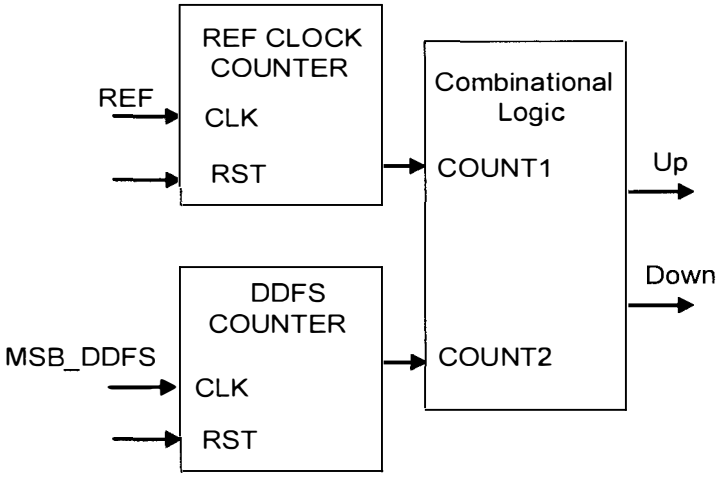



**Figure 3.2:** Block Diagram of the All Digital Phase Frequency Detector

### 3.2.2 Frequency Synthesizer

#### 3.2.2.1 Introduction

Frequency Synthesizers are electronic devices that accept some reference frequency and generate or synthesize new frequencies as commanded by an input control word so that the stability, accuracy and spectral purity of the output correlates with the performance of the reference input. The block diagram of a Frequency Synthesizer is shown in Figure 3.3.

Three conventional techniques are available by which this function can be accomplished: Indirect, Direct Analog and Direct Digital. Indirect Synthesizers use Phase Locked Loop techniques to multiply an input reference with moderate output frequency range and step sizes. Direct Analog Frequency Synthesizers use multiplication, division and mathematical manipulations to produce the desired frequency. In these synthesizers, the error correction process is avoided and hence the quality of the output correlates directly with the quality of the input and hence called Direct Synthesizers. Direct Digital Frequency Synthesizers use digital techniques to generate lower frequencies with small step sizes.


**Figure 3.3:** Block Diagram of a Frequency Synthesizer

Direct Digital Synthesis (DDS) has become attractive over other synthesis techniques and various designs and architectures have been proposed according to the requirements of the applications. The basic principles of the DDS technology have been outlined in [22] [13]. A standard DDS uses an analog part: a Digital to Analog Converter (DAC). Goldberg in [22] has presented a few classical approaches to all digital DDS and standard DDS using a DAC.

#### 3.2.2.2 Direct Digital Frequency Synthesizer

The block diagram of a conventional Direct Digital Frequency Synthesizer is shown in Figure 3.4. The Direct Digital Frequency Synthesizer also sometimes referred to as Numerically Controlled Oscillator (NCO) has the following blocks; a Phase Accumulator (PA), a Phase to Amplitude Converter (conventionally a sine ROM), a Digital to Analog Converter (DAC) and a Filter.

The Phase Accumulator consists of a *j*-bit frequency register that stores a digital phase increment word followed by a phase register and a *j*-bit full adder. The digital phase increment word is entered into the frequency register. At each clock pulse, this word is added to the previously held data in the phase register. The phase value is generated using the modulo $2^j$ overflowing property of a *j*-bit phase accumulator. The contents of the Phase Accumulator are used to access a Read Only Memory (ROM) which converts the phase information into corresponding amplitudes of the sine wave. The ROM output is presented to a Digital to Analog Converter (DAC) which produces a quantized sine wave output. The quantized sine wave output is smoothed using a filter that removes the high frequency components of the signal. Figure 3.5 shows the outputs of the different blocks of the “standard DDS”.

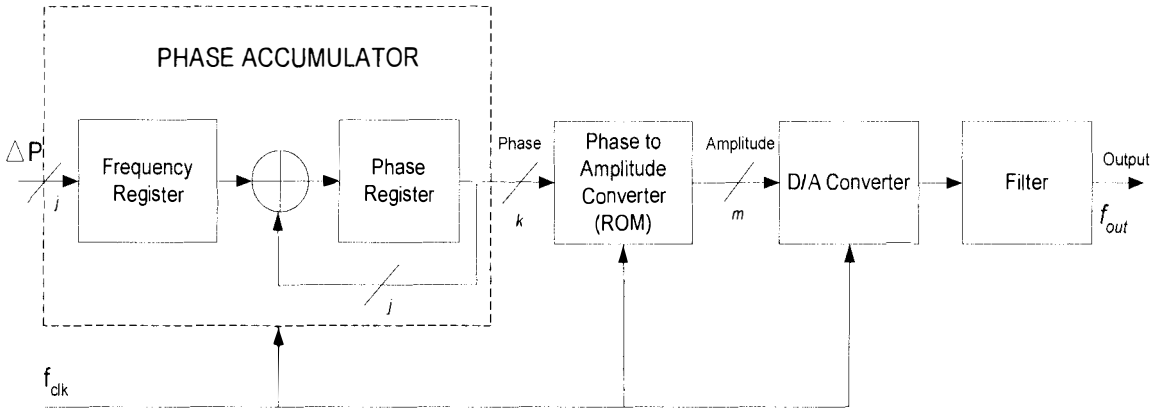



**Figure 3.4:** Block Diagram of the Direct Digital Frequency Synthesizer

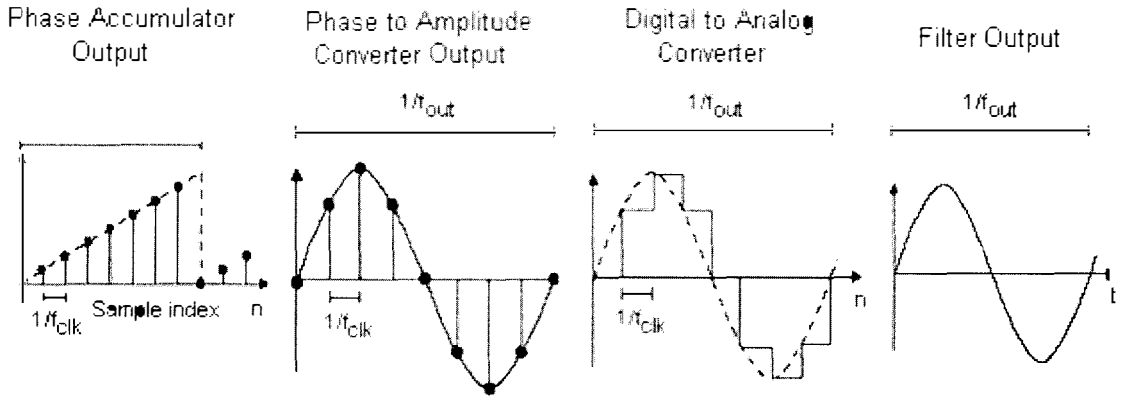



**Figure 3.5:** Outputs of the different blocks of a "standard DDS"

### 3.2.2.3 Pulse Output Direct Digital Frequency Synthesizer

The pulse output Direct Digital Frequency Synthesizer is the simplest kind of Direct Digital Frequency Synthesizer [13]. It has only a Phase Accumulator (PA). The MSB or the carry output signal of the Phase Accumulator is used as the output. The output of a DDFS represents the instantaneous amplitude of the signal. The task of the DDFS is therefore to generate this sinusoid oscillating number. For a constant frequency ($\omega$), the phase $d\phi/dt$ is also a constant. The phase of a constant frequency is linear. Therefore a digital phase accumulator can be used to integrate the frequency to provide the digital phase information.

Figure 3.6 represents the block diagram of this type of DDFS. It consists of a FTW Register in conjunction with an n-bit Phase Register driven by system clock ($f_{clk}$). The frequency register holds the n-bit FTW. For each clock cycle, the frequency is added to the phase accumulator. The lowest frequency that can be generated occurs when just 1 Least Significant Bit (LSB) is added to the accumulator.

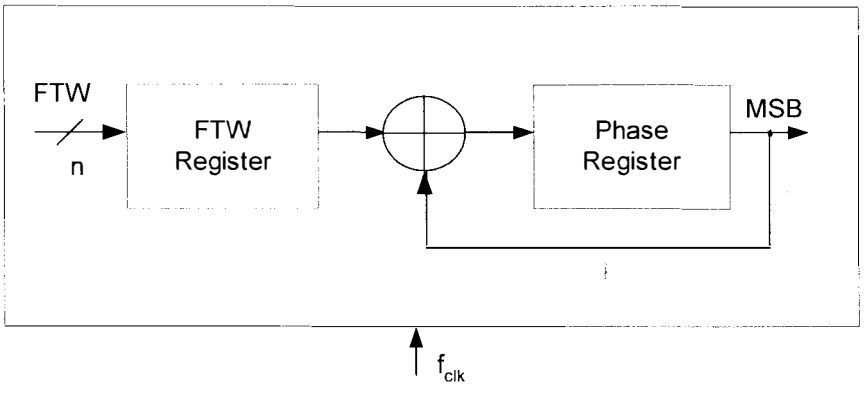



**Figure 3.6:** Block Diagram of the Pulse Output Direct Digital Frequency Synthesizer

Thus the phase accumulator increments through all the possible states until it overflows. This takes $2^n$ clock cycles for an n-bit accumulator thus the output frequency is $f_{clk}/2^n$. Thus the addition of the number FTW to the accumulator for each cycle of the clock results in the output frequency of $FTW * f_{clk}/2^n$. The following equation relates the output frequency of a DDFS $(f_{out})$ to the tuning word,

$$f_{out} = \frac{(FTW) * (f_{clk})}{2^n} \tag{3.1}$$

where $f_{clk}$ is the system clock frequency and $n$ is the length of the FTW in bits.

For the pulse output DDFS, the MSB or the carry output of the phase accumulator is used as the output. The sine wave oscillation can be visualized as a vector rotating around a phase wheel as shown in Figure 3.7.

Each point on the wheel corresponds to an equivalent point on the sine waveform. Thus as the phase accumulator increments through all the possible states, it results in the completion of one cycle of the sine waveform. The number of discrete points on the wheel is determined by the resolution, $n$ of the accumulator. The phase accumulator provides the vector's linear representation around the phase wheel. Figure 3.8. shows the output of the phase accumulator. The accumulator follows the following equation

$$S(n) = S(n-1) + FTW \tag{3.2}$$

where $S(n)$ is the output of the accumulator at clock tick $n$ and $FTW$ is the input control Frequency Tuning Word.

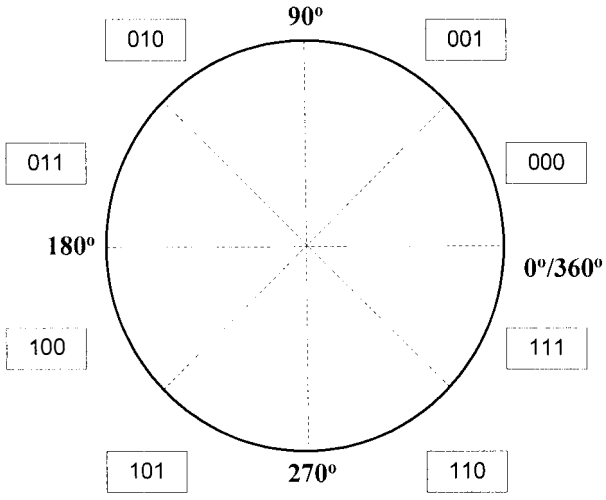



**Figure 3.7:** Phase Wheel Representation

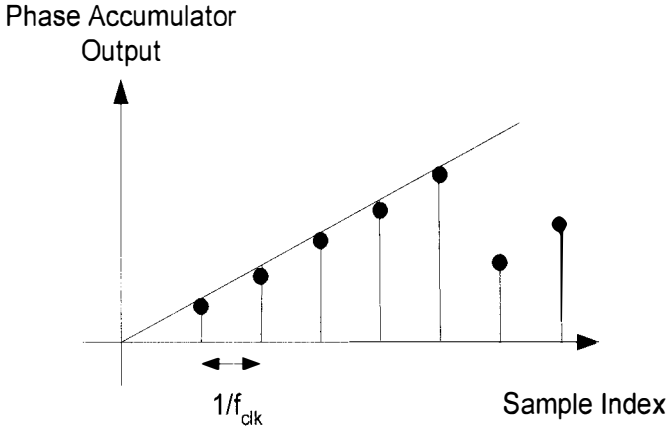



**Figure 3.8:** Phase Accumulator Output

As long as FTW evenly divides into $2^n$, the output is smooth and periodic but all other cases cause jitter. This is illustrated in Figure 3.9. If the desired output frequency is not a factor of $2^n$, then a phase error is created between the ideal and the actual output. The phase error will continue to increase or decrease until it reaches a full clock period, when it returns to zero and starts to build up again. Table 3.1 shows the outputs of the accumulator when $FTW = 3$, $FTW = 4$ and $n = 4$. The rate of error or the Error Periodicity is given by

$$Error\ periodicity = T * \gcd(FTW, 2^n) \tag{3.3}$$

where gcd(*FTW*, $2^n$) is the greatest common divisor of FTW and $2^n$ and $T = 1/f_{clk}$, the clock tick duration. When gcd(FTW, $2^n$) = $2^n$ there is no error. Hence, for the case when n = 4 and FTW = 4, the Error Periodicity = 0 cycles.

However when n = 4 and FTW = 3 the Error Periodicity is given by,

$$\gcd(3,16) * T = 48 * T = \frac{48}{16} = 3\ cycles \tag{3.4}$$

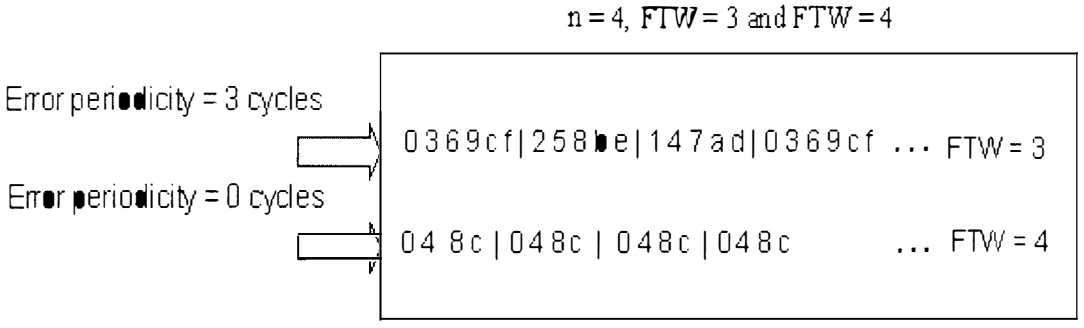



**Figure 3.9:** Single bit DDS Waveform

**Table 3.1:** For an accumulator with n = 4 and an input of FTW = 3 and FTW = 4

| Accumulator output FTW = 4 | Carry Output | Accumulator output FTW = 3 | Carry Output |
|---|---|---|---|
| 0000 (0) | 1 Start of Cycle | 0000 (0) | 1 Start of Cycle |
| 0100 (4) | 0 | 0011 (3) | 0 |
| 1000 (8) | 0 | 0110 (6) | 0 |
| 1100 (c) | 0 | 1001 (9) | 0 |
| 0000 (0) | 1 | 1100 (c) | 0 |
| 0100 (4) | 0 | 1111 (f) | 0 |
| 1000 (8) | 0 | 0010 (2) | 1 |
| 1100 (c) | 0 | 0101 (5) | 0 |
| 0000 (0) | 1 | 1000 (8) | 0 |
| 0100 (4) | 0 | 1011 (b) | 0 |
| 1000 (8) | 0 | 1110 (e) | 0 |
| 1100 (c) | 0 | 0001 (1) | 1 |

### 3.2.3 Control Unit

The Control Unit works in conjunction with the All Digital Phase Frequency Detector (ADPFD). The Control Unit consists of Gain registers namely the *ADD_GAIN* and *SUB_GAIN* registers which are reset to some initial values. Based on the occurrence of the *Up* or *Down* signal, the DDFS frequency is changed by either incrementing or decrementing the gain value from the present DDFS frequency tuning word. The gain is changed to a value which is the previous gain right shifted by 2 bits whenever *Up* changes to *Down* or *Down* changes to *Up* during successive comparisons. A fixed window width was chosen long enough to make an accurate decision, especially when the DDFS tries to lock to the reference clock input. The following examples illustrate the algorithm used by the Control Unit to achieve the required Frequency Tuning Word.

**Illustration 1:** If the Reference Clock Frequency is 50 MHz and System Clock Frequency is 100 MHz and if the frequency of the DDFS is locked to the reference clock frequency, then the desired frequency tuning word is 8000 (hex). The number of bits in the FTW is $n = 16$.

$$50\,MHz = \frac{(FTW)\,(100MHz)}{2^{16}} \Rightarrow FTW = 8000\,(hex)$$

Table 3.2 illustrates how the FTW is changed to the desired value of 8000 (hex). Figure 3.10 shows the MATLAB simulation results of the above example.

**Illustration (ii):** If the System Clock Frequency is 400 MHz and if the DDFS is locked to the reference clock frequency (50 MHz) the desired tuning word is 2000 (hex).

$$50\,MHz = \frac{(FTW)\,(400MHz)}{2^{16}} \Rightarrow FTW = 2000\,(hex)$$

Table 3.3 illustrates how the FTW is changed to the desired value of 2000 (hex). Figure 3.11 shows the MATLAB simulation results of the above example.

Figure 3.12 presents the results of various cases, (a) – (d) of reference and system clock frequencies and illustrate the changes in the output clock frequency until it settles down to match the frequency of the reference clock when the computed frequency tuning word equals the required tuning word. The number of iterations needed to achieve locking also varies in each case.

**Table 3.2:** *REF_Clk* = 50 MHz and *DDS_CLK* = 100 MHz

| Signal Asserted | Add Gain (hex) | Sub Gain (hex) | FTW (hex) |
|---|---|---|---|
| Initialization (Upon Reset) | 8000 | 4000 | 0000 |
| *Down* | 8000 | 4000 | 8000 |

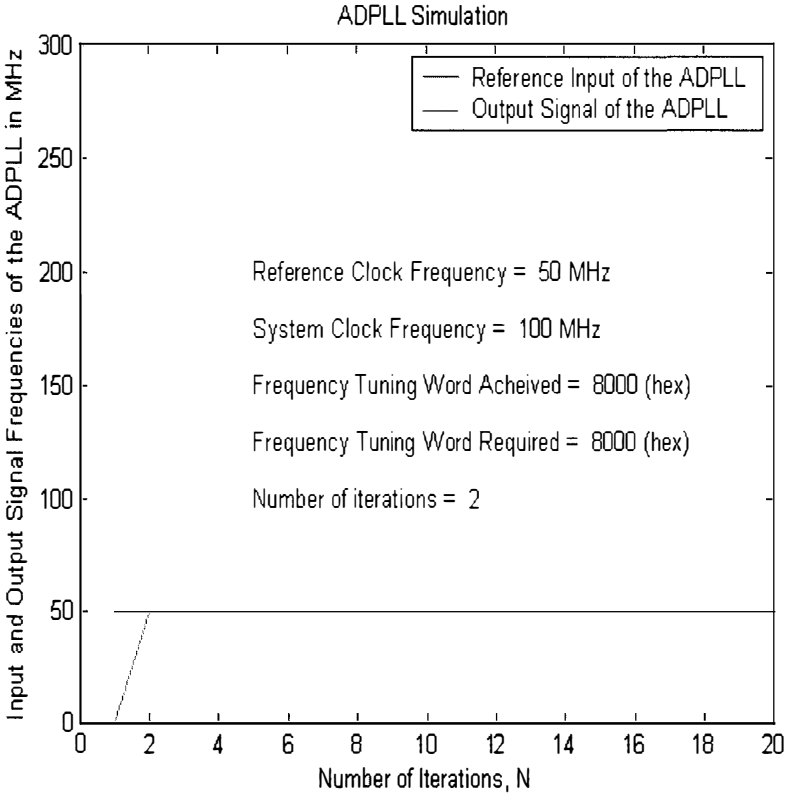



**Figure 3.10:** MATLAB Results: *REF_Clk* = 50 MHz and *DDS_CLK* = 100 MHz

**Table 3.3:** *REF_Clk* = 50 MHz and *DDS_CLK* = 400 MHz

| Signal Asserted | Add Gain (hex) | Sub Gain (hex) | FTW (hex) |
|---|---|---|---|
| Initialization (Upon Reset) | 8000 | 4000 | 0000 |
| *Down* | 8000 | 4000 | 8000 |
| *Up* | 2000 | 4000 | 4000 |
| *Up* | 2000 | 4000 | 0000 |
| *Down* | 2000 | 1000 | 2000 |

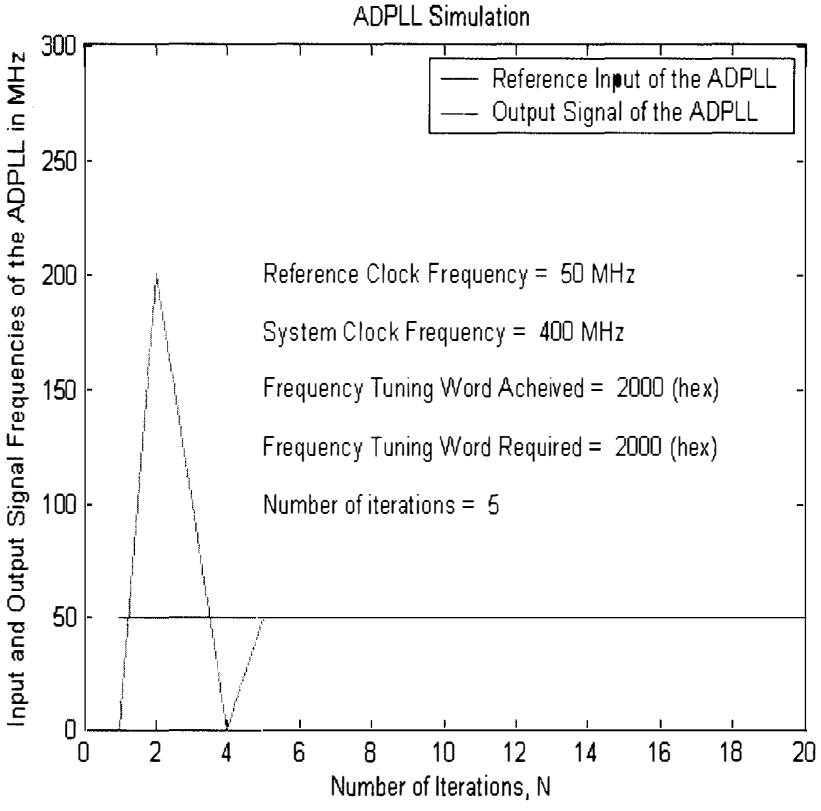
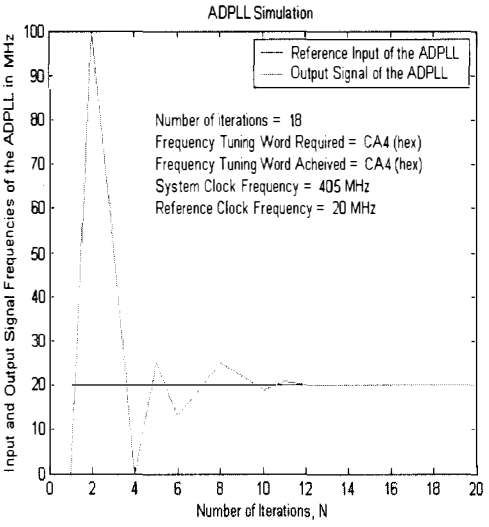
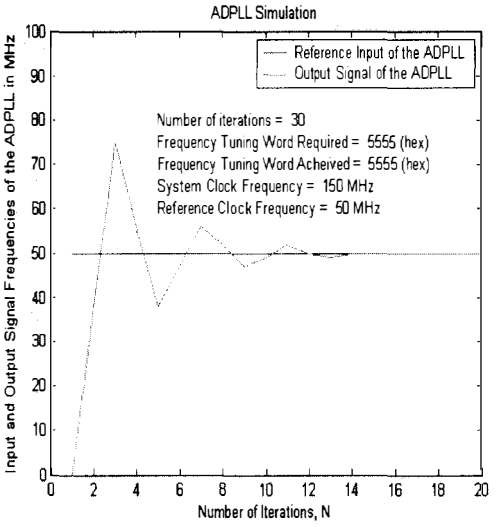


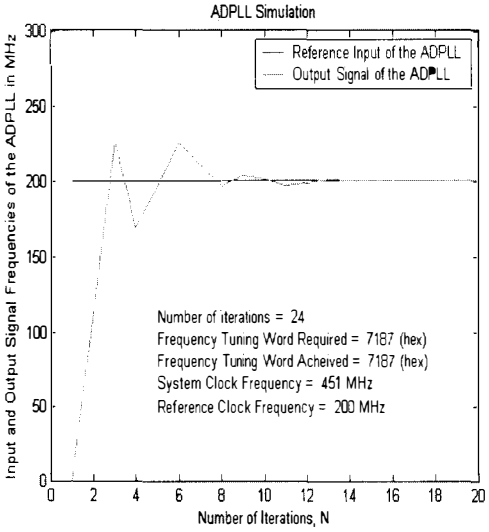

**Figure 3.11:** MATLAB Results: *REF_Clk* = 50 MHz and *DDS_CLK* = 400 MHz



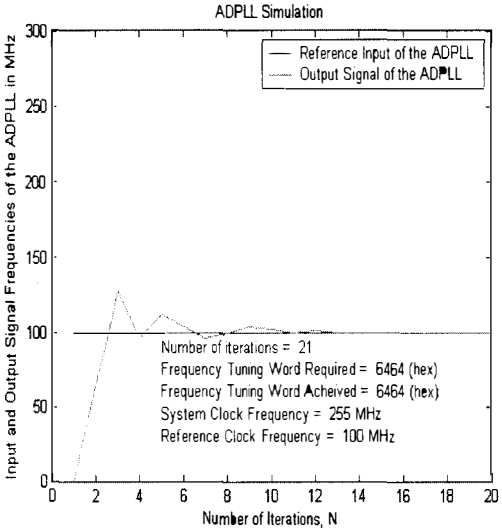

(a)




(b)




(c)




(d)

**Figure 3.12:** Illustrations of the Locking Process for various cases (a) – (d)

# Chapter 4

# Results

## 4.1 Introduction

Chapter 3 discussed the different blocks of the ADPLL and the implementation details. This chapter is divided into two sections: Section 4.1.1 presents the Simulation Results of the ADPLL. Section 4.1.2 presents the Test Results of the ADPLL. The Register Transfer Level (RTL) Verilog codes and the test benches are included in the Appendix A. A MATLAB program was written to simulate the working of the ADPLL and is given in Appendix B. Appendix C shows the set up used in the testing the ADPLL prototype.

### 4.1.1 ADPLL Simulation Results

The test bench for the top module of the ADPLL instantiates the device under test (DUT), creates the required clock signals and inputs the test vectors into the device. Figure 4.1 shows the input and output ports of the different blocks of the ADPLL and the top-level signals of the ADPLL that were routed to the pads of the ADPLL IC.

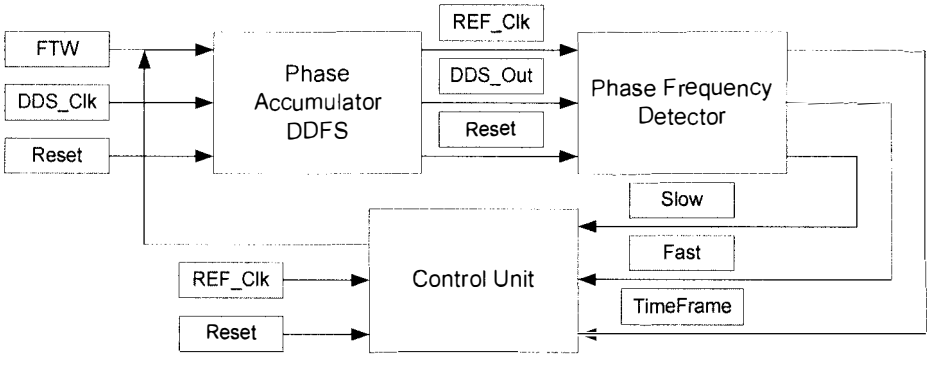



**Figure 4.1:** ADPLL Signals

Figure 4.2 shows the Post Synthesis Simulation results of the Phase Frequency Detector (PFD) when *REF_Clk* = 25 MHz and *DDS_Clk* = 100 MHz. The *Time_Frame* signal is used to select a fixed window (40 *REF_Clk* cycles) to perform the comparison. The screenshot has *Slow* = 0; *Fast* = 0 when the two signals *REF_Clk* and *DDS_Out* are locked to each other. The "*Slow*" and "*Fast*" signals are analogous to the "*Down*" and "*Up*" signals of the PFD discussed in Chapter 3. Figure 4.2 shows the period of the *DDS_Out* = $19985 - 19945 = 40\ ns$ which corresponds to 25 MHz.

Figure 4.3 shows the Post Synthesis Simulation results of the pulse output DDFS. The contents of the Frequency Tuning Word are modified from 3C00 (hex) to 4000 (hex) by the Control Unit. This is the required tuning word to achieve the locking of the input signal with the output of the DDFS. The period of the output of the DDFS is $21345 - 21305 = 40\ ns$.

Figure 4.4 shows the Post Synthesis Simulation results of the Control Unit. The Control Unit computes the Frequency Tuning word (3c00 hex) in the previous iteration and uses a comparison window of 40 *REF_Clk* cycles and since the PFD outputs a "*Slow*" signal, it uses the contents of the gain registers and changes the FTW to 4000 (hex).

Figure 4.5 shows the Post Synthesis Simulation results of the Complete ADPLL. The ADPLL output ports are *DDS_Out* and input ports are *REF_Clk, DDS_Clk* and *Reset*.

Figure 4.6 shows the Post Layout Simulation results of the Complete ADPLL when *REF_Clk* =25 MHz and *DDS_CLK* = 100 MHz.

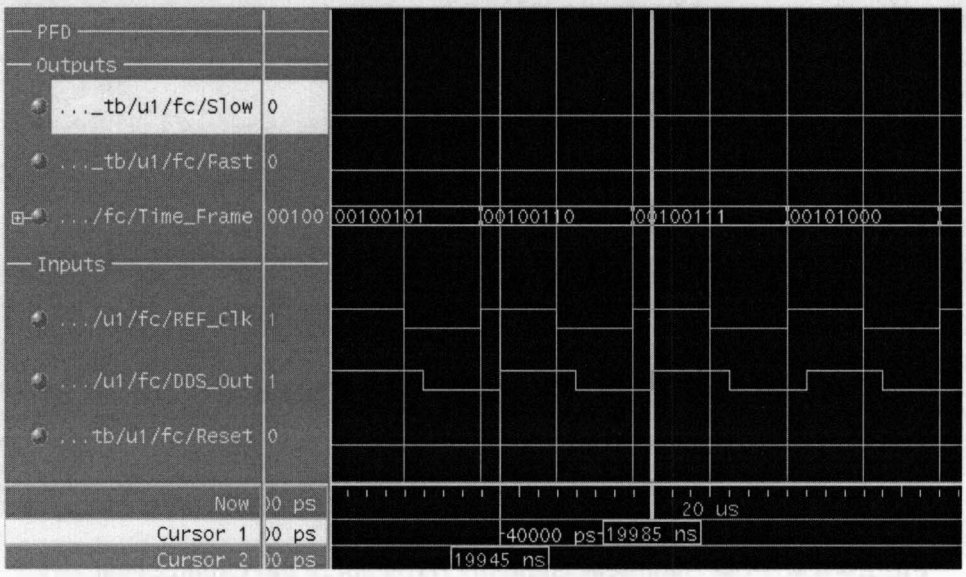



**Figure 4.2:** Post Synthesis Simulation results of the PFD

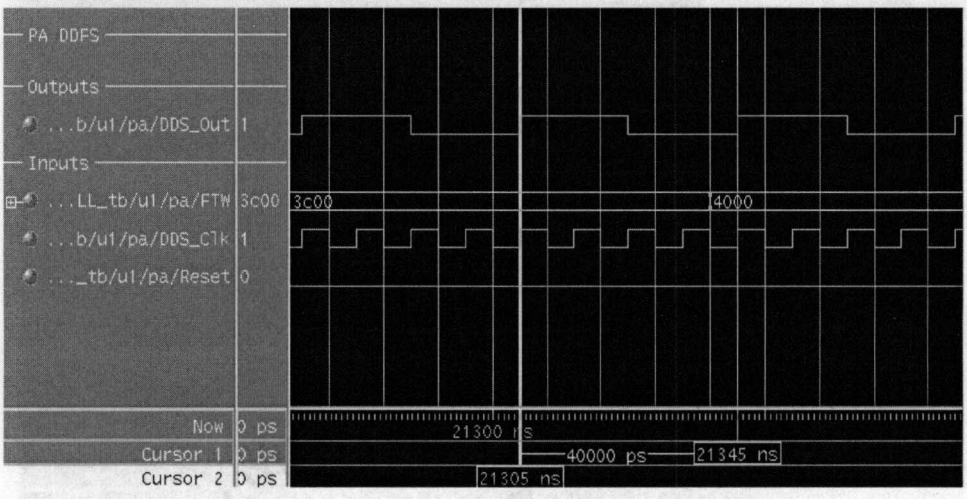



**Figure 4.3:** Post Synthesis Simulation results of the PA DDFS

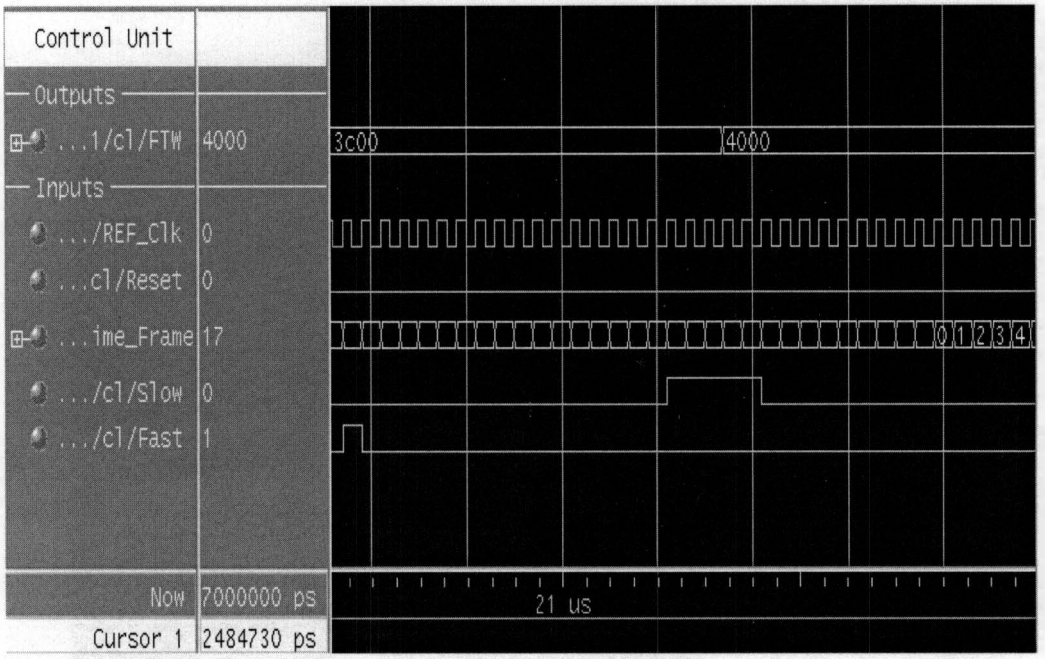



**Figure 4.4:** Post Synthesis Simulation results of the Control Unit

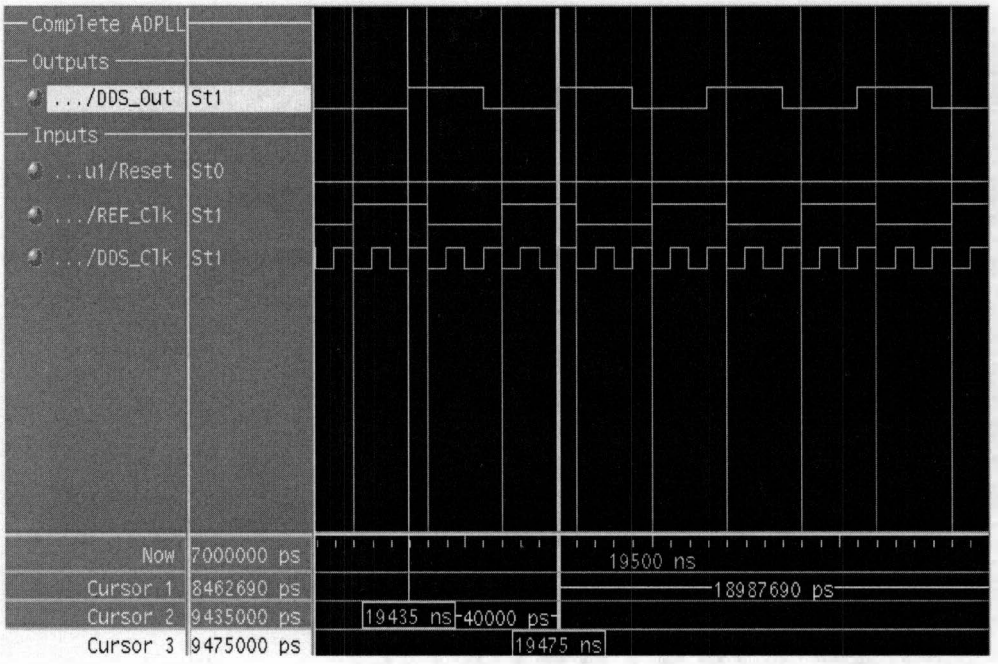



**Figure 4.5:** Post Synthesis Simulation results of the Complete ADPLL

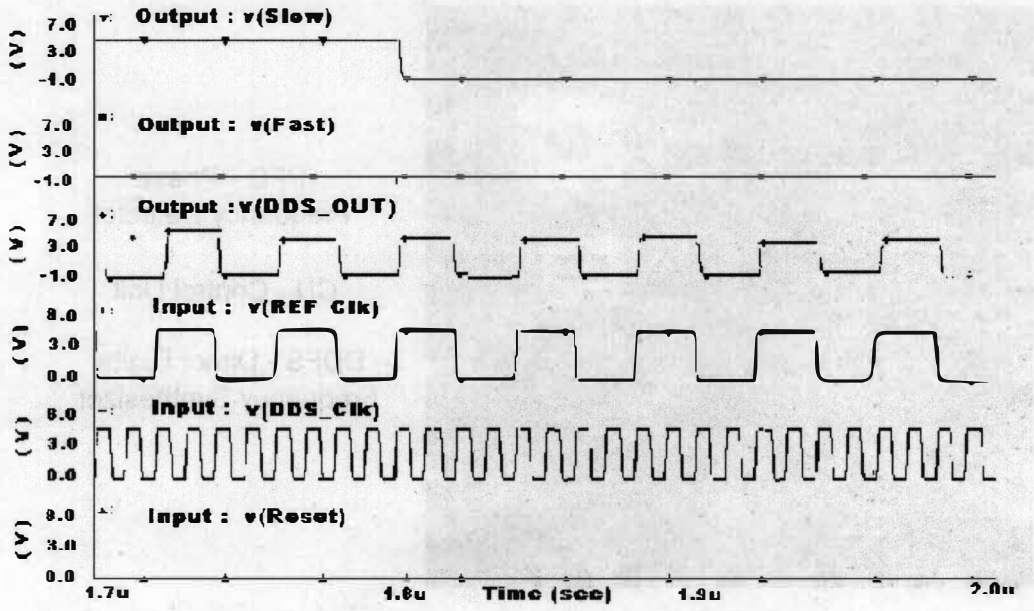



**Figure 4.6:** Post Layout Simulation results of the Complete ADPLL

### 4.1.2 Experimental Results

The ADPLL was fabricated in a 0.6-μm 1-poly 3-metal CMOS technology. Figure 4.7 shows the chip layout. The ADPLL Control Unit, Phase Detector and the DDFS modules were synthesized individually and placed and routed using Automatic Place and Route (APR) tools and Cadence Virtuoso tools to facilitate the ease of testing. The total size of the test circuit (excluding the bond pads, power supply, test pins etc), is 0.87 $mm^2$. Figure 4.8 shows the test results obtained with the ADPLL chip and the test board set up shown in Appendix C. The "Reset board" provides a reset signal to the chip with a pulse width of about 500 ns, which initializes the gain registers. The frequency of the *REF_Clk* was 1 MHz and the frequency of the *DDS_Clk* was 4 MHz. The output frequency of the PLL is exactly 1 MHz and the output voltage amplitude is 4.96 V.

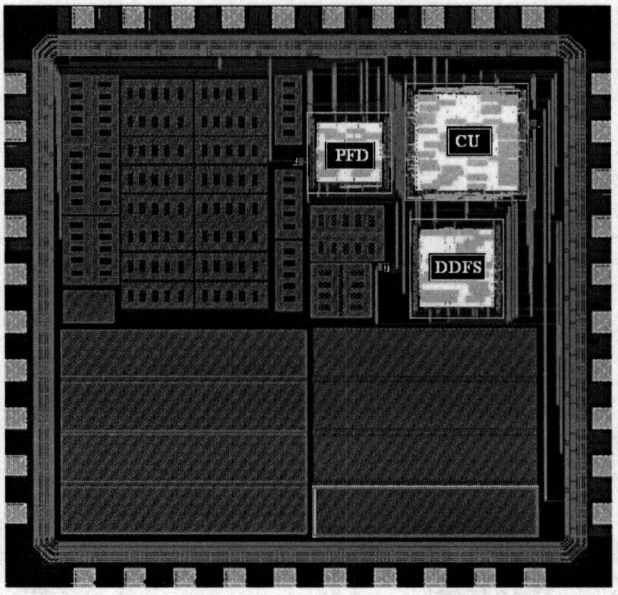



**Figure 4.7:** Layout of the Complete ADPLL

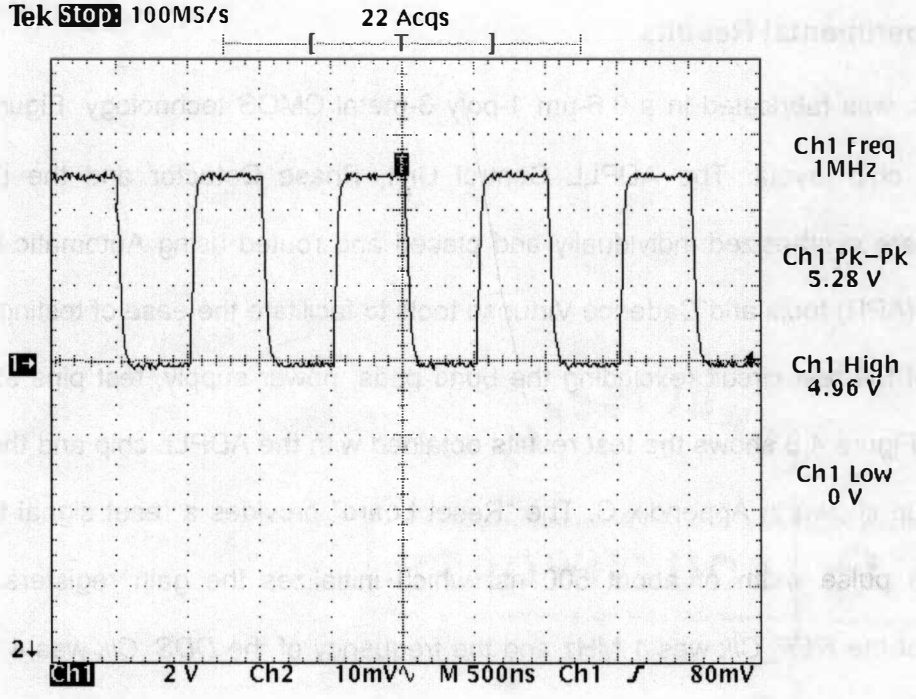



**Figure 4.8:** Measured output when *REF_Clk* is 1 MHz and *DDS_Clk* is 4 MHz

Figure 4.9 shows the measured output signal when the *REF_Clk* was changed to 3 MHz and the *DDS_Clk* changed to 7 MHz. The Reset board was used to provide the reset signal to the ADPLL IC to initialize the gain registers. The output signal frequency was 2.995 MHz and the output signal amplitude was 5.28 V. The Duty Cycle Distortion (DCD) was measured to be $169.61\ ns - 164.27 ns = 5.34 ns$.

Figure 4.10 shows the measured output signal when *REF_Clk* is 15 MHz and *DDS_Clk* is 60 MHz. The output waveform in Figure 4.10 is not a perfect square wave when the frequency of the *REF_Clk* is 15 MHz. The reason of this result could be that the outputs of the PFD are determined after a time window of 40 *REF_Clk* cycles and the phase error builds up during this time to cause non-zero PFD pulses after locking-in.

Table 4.1 summarizes the characteristics of the reported ADPLLs and the proposed ADPLL.

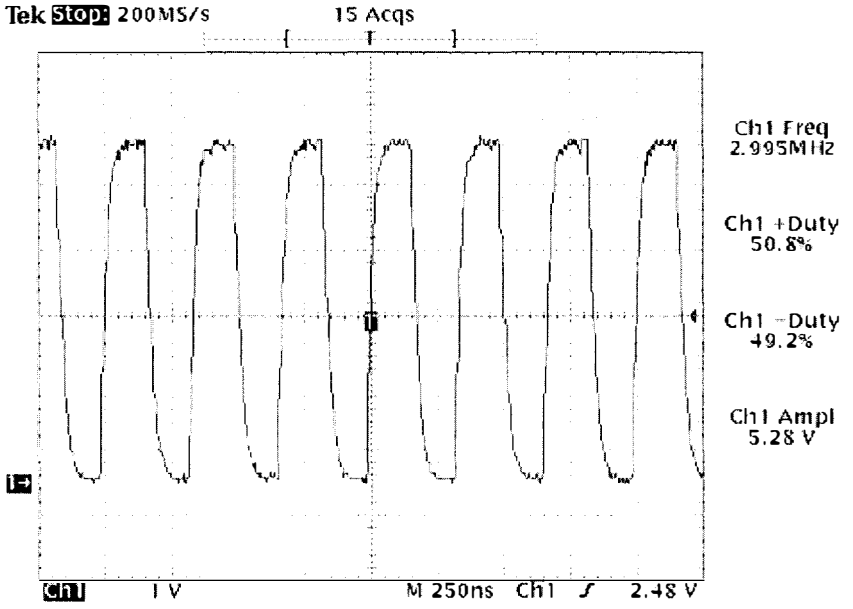



**Figure 4.9:** Measured output when *REF_Clk* is 3 MHz and *DDS_Clk* is 7 MHz

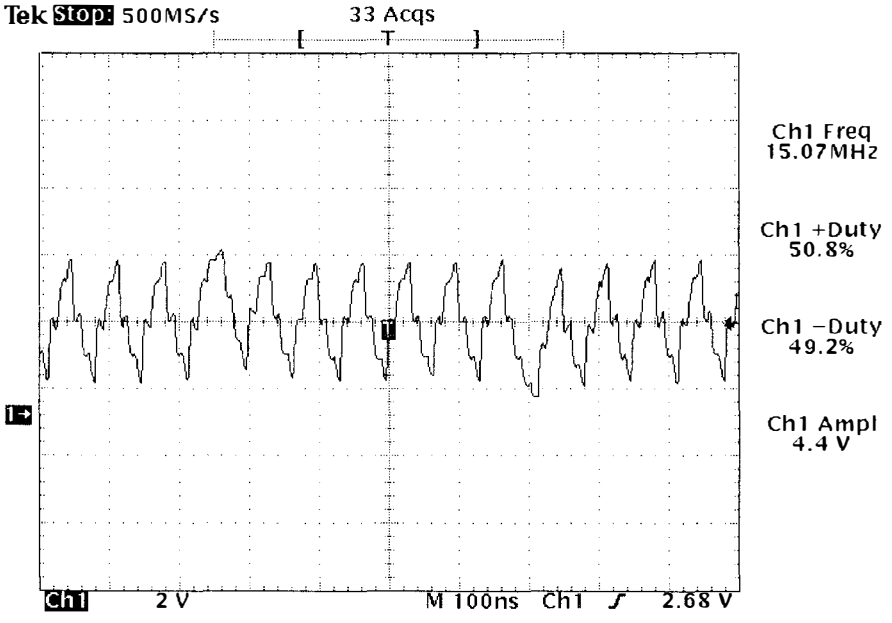



**Figure 4.10:** Measured output when *REF_Clk* is 15 MHz and *DDS_Clk* is 60 MHz

**Table 4.1:** Summary of the characteristics of the reported and proposed ADPLLs

| Performance Characteristic | [6] | [10] | [11] | [12] | Proposed ADPLL |
|---|---|---|---|---|---|
| Process | 0.25-μm CMOS | 0.5-μm CMOS | 0.6-μm CMOS | 0.6-μm CMOS | 0.6-μm CMOS |
| Area | 0.09 $mm^2$ | 0.71 $mm^2$ | 0.83 $mm^2$ | 2.75 $mm^2$ | 0.87 $mm^2$ (6930 transistors) |
| Methodology | Analog | All Digital | Semi Digital | All Digital | All Digital |
| Lock Time | < 720 cycles | < 50 cycles | < 16 cycles | < 25 cycles | 50 cycles |
| Minimum Frequency | 8.5 MHz | 50 MHz | 300 MHz | 360 MHz | 1 MHz |
| Maximum Frequency | 660 MHz | 550 MHz | 800 MHz | 800 MHz | 500 MHz |
| Supply Voltage | 1.9 V | 3.3 V | 3.3 V | 3.3 V | 5 V |
| Duty Cycle Distortion | NA | NA | NA | NA | < 2% |

# Chapter 5

# Conclusion

## 5.1 Summary

This thesis discussed the basic techniques used for the design of All Digital Phase Locked Loops. The design was implemented using 0.6-μm CMOS technology. The contributions of this work include:

1. Designing of an All Digital Phase Locked Loop (ADPLL) using a Direct Digital Frequency Synthesizer (DDFS) and an All Digital Phase Frequency Detector (ADPFD) with a faster lock-in time.

2. Designing of an All Digital Phase Locked Loop (ADPLL) using digital components so as to make the design scalable and portable.

## 5.2 Future Work

Although the design has been tested at all levels and satisfactory results were obtained, a great deal of improvement could be made to enhance the performance of the existing design.

Firstly, the design of an All Digital PLL that presents a better jitter performance could be considered which is extremely critical in some applications. Since the output of the Phase Accumulator represents the amplitude of the sinusoid, modifying the Phase

Accumulator DDFS (PADDFS) by using all the bits of the PADDFS output can be considered.

Secondly, the current design requires resetting of the gain registers before the frequency of the input signal is varied. This is necessary as the gain registers hold different values at the end of a comparison. The important characteristic of a PLL is its ability to track changes in the input signal and acquire locking. Hence there is a need to modify the algorithm so that the PLL can track changes in the input frequency without the need to reset the registers.

Also, in some applications it might be necessary to convert the square wave digital output to a sine wave output. Hence it might be necessary to use a Digital to Analog Converter (DAC) to support such applications.

As the feature size of the CMOS technology shrinks down toward 0.1-μm and the clock frequency is beyond 1 GHz, these techniques have to be pursued to achieve robust performance over process, supply voltage and temperature variations. The concepts of this design have been proven on silicon using 0.6-μm CMOS.

In addition to the design concepts of the Digital Phase Locked Loops, this thesis also presents an insight into the comparison between the different implementations: Analog, Digital and All Digital PLLs which provides leads to further research and development in these areas.

## References

# References


[1] Thomas H. Lee, The *Design of CMOS Radio Frequency Integrated Circuits*, Cambridge University Press, 1998.

[2] Clock Jitter and Skew - http://www.eie.polyu.edu.hk/~swng/eie502/L8.pdf

[3] Roland E. Best, *Phase Locked Loops: Design, Simulation and Applications*, McGraw Hill, 1999.

[4] Semiconductor Industry Association (SIA), *International Roadmap for Semiconductors 2001 Edition*. Austin, TX: International SEMATECH, 2001, Executive Summary Chapter. Available: http://public.itrs.net.

[5] Lee, G.B., Chan, P.K., Siek, L., "A CMOS Phase Frequency Detector for Charge Pump Phase-Locked Loop," *42nd Midwest Symposium on Circuits and Systems*, vol. 2, pp.601-604, Aug. 1999.

[6] H.-T. Ahn and D.J. Allstot, "A low-jitter 1.9-V CMOS PLL for ultra-SPARC microprocessor applications," *IEEE J. Solid State Circuits*, vol. 35, pp.450-454, May 1999.

[7] I.A. Young, J.K. Greason and K.L. Wong, "PLL clock generator with 5 to 110 MHz of lock range," *IEEE J. Solid-State Circuits*, vol.27, pp.1599-1606, Nov.1992.

[8] J. Goto, M. Yanashina, T. Inoue, *et al* "A PLL-base programmable clock generator with 50 to 350 MHz oscillating range for video signal processors," *IEICE Trans. Electron*, vol. E77-C, no.12, pp.1951-1955, Dec.1994.

[9] H. Kondoh, H. Notani, T. Yosimura, *et al*, "A 1.5V 250 MHz operation CMOS phase-locked loop with precharge type phase-frequency detector," *IEICE Trans. Electron*, vol. E78-C, no.4, pp.381-388, Apr.1995.

[10] J. Dunning, G. Garcia, J. Lundberg, and E. Nuckolls, "An all-digital phase-locked loop with 50-cycle lock time suitable for high performance microprocessors," *IEEE J. Solid-State Circuits*, vol.30, pp.412-422, Apr. 1995.

[11] I. Hwang, S. Lee, S. Lee and S. Kim, "A digitally controlled oscillator phase-locked loop with false locking scheme for clock synthesis applications," *IEEE Int. Solid State Circuits Conf. Dig. Tech Papers*, Feb. 2000, pp.168-169.

[12] T.-Y. Hsu, C.C-Wang and C.-Y. Lee, "Design and analysis of a portable high-speed clock generator," *IEEE Trans. Circuits Systems. II*, vol. 48, pp. 367-375, Apr. 2001.

[13] J. Vankka, "*Direct Digital Synthesizers, Theory, Design and Applications*", Ph.D. dissertation, November 2000.

[14] R. Jacob Baker, Harry W. Li and David E. Boyce, "*CMOS Circuit Design, Layout and Simulation*," Prentice Hall of India Private Ltd, New Delhi, 2000.

[15] M. Horowitz, A. Chan, J. Corbrunson, et al, "PLL Design for a 500 Mb/s interface," *ISSCC Dig. Tech. Papers*, pp. 160-161, Feb 1993.

[16] M. Negahban, R. Behrasi, G. Tsang, et al, "A two chip CMOS read channel for hard-disk drives," *ISSCC Dig. Tech. Papers*, pp. 216-217, Feb 1993.

[17] W.D. Llewellyn, M.M.H. Wong, G.W. Tietz and P.A. Tucci, "A 33 Mb/s data synchronizing phase-locked loop circuit," ISSCC *Dig. Tech. Papers*, pp. 12-13, Feb. 1998.

[18] Khalil, A.H, Ibrahim, K.T., Salama, A.E., "Design of ADPLL for good phase and frequency tracking performance," *Proceedings of the Nineteenth National Radio Science Conference*, March 19-21, 2002, pp.284-290

[19] R. Saban and A. Efendovich, "A fully-digital, 2MB/sec CMOS data separator," in *ISCAS'94*, vol.3, pp.53-56.

[20] J.-S. Chiang and K.-Y. Chen, "A 3.3 V all digital phase locked loop with small DCO hardware and fast phase lock," in *Proc. ISCAS '98*, vol.3 pp.554-557.

[21] J.-S. Chiang and K.-Y. Chen, "The Design of an All-Digital Phase-Locked Loop with Small DCO hardware and Fast Phase Lock," *IEEE Trans. On Circuits and Systems*, vol. 46, No.7, pp. 945-950

[22] Bar-Giora Goldberg, "*Digital Techniques in Frequency Synthesis,*" McGraw-Hill, 1996.

[23] N. Retdian, S. Takagi and N. Fujii, "Voltage Controlled Ring Oscillator with Wide Tuning Range," *Proceedings of AP ASIC Conference 2002*. http://www.ap-asic.org/2002/proceedings/4A/4A-5.PDF

[24] R. Fried, "Low-power digital PLL with one cycle frequency lock-in time for clock syntheses up to 100 MHz using 32,768 Hz reference clock," *Proc. IEEE*, vol. 84, pp. 291-294, Feb. 1996.

# Appendices

# Appendix A: Verilog RTL Code

**topmodulePLL.v – Top Level module**

/* This module instantiates the modules PFD, Phase Accumulator and Control Unit. The signals in the module instantiation are linked to the ports of the modules using named associations. The input ports of the top module are DDS_Clk, Reset, REF_Clk. The top module has a single output port, DDS_Out. In addition to the input and output ports the top module also uses inout ports for the FTW, Slow and Fast Signals. */

```
`timescale 1ns/10ps
module topmodulePLL (DDS_Out, Slow, Fast, DDS_Clk, Reset, REF_Clk);

/* Declaration of Output Ports */
output DDS_Out;

/* Declaration of Input Ports */
input DDS_Clk;
input Reset;
input REF_Clk;
inout [15:0] FTW;
inout Slow, Fast;
wire [7:0] Time_Frame;

/* Instantiation of the PFD module */
PFD pd(.Time_Frame (Time_Frame), .Slow (Slow), .Fast (Fast), .REF_Clk (REF_Clk),
.DDS_Out (DDS_Out), .Reset (Reset));

/* Instantiation of the Phase Accumulator module */
PhaseAccumulator pa(.DDS_Out (DDS_Out), .DDS_Clk (DDS_Clk), .FTW(FTW),
.Reset(Reset));
```

```
/* Instantiation of the Control Unit module */
ControlLogic cl(.FTW(FTW), .REF_Clk(REF_Clk), .Reset(Reset), .Time_Frame
(Time_Frame), .Slow(Slow), .Fast(Fast));

endmodule

/* End of Module */
```

**topmodulePLL_tb.v – Test bench module**

```
/* This module creates the input test vectors for the device under test. The time periods
of the REF_Clk and DDS_Clk can be changed according to the frequency of the
clocks. */

`timescale 10ps/1fs
module topmodulePLL_tb();

wire [15:0] FTW;
reg REF_Clk;
reg Reset;
reg DDS_Clk;
wire [15:0] ph_acc;
wire DDS_Out;
reg select;
reg REF_Clk1, REF_Clk2;

initial
begin
        $shm_open("waves.shm");
        $shm_probe("AS");
end

topmodulePLL t1(.DDS_Out(DDS_Out), .DDS_Clk(DDS_Clk), .Reset(Reset),
.REF_Clk(REF_Clk), .FTW(FTW));
```

```
always@(REF_Clk1 or REF_Clk2 or select)
begin
        if(select == 1) REF_Clk = REF_Clk1;
        else
        if(select == 0) REF_Clk = REF_Clk2;
end

initial
begin
        select = 1'b1;
        REF_Clk1 = 1'b0;
        REF_Clk2 = 1'b0;
        DDS_Clk = 1'b0;
        Reset = 1'b0;
        #5 Reset = 1'b1;
        #500 Reset = 1'b0;
        #1000000 select = 1'b1;
end

always REF_Clk1 = #10 ~REF_Clk1; /* Reference clock frequency - 50 MHz */
always REF_Clk2 = #50 ~REF_Clk2;
always DDS_Clk = #5 ~DDS_Clk; /* System clock frequency - 100 MHz */

endmodule
/* End of Module */
```

**pfd.v – Phase Frequency Detector Module**

```
/* This module outputs the Slow (Down) or Fast (Up) signals based on the comparison
between the REF_Clk and the DDS_Out inputs. It uses a time window, Time_Frame to
count the clock cycles and after the time window compares the two counters and
generate the Fast/Slow Signals */
```

```
`timescale 1ns/10ps

module FreqComp(Time_Frame, Slow, Fast, REF_Clk, DDS_Out, Reset);

/* Declaration of Output Ports */
output Slow,Fast;
output [7:0] Time_Frame;

/* Declaration of Input Ports */
input DDS_Out;
input REF_Clk, Reset;

reg [7:0] Time_Frame;
reg Slow, Fast;
reg [7:0] count1, count2;
wire comp1 = Time_Frame > 9 & Time_Frame < 31;
always@(posedge REF_Clk or posedge Reset)
begin
        if(Reset)
                begin
                        count1 = 0;
                        Time_Frame = 0;
                end
        else
        if(Time_Frame < 41)
                        begin
                                if(comp1)
                                        begin
                                                count1 = count1 + 1;
                                                Time_Frame = Time_Frame + 1;
                                        end
                                else if(Time_Frame > 30)
                                        begin
```

```
                                        count1 = 0;
                                        Time_Frame = Time_Frame + 1;
                                end
                        else Time_Frame = Time_Frame + 1;
                end
        else if (Time_Frame > 40) Time_Frame = 0;
end

always@(posedge DDS_Out or posedge Reset)
begin
        if(Reset)
                count2 = 0;
else
if(Time_Frame > 30)
                        count2 = 0;
else if(comp1)
                        count2 = count2 + 1;
end
always@(posedge REF_Clk)
begin
                if(count1 > count2)
                        begin
                                Slow = 1'b1;
                Fast = 1'b0;
                        end
                else
                if(count2 > count1)
                                begin
                                        Fast = 1'b1;
                                        Slow = 1'b0;
                        end
                        else
                                begin
```

```
                    Slow = 1'b0;
                    Fast = 1'b0;
                end
end
endmodule
/* End of Module */
```

### controllogic.v – Control Unit Module

/* This module uses the outputs of the PFD module and generates the Frequency Tuning Word by using the values in the Gain Registers, Add_Gain and Sub_Gain respectively. */

```
`timescale 1ns/10ps
module ControlLogic(FTW, REF_Clk, Reset, Time_Frame, Slow, Fast);

//* Declaration of Input and Output Ports
input REF_Clk, Reset;
input Slow, Fast;
input [7:0] Time_Frame;
output [15:0] FTW;

/* Declaration of Registers */
reg [15:0] FTW;
reg [14:0] Add_Gain, Sub_Gain;
reg flag1, flag2;

always@(posedge REF_Clk or posedge Reset)
begin
    if(Reset)
        begin
            FTW = 16'h0000;
            Add_Gain = 15'h8000;
            Sub_Gain = 15'h4000;
```

```
                flag1 = 1'b0;
                flag2 = 1'b0;
            end
        else
            if(Time_Frame == 30) begin
                if(Slow)
                    begin
                        FTW = FTW + Add_Gain;
                        flag2 = 1'b1;
                        if(flag1)
                            begin
                                Sub_Gain = Sub_Gain >> 2;
                                flag1 = 1'b0;
                            end
                        else Sub_Gain = Sub_Gain;
                        end
                else if(Fast)
                    begin
                        FTW = FTW - Sub_Gain;
                        flag1 = 1'b1;
                    if(flag2)
                        begin
                            Add_Gain = Add_Gain >> 2;
                            flag2 = 1'b0;
                        end
                    else Add_Gain = Add_Gain;
                end
            end
        else FTW = FTW;
end
endmodule
/* End of Module */
```

# Appendix B: MATLAB Code

```
% This MATLAB program accepts the frequencies of the reference clock frequency
and % the system clock frequency and plots the frequencies of the output signal versus
the % number of iterations till it achieves lock with the input signal.
% Inputs to the Program: Reference Clock Frequency in MHz (f_ref) and System Clock
% Frequency in MHz (f_sys) entered as dialog box entries.
% Outputs of the Program: Plot of the Output Signal Variations with the number of
% iterations, N until it achieves lock with the reference signal. It also outputs as text the
% Desired Frequency Tuning Word and the computed Frequency Tuning Word at the
% end of N iterations.

% Clear Memory and Command Window

clear all; clc;

% Default Values of the Reference and System Clock Frequency in MHz

fref = 20;
fsys = 40;

% Error checking of the values entered by the user for the Reference and System
Clock % Frequencies

check = 0;
while check == 0
values = {num2str(fref),num2str(fsys)};
values = inputdlg(...
{'Reference Frequency in MHz', ...
 'System Clock Frequency in MHz'}, ...
 'ADPLL Simulation', 1, values);
if length(values) ==2
```

```
        check = 1;
        fref   = str2num(char(values(1)));
        fsys = str2num(char(values(2)));
                if fref < 0
                        waitfor(errordlg('Invalid Reference Clock Frequency'));
                        check = 0;
                end

              if fsys < 0
                        waitfor(errordlg('Invalid System Clock Frequency'));
                        check = 0;
                end
        end
end

% Initialization of the Add and Subtract Gains to 4000 (hexadecimal) and 2000
% (hexadecimal) respectively

add_gain = 8000;
sub_gain = 4000;

% Calculation of Desired Frequency Tuning Word (FTW). These had to be rounded off
% to prevent the error in using hex2dec and dec2hex conversions.

ftw_desired = round (fref  .* (2.^16) ./ fsys);

% Conversion of Add and Subtract Gains from hexadecimal to decimal format.

add_gain_dec = hex2dec (num2str (round (add_gain)));
sub_gain_dec = hex2dec (num2str (round (sub_gain)));

% Initialization of the variables. The variable “ftw” stores the calculated frequency
tuning % word in each iteration and is initialized to zero.
```

```
ftw = 0;

% The variable "count" keeps track of the number of iterations. It also serves as the
% index to the vector "ftw_vector" and "fout" which store the values of the Frequency
% Tuning Word and the Output Frequency during the successive iterations.

count = 1;

% The variable "comp" is used to check the Equality of variables "ftw" and
"ftw_desired"

comp = 0;

% Variables "flag1" and "flag2" keep track of whether the variable "up" was asserted
after "down" was asserted and vice versa in which case the corresponding gains are
shifted right by 2 bits or divided by 4.  The flag bits are initially zero.

flag1 = 0;
flag2 = 0;

% Variables "up" and "down" are used to check to perform the relative comparison of
% required tuning word and achieved tuning word. "up" and "down" are initially zero.

up = 0;
down = 0;

% Assignment of zero to the first element of the Frequency Tuning Word Vector as the
% output of the DDFS initially is a zero. This was done to avoid the "Divide by Zero"
% error while trying to divide by a Frequency Tuning Word of zero.

ftw_vector(count) = ftw;
fout(count) = 0;
```

```
% Incrementing the counter variable so as to begin from the
count = count + 1;

% The following statements perform the comparison of the frequency tuning words
% (desired and achieved) and increment or decrement the frequency tuning word
based % on the comparison.

while(count < 50 & comp == 0)
        if ftw < ftw_desired down = 1;
                ftw = ftw + add_gain_dec;
                ftw_vector(count) = ftw;
                fout(count) = round((ftw_vector(count).* fsys) ./ (2.^16));
                flag1 = 1;
                if(flag2 == 1)
                        sub_gain_dec = sub_gain_dec ./4;
                        flag2 = 0;
                end

        elseif ftw > ftw_desired up = 1;
                ftw = ftw - sub_gain_dec;
                ftw_vector(count) = ftw;
                fout(count) = round((ftw_vector(count).* fsys) ./ (2.^16));
                flag2 = 1;
                if(flag1 == 1)
                        add_gain_dec = add_gain_dec ./4;
                        flag1 = 0;
                end
        end

        count = count + 1;
        if (ftw == ftw_desired)
                comp = 1;
        end
```

```
end

% Calculation of the final values of the Add and Subtract Gains in hexadecimal format

add_gain = dec2hex(round(add_gain_dec));
sub_gain = dec2hex(round(sub_gain_dec));

% Open figure, reset all parameters and put hold "on" so that multiple
% item can be drawn on the same plot.
figure(1);
clf reset;
hold on;
box on;

% Assignment of the Reference Clock Frequency to vector “fref1” as it remains
constant % during the successive iterations

for ii=1:1:50
        fref1(ii) = fref;
end

% Assignment of the Output Clock Frequency to all the successive elements of the
“fout” % vector once the output frequency has locked to the reference. This is done to
set the % matrix dimensions equal.

jj = count - 1;
for ii = jj:1:50
        fout(ii) = fout(jj);
end

% Conversion of Achieved Frequency Tuning Word and the Desired Frequency Tuning
% Word to hexadecimal format
```

```
ftw = dec2hex(ftw);
ftw_desired = dec2hex(ftw_desired);

% Plots of the reference clock signal and the output signal with respect to the number
of % iterations

ii = [1:1:50]
plot(ii,fref1,'b',ii,fout,'r');
axis([0 20 0 300]);
title('ADPLL Simulation');
xlabel('Number of Iterations, N');
ylabel('Input and Output Signal Frequencies of the ADPLL in MHz');
legend('Reference Input of the ADPLL','Output Signal of the ADPLL',1);

% Add text notes to display the obtained results
text(5, 30, ['Reference Clock Frequency =  ',num2str(fref),' MHz '],...
    'HorizontalAlignment','left');

text(5, 45, ['System Clock Frequency =  ',num2str(fsys),' MHz '],...
    'HorizontalAlignment','left');

text(5, 60, ['Frequency Tuning Word Acheived =  ',num2str(ftw),' (hex) '],...
    'HorizontalAlignment','left');

 text(5, 75, ['Frequency Tuning Word Required =  ',num2str(ftw_desired),' (hex) '],...
    'HorizontalAlignment','left');

 text(5,90, ['Number of iterations =  ',num2str(count-1),],...
    'HorizontalAlignment','left');
```

# Appendix C: Testing Set up

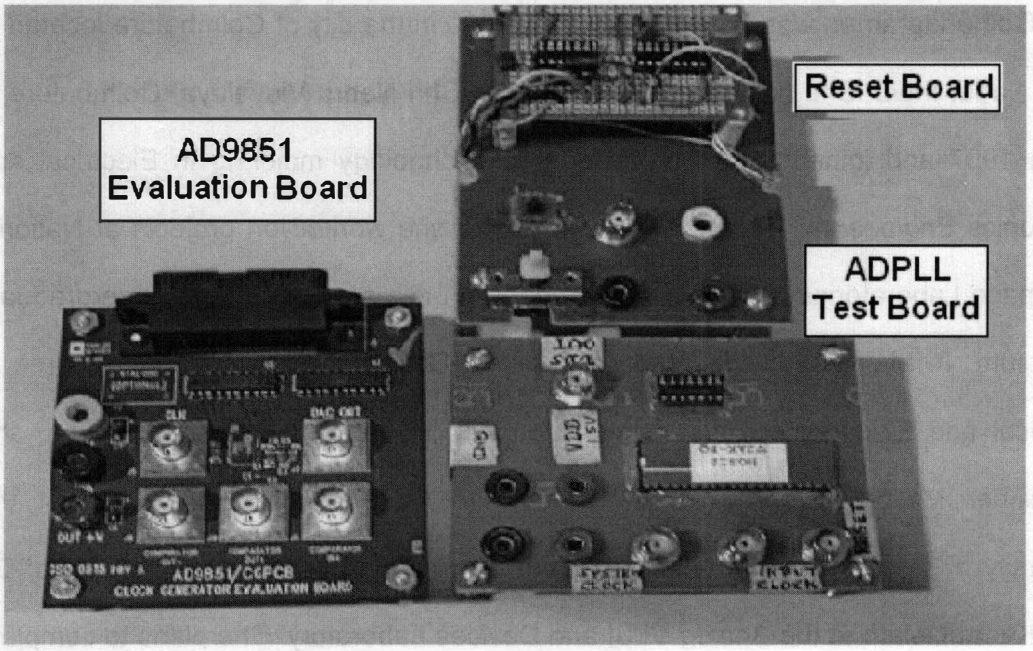



**Figure A.1:** Test Board

**Figure A.2:** Test Measurement Set up

## Vita

Akila Gothandaraman was born on July 16th, 1980 in the city of Coimbatore located in Tamil Nadu, India. She finished her schooling in Shri Nehru Vidyalaya, Coimbatore in March 1997 and joined the PSG College of Technology majoring in Electrical and Electronics Engineering. During the Fall of 2000, she worked on projects at National Aerospace Laboratories and Salem Steel Plant of India as a part of the undergraduate curriculum. In May 2001, she graduated with a Bachelor of Engineering Degree in Electrical and Electronics Engineering. Interested in pursuing higher education, she began her Master's degree in the Department of Electrical Engineering at the University of Tennessee, Knoxville and currently working under the supervision of Dr. Syed Kamrul Islam at the Analog VLSI and Devices Laboratory. She plans to complete her Master's degree in May 2004 with a specialization in Digital and Analog VLSI.