國 立 交 通 大 學

資訊科學與工程研究所

碩 士 論 文

鎖相迴路軟體化之研究

The Study of Software-defined Phase-locked Loop

研 究 生：莊承穎

指導教授：許騰尹 教授

中 華 民 國 九 十 七 年 七 月

鎖 相 迴 路 軟 體 化 之 研 究

The Study of Software-defined Phase-locked Loop

研 究 生：莊承穎　　Student：Chang-ying Chuang

指導教授：許騰尹　　Advisor：Terng-Yin Hsu

國 立 交 通 大 學
網 路 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

July 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

摘要

這篇論文主要實作軟體鎖相迴路的平台，在此平台中結合了 OPEENRISC 和全數位鎖相迴路的特性與功能。實作此平台總共有三個主要設計議題，分別是溝通介面的建築、以智財（ＩＰ）爲基礎的電路設計和軟體演算法的開發。在介面上特別用於同步、通訊和控制。在智財（ＩＰ）爲基礎的電路裡，本論文著重於開發時間數值轉換器（全數位鎖相迴路中的模組），在不需要寄生電容跟寄生電阻的效應的情況下，設計出精準度高達 1ps 的時間數值轉換器。在軟體演算法的執行上，主要是在演算法執行的週期數目作出最佳化，在此平台上所執行的演算法，鎖定頻率跟相位，共需５個週期。

這個平台在重複利用上、製程轉移上和彈性上都有很好的表現，可以去縮減設計成本，特別是時間上的成本。最後，這個平台是完整的建立在 Faraday 90nm 製程下，可以完整運作。

Abstract

This paper is proposed to the platform of software defined phase locked loop. The platform is combined of OPENRISC and the all-digital phase locked loop. There are three major issues in this platform: the interface, the intellectual property (IP) based design and the software algorithm. First, the interface is used for synchronization, communication and controlling. Second, the time-to-digit converter is one of IP in the all-digital phase locked loop. We propose the time-to-digit converter which resolution is 1ps without the parasitical capacitance and parasitical resistance. Finally, the topic of software algorithm is the cycle count. The platform needs 5 cycles to lock the frequency and the phase.

This platform has good performance at the reusability, the process portability, and flexibility. The platform can reduce the design cost, especially at time. We implement this platform on Faraday 90nm process.

Acknowledgement

There are many people who I want to thank. First and foremost I would like to thank my advisors, Dr. Terng-Yin Hsu for advice and guidance. Without the Integration System and Intellectual Property (ISIP) Lab members, I can not finish this work. I would like to thank all ISIP Lab members. Finally, I want to thanks my parents for your support and your love.

Sincerely, Chang-Ying Chuang
August 2008

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1. Thesis Background

The phase locked loop is primarily used in communication applications, such as: the frequency synthesizer, the clock multiplier, the clock recovery circuit, the data recovery circuit, and clock de-skew applications. However, in different applications, the phase locked loop may not be reusable. The phase locked need to redesign for a variety of purposes. It is a challenge to design a flexible phase locked loop.

In modern SoC design, the reusability and the process portability are very important. Because of time-to-market issue, the design cycle can be decreased. Therefore, how to design the circuit efficiently becomes more and more central. The design integrated with the CPU and IP designs is current trend.

1.2. Thesis Motivation

There are different kinds of phase-locked loops such as analog phase-locked loop, digital phase-locked loop, and all-digital phase-locked loop (ADPLL). Especially, the ADPLL is flexible in process migration. However, the ADPLL and the CPU can be the hardware system of software-defined phase-locked loop (SDPLL). The SDPLL inherits both the flexibility of the ADPLL and the software control of the CPU. Therefore, the SDPLL do not need other calculational circuit because the CPU is powerful enough to deal with most calculation. However, how to integrate the ADPLL and the CPU to the SDPLL is a challenge.

1.3. Thesis Organization

The organization of this thesis is as follows:

In chapter 1, we introduce the phase locked loop and the importance of the flexibility.

In chapter 2, we give the basic concept of SDPLL, ADPLL and OPENRISC.

In chapter 3, the proposed system hardware architecture is introduced.

In chapter 4, the proposed system software architecture is introduced.

In chapter 5, some concluding remarks will be derived from this research. Finally, we describe some design issues that needed be further explored in the near future.

Chapter 2
Overview of Software-defined Phase-locked Loop

2.1. Basic Concept Software-defined Phase-locked Loop

Software-defined Phase-locked loop (SDPLL) is flexible to the hardware architecture and the software operation. At the software level, SDPLL only needs to modify the instructions so as to supply different functions. At the hardware level, SDPLL integrates the ALL-Digital Phase-locked Loop with CPU. Therefore, the core of SDPLL is CPU. The core of proposed SDPLL is OPENRISC which will be described in section 2.2.

2.2. OPENRISC

OPENRISC which is the open source is a 32-bit scalar RISC with Harvard microarchitecture, 5 stage integer pipeline, virtual memory support (MMU) and basic DSP capabilities. Default caches are 1-way direct-mapped 8KB data cache and 1-way direct-mapped 8KB instruction cache, each with 16-byte line size. Both caches are physically tagged. By default MMUs are implemented and they are constructed of 64-entry hash based 1-way direct-mapped data TLB and 64-entry hash based 1-way direct-mapped instruction TLB. Supplemental facilities include debug unit for real-time debugging, high resolution tick timer, programmable interrupt controller and power management support. Instruction and Data host interface is WISHBONE SoC Interconnection. The OPENRISC Specification shows in Table 2.1

2.3. ALL-Digital Phase-locked Loop

The all-digital phase-locked loop (ADPLL) consists of the phase detector, the time-to-digit converter (TDC), the controller, the loop filter, the digital controlled oscillator (DCO), and the frequency divider. The basic block diagram is shown in Fig. 2.1. The working flow is as following:

Step 1: The phase detector detects the phase error between the reference clock and the divided clock and outputs the phase error to TDC.

Step 2 : TDC converts the phase error to digits and output the value to controller.

Step 3 : The controller calculates the proper the DCO control word and passes the DCO control word through the loop filter to DCO.

Step 4 : DCO use the DCO control word to oscillate proper clock frequency.

Step 5 : The frequency divider divides the frequency of DCO clock cycle.

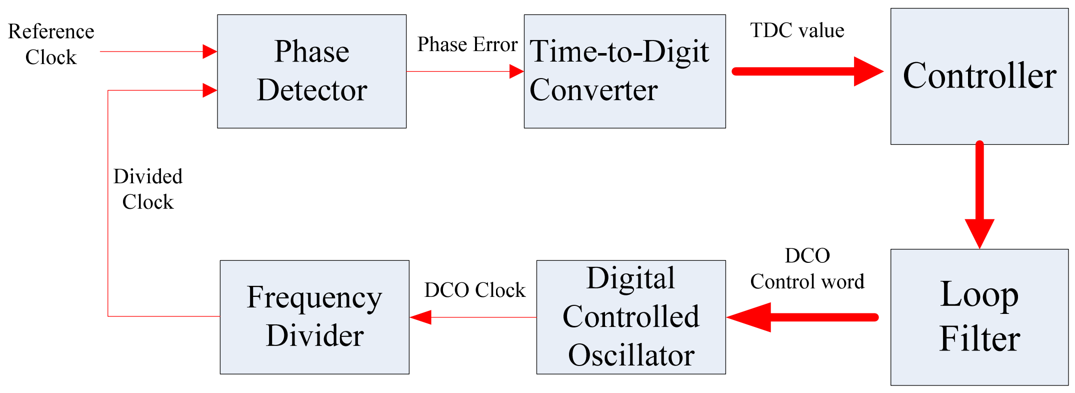


Fig. 2.1 The basic block diagram of ADPLL

OPENRISC	
Instruction length	32bits
Register length	32bits
Number of general purpose registers	32
Support multiplication	Yes
Support division	No
Cycle count of multiplication instruction	4
Cycle count of store instruction	4
Gate count	88000
Host interface	WISHBONE

Table 2.1 The OPENRISC Specification

Chapter 3
The Proposed SDPLL Architecture

3.1. SDPLL Architecture Overview

The proposed SDPLL architecture is shown in Fig. 3.1. There are nine basic modules in the proposed SDPLL. The first is Semi Asynchronous Clock Access. Semi Asynchronous Clock Access applies the entire architecture clock access. The second is Memory Controller. Memory Controller controls Memory and communicates with CPU BUS, Memory and Error Detector. The third is Error Detector. Error Detector includes Time-to-Digit Converter, Phase Detector, Frequency Divider and Pulse Amplifier. The fourth CPU BUS is the bridge of OPENRISC. The fifth is Digital Control Oscillator Interface. Digital Control Oscillator Interface controls DCO and synchronizes the DCO control words and DCO_CLK. The sixth is State controller. State Controller decodes the CPU output message to change states of other modules. The seventh is Digital Control Oscillator. The eighth is 256X32 bits-Memory. The last is OPENRISC, which is described in chapter 2. The details of modules will be illustrated in next section and the system Specification shows in Table 3.1.

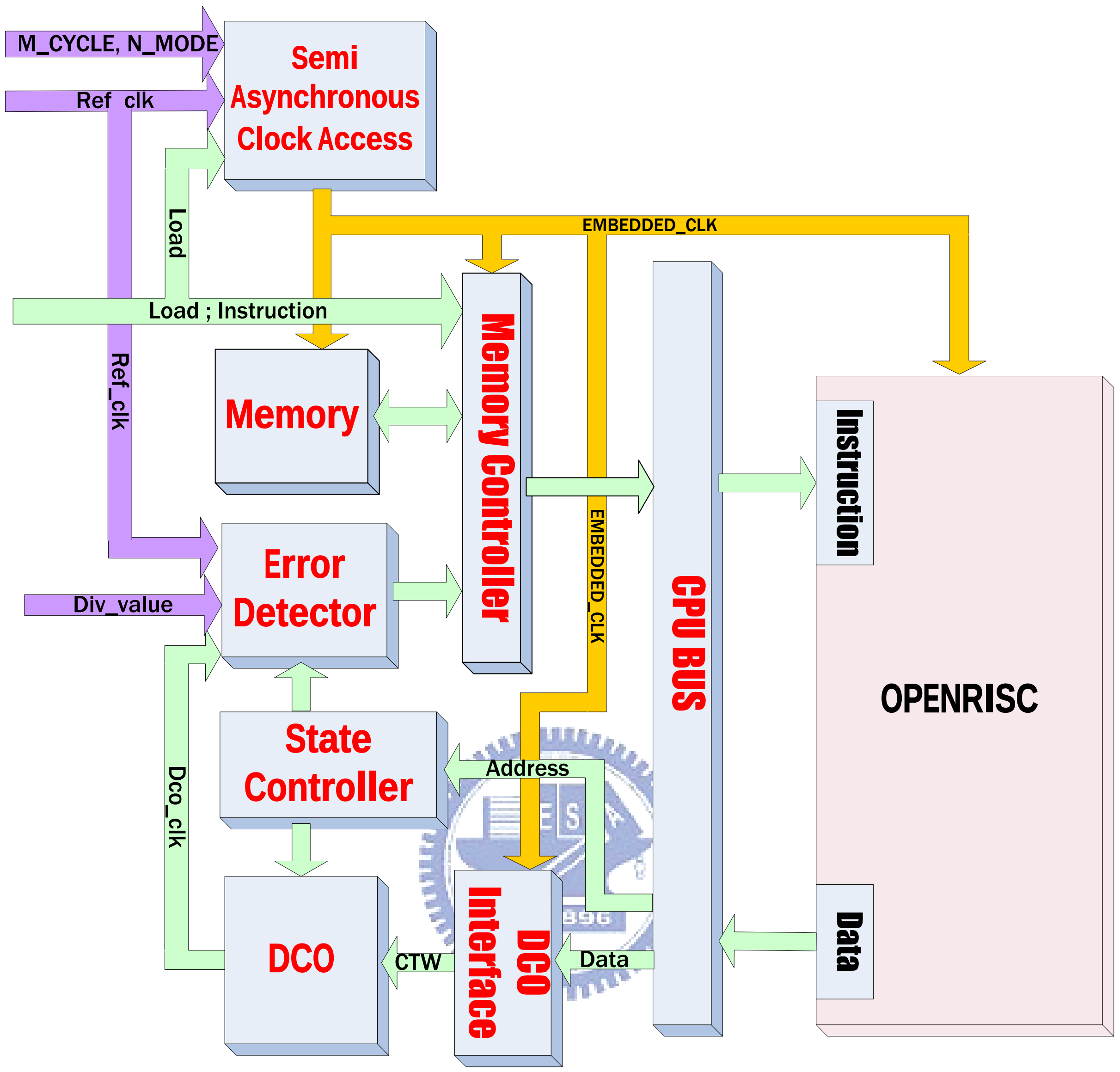


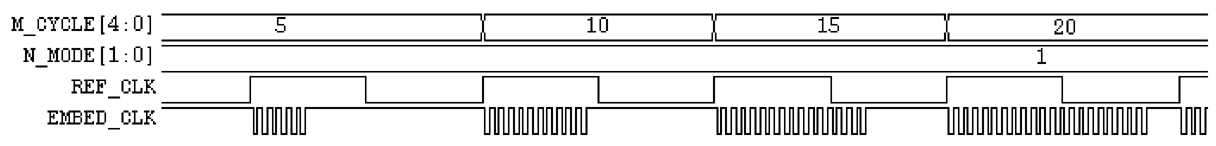
Fig. 3.1 The proposed SDPLL architecture

System Spec	
DCO base frequency	333MHz
DCO resolution	10fs
TDC resolution	1ps
TDC detecting max pulse	4ms
TDC detecting min pulse	2.358ns
Memory	256X32bits
SACA max frequency	263 MHz
SACA min frequency	67 MHz
Reference clock max frequency	6.3 MHz
Reference clock min frequency	50KHz

Table 3.1 The system Specification

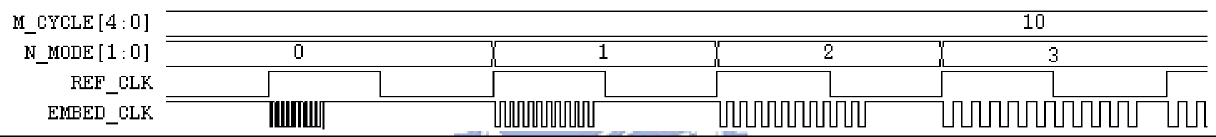
3.2. Semi Asynchronous Clock Access

Semi Asynchronous Clock Access can apply the better performance in circuit noise environment and power consumption. By means of modifying two parameter, it can perform low noise and low power environment. One parameter is operating frequency and the other is operating cycle count. The parameters affect the circuit transition. Controlling circuit transition results in low power environment. Fig. 3.2(a) and Fig. 3.2(b) illustrate the relation of transition tuning parameter. As the clock signal is idle, the clock signal is maintained high because of construct low noise environment.



Cycle count	5	10	15	20

Fig. 3.2(a) The relation of transition tuning the cycle count in 134MHz frequency



Frequency	263MHz	134MHz	90MHz	67MHz

Fig. 3.2(b) The relation of transition tuning the frequency at 10 cycles

Fig. 3.3 shows Semi Asynchronous Clock Access which includes four major parts depend on application. The first is the clock signal synchronizer. The synchronizer is combined with two D flip-flops with the clear pin. If Ref_clk rise, the synchronizer sends the pulse to counters to reset the counter so that EMBED_CLK begins to oscillate. The second is the switch of delay matrix. The basic idea of the switch is a NAND gate. If one input of the NAND gate is high, the NAND gate has the same function as inverter. On the other hand, if one input is low, the output of the NAND gate maintain high. Accordingly, the switch decides EMBED_CLK oscillated or idle based on the two states of the NAND gate. The third is delay matrix to provide the variable clock period. The last is counter combined with comparator to control the operating cycle count. As the counter number equals to M_CYCLE comparator sends a signal to the counter and the switch of delay matrix in order to disable counter and make EMBED_CLK idle.

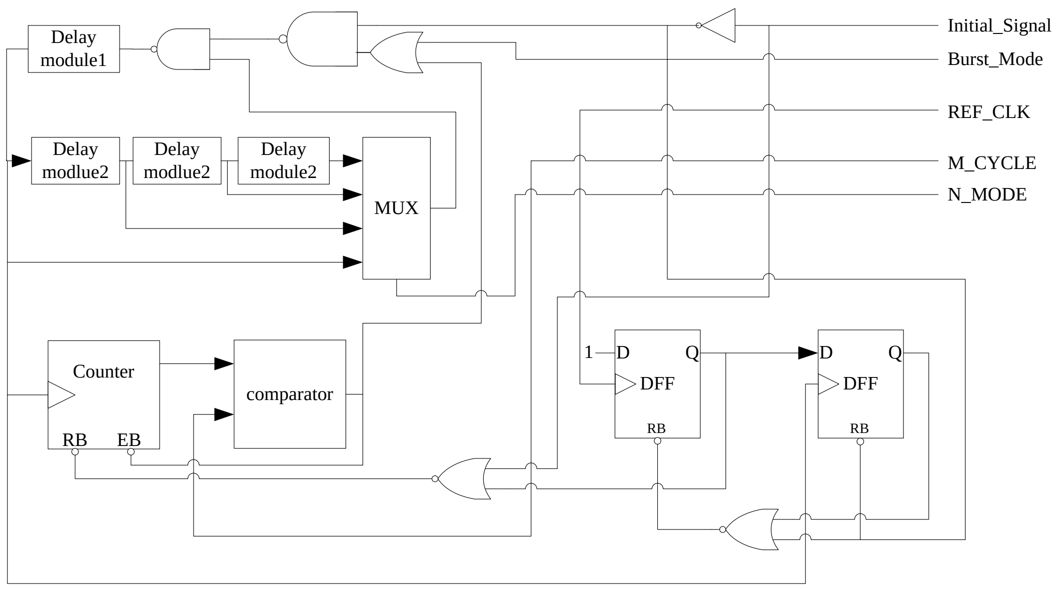


Fig. 3.3 The hardware architecture Semi Asynchronous Clock

3.3. Memory Controller

Memory Controller which plays an important role in the proposed SDPLL is shown in Fig. 3.5. CPU BUS, Memory and Error Detector communicate with different protocols through Memory Controller. Memory Controller also has strong connection with the proposed instruction set architecture. The proposed instruction set architecture (ISA) will be discussed in chapter 4.

The flow chart of the propose state of Memory Controller is shown in Fig. 3.4. When the system resets, Memory Controller is in the initial state. In the meantime, Memory Controller initializes the address counter. As the load signal high, Memory Controller changes the state to the load state. In this state, Memory Controller loads executing instructions to Memory until load signal falls. After that, Memory Controller changes the state to the transition state. In the transition state, Memory Controller receives the error value as Error Valid is high. After that, Memory Controller changes the state to the algorithm state. In order to send this instruction to CPU BUS, Memory Controller reads instruction from memory and combines the error value to instruction in the algorithm state. After sending instructions, Memory controller changes the state back to the transition state. Subsequently, memory controller repeats the above flow again.

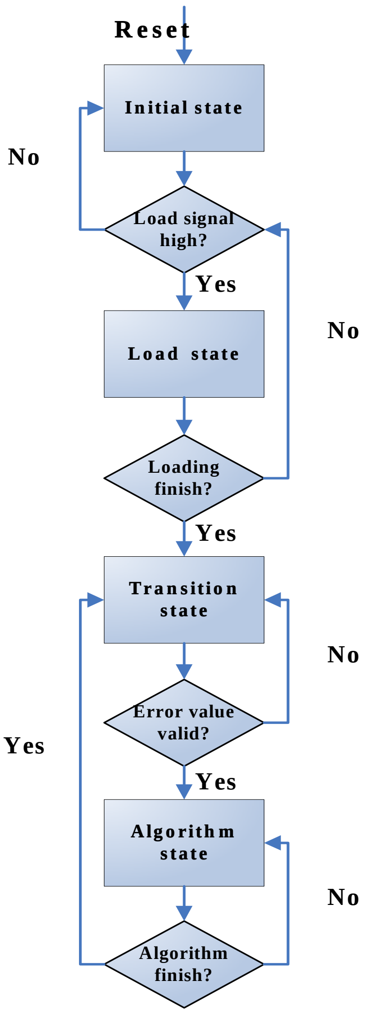


Fig. 3.4 The flow of the memory controller

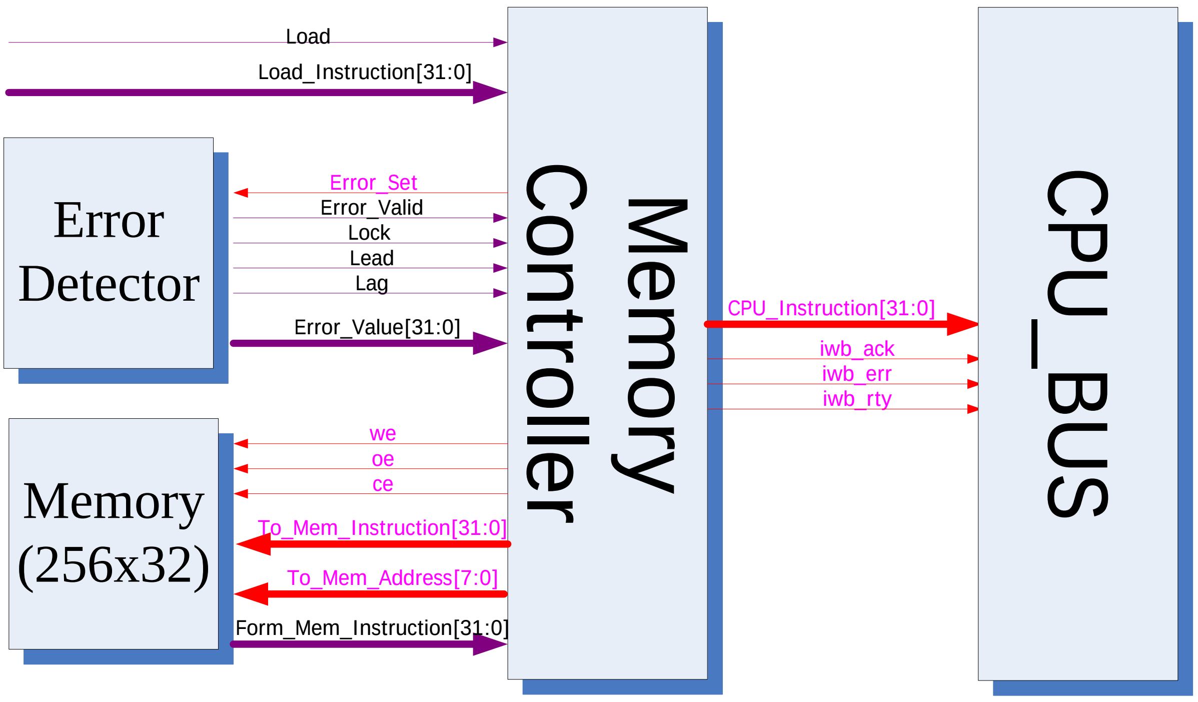


Fig. 3.5 The pin assignment of the memory

3.4. Error Detector

Error Detector which is shown in Fig. 3.6 has four functional blocks such as Time-to-Digit Converter (TDC), Phase Detector, Frequency Divider and Pulse Amplifier with One Pulse Lock. The core of Error Detector is TDC. The objects which are measured by TDC are Ref_Clk and the phase error between Ref_Clk and the Div_Clk. TDC will be described in section 3.5. However, Error Detector can not only detect the error but also divide the frequency of clock cycle. The details of functional blocks will be illustrated in next section.

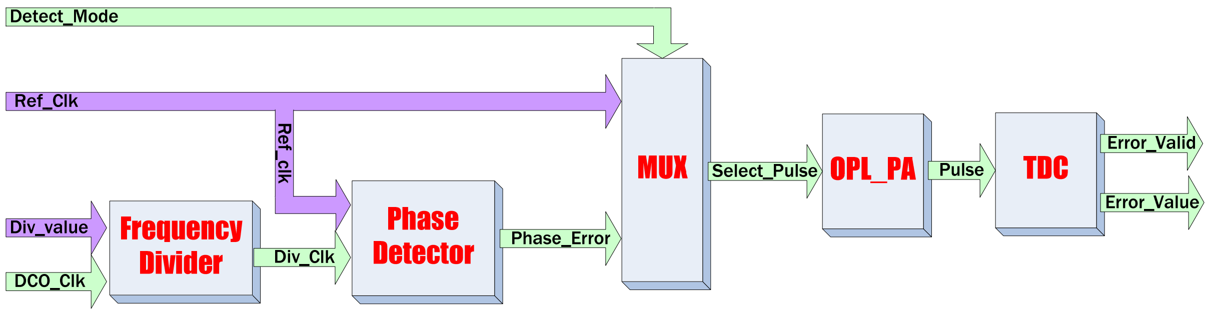


Fig. 3.6 The block architecture of Error Detector

3.4.1. Phase Detector

Phase Detector converts the difference of Ref_Clk and Div_Clk to the pulse and judges which clock signal is lead to another.

3.4.2. Frequency Divider

Frequency Divider also divides the frequency of DCO clock cycle. The core of Frequency Divider is the counter which driven by DCO clock cycle.

3.4.3. Pulse Amplifier with One Pulse Lock

There are two serious problems in the input of TDC such as the pulse account and the pulse width. First, Pulse Amplifier which is showed in Fig. 3.7 with One Pulse Lock modifies the narrow pulse to meet the circuit requirement so as to prevent the input width violation of TDC. If the pulse is wide enough to the circuit requirement, Pulse Amplifier with One Pulse Lock keeps the pulse in the original width. Second, Pulse Amplifier with One Pulse Lock filters the pulse after the first pulse has come. The reason is that TDC accumulates all pulses width until Error_set rises.

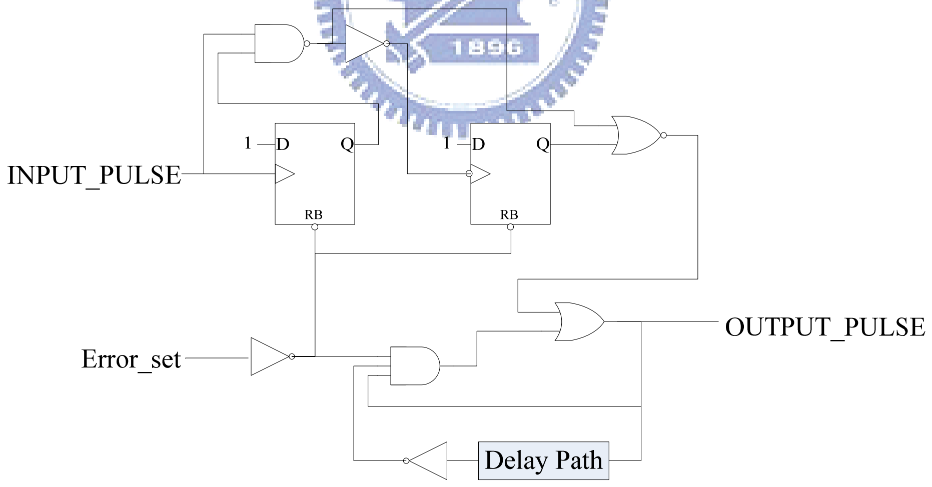


Fig. 3.7 The hardware architecture of Pulse Amplifier with One Pulse Lock

3.5. Time-to-Digit Converter

The proposed Time-to-Digit Converter (TDC) which is shown in Fig. 3.8 consists of two major modules, Gate delay TDC and Differential delay TDC. Gate delay TDC has different resolution form Differential delay TDC. The resolution of Gate delay TDC is 10ps but the resolution of Differential delay TDC. Although Differential delay TDC has higher resolution than Gate delay TDC, Differential delay TDC needs ten times gate counts to Gate delay TDC. Therefore, the proposed TDC combine Gate delay TDC and Differential delay TDC to minimize the total gate count. The resolution of the proposed TDC is 1ps but the gate count of the proposed TDC is double gate count to Gate delay TDC. Differential delay TDC decides the least digit of TDC_OUT and Gate delay TDC decides the other digits. Thus, the resolution of proposed TDC is 1ps.

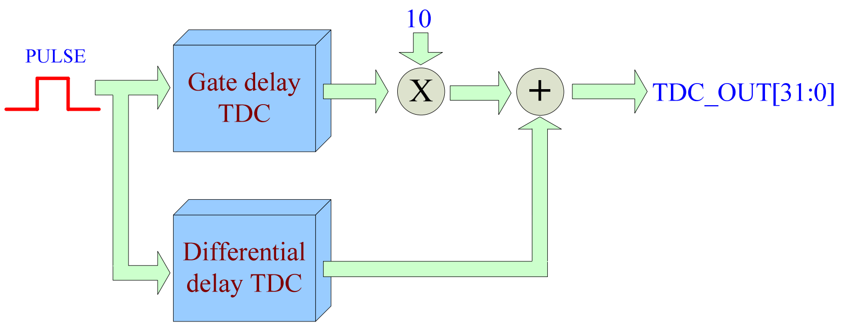


Fig. 3.8 The functional blocks of the proposed TDC

3.5.1. Gate delay TDC

Gated delay TDC which is shown in Fig. 3.9 is comprised of four functional blocks. The four functional blocks are TDC_CHAIN, Latch Chain Buffer, Counter and TDC Decoder.

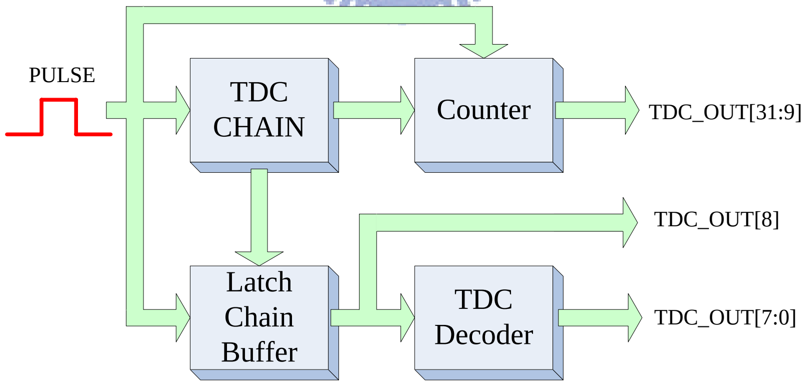


Fig. 3.9 The functional blocks of Gate delay TDC

First, TDC_CAHIN which is shown in Fig. 3.10 is composed of 255 inverters and 1 and gate. Each inverter connects another inverter. Thus, the inverters constitute a delay chain. One input of the and gate connects the end of the delay chain and the output of the and gate connects to the start of the delay. The delay path and the and gate make up the delay ring. The purpose of the and gate is a switch to control the delay ring. When the other input of the switch is high, the switch does the same function as the buffer. On the other hand, the switch clears the delay ring. The propagate delay of the inverter is 10ps on Faraday 90nm process. Therefore, the resolution Gate delay TDC is 10ps.

Second, Latch Chain Buffer which is shown in Fig. 3.10 is composed of the D latches. As the pulse is high, Latch Chain Buffer stores the state of TDC CHAIN. On the other hand, Latch Chain Buffer keeps the storing information until Error_Set rises. We choose the D latch to be the unit of Latch Chain Buffer due to the issue of the latching time.

Third, Fig. 3.10 illustrates TDC Decoder. The basic idea of TDC Decoder is finding out the position of the transition in the TDC Chain. Thus, we choose the prienc decoder at Deignware library to accomplish TDC Decoder.

Last, Fig. 3.10 illustrates the counter. The output of the last inverter of TDC CHAIN triggers the counter. The counter combines the multiplexer inside because of the dead zone. Since the pulse goes down, TDC CHAIN will not be clear immediately. Thus, counter operates at expected case. The situation results in wrong value of TDC_OUT. Therefore, the counter which combines the multiplexer inside can prevent this situation.

As a result, Gate delay TDC has 10-ps resolution because of the inverter. As the process upgrades, the resolution of Gate delay TDC gets more higher.

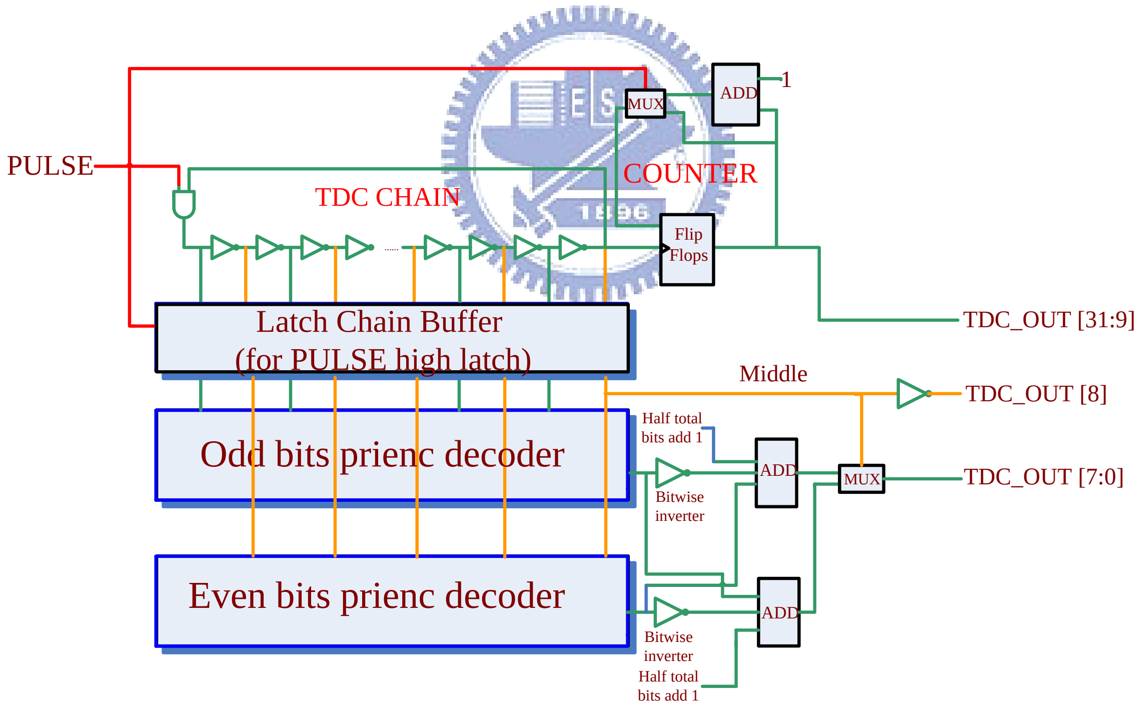


Fig. 3.10 The hardware architecture of Gate delay TDC

3.5.2. Differential delay TDC

Fig. 3.11 shows the functional blocks of Differential delay TDC. The functional blocks are DELAY PULSE, EVEN GROUP, ODD GROUP and Differential Decoder. Some parts of Differential delay TDC are similar to Gate delay TDC. Differential delay TDC uses the different delay pulse to decode the value of the different delay pulse. Because of the ten values, we can convert least digit of TDC_OUT. Therefore, the resolution of Differential delay TDC is 1ps.

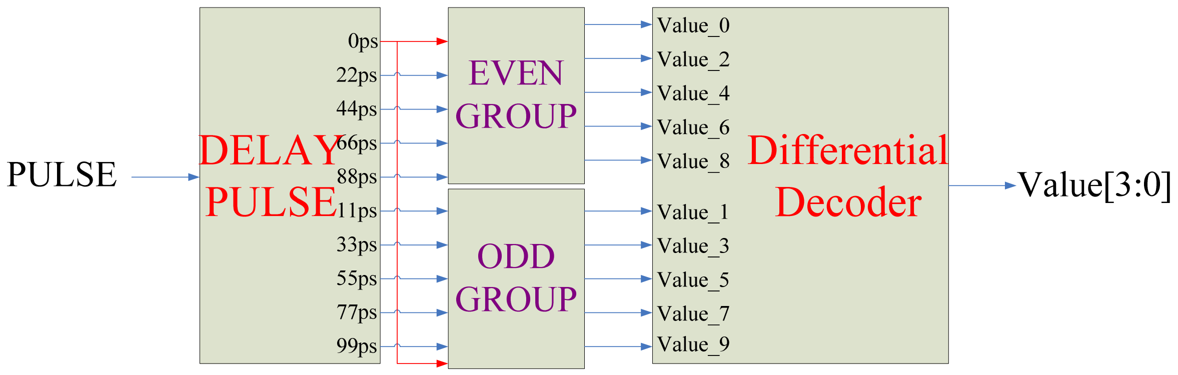


Fig. 3.11 The function blocks of differential delay TDC

First, the DELAY PULSE produces ten different delay pulses. Fig. 3.12 shows that different delays are 11ps, 22ps, 33ps, 44ps, 55ps, 66ps, 77ps, 88ps and 99ps. The delay are 11ps, 33ps, 55ps, 77ps, and 99ps belong to ODD GROUP. On the other hand, The delay are 0ps, 22ps, 44ps, 66ps, and 88ps belong to EVEN GROUP. DELAY PULSE has nine inverters which have 11-ps resolution. The purpose of choosing 11-ps resolution is that 11ps minus 10ps leaves 1ps. Therefore, Differential delay TDC uses the 10-ps resolution inverter and the 11-ps resolution inverter to reach the 1-ps resolution.

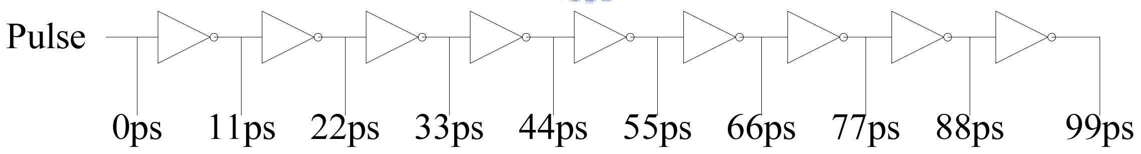


Fig. 3.12 The hardware architecture of the delay pulse module

Second, Fig. 3.13 and Fig. 3.14 illustrate EVEN GROUP and ODD GROUP. EVEN GROUP has five EVEN SUB TDC (ESTDC). ODD GROUP has five ODD SUB TDC (OSTDC). The difference between ESTDC and OSTDC is the switch. One is the and gate; the other is the and gate with one inverse pin. ESTDC and OSTDC are similar with Gate delay. TDC .CHAIN of ESTDC and OSTDC is the chain with twenty 10-ps resolution inverter. Latch Chain Buffer is the latch which stores the chain information. When Pulse is high, Latch Chain Buffer stores the information. On the other hand, Latch Chain Buffer maintains the storing information. Each output pin of Latch Chain Buffer connects to the nor gate with the next to output pin. The output of the nor gate is the input the prienc decoder. Finally, the prienc decoder outputs the value.

However, ESTDC and OSTDC have one difference from Gate delay TDC. At Gate delay TDC, the same pulse supplies the chain and buffer. At ESTDC and OSTDC, the original pulse supplies the buffer and the delayed pulse supplies the chain. In this action, ESTDC and OSTDC output the value the diminished pulse. For example, if the 22ps-delayed pulse is the input of ESTDC, ESTDC outputs the value of 22ps-diminished pulse. Similarly, if the 11ps-delayed pulse is the input of OSTDC, OSTDC outputs the value of 11ps-diminished pulse.

Finally, the ten values form EVEN GROUP and ODD GROUP has regular patterns. We use the regular patterns to decide the TDC_OUT [3:0] by the differential decoder shown in Fig. 3.15.

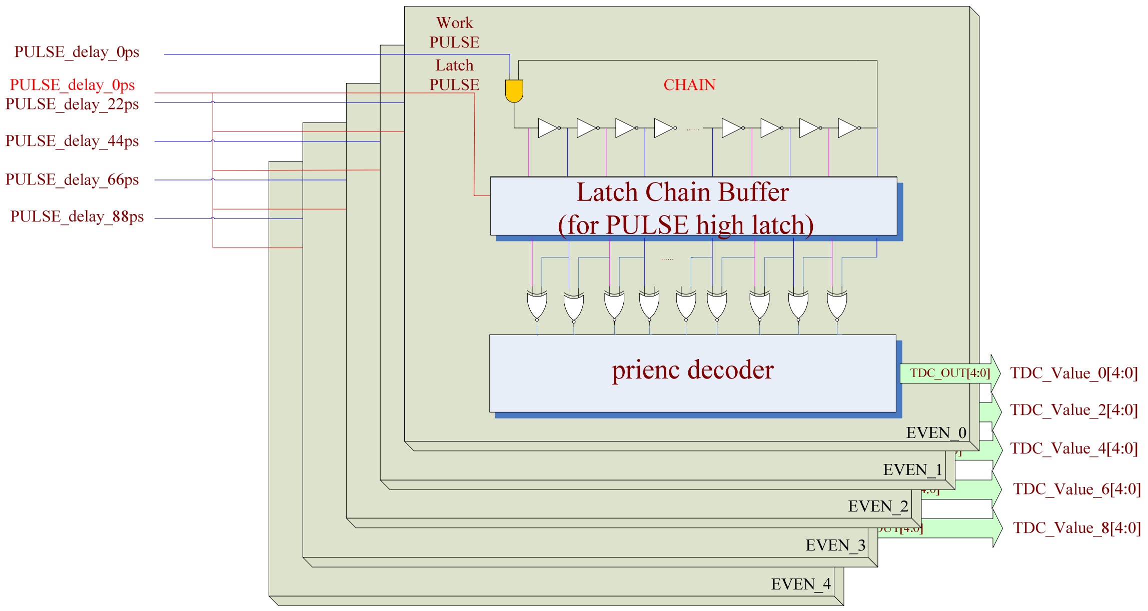


Fig. 3.13 The hardware architecture of EVEN GROUP

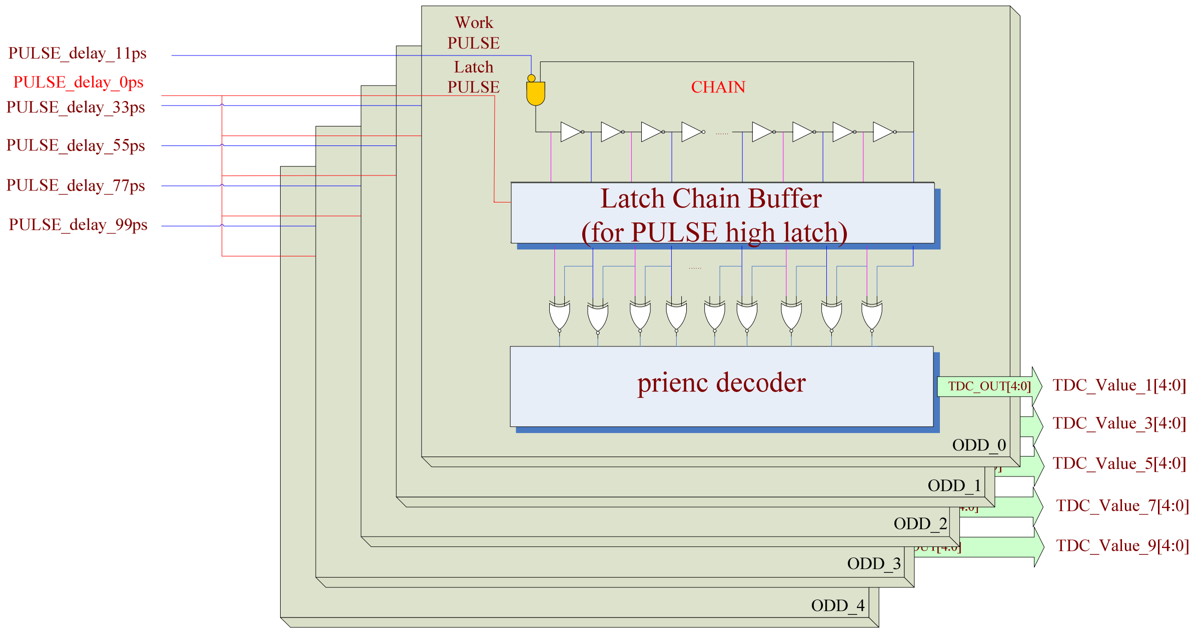


Fig. 3.14 The hardware architecture of ODD GROUP

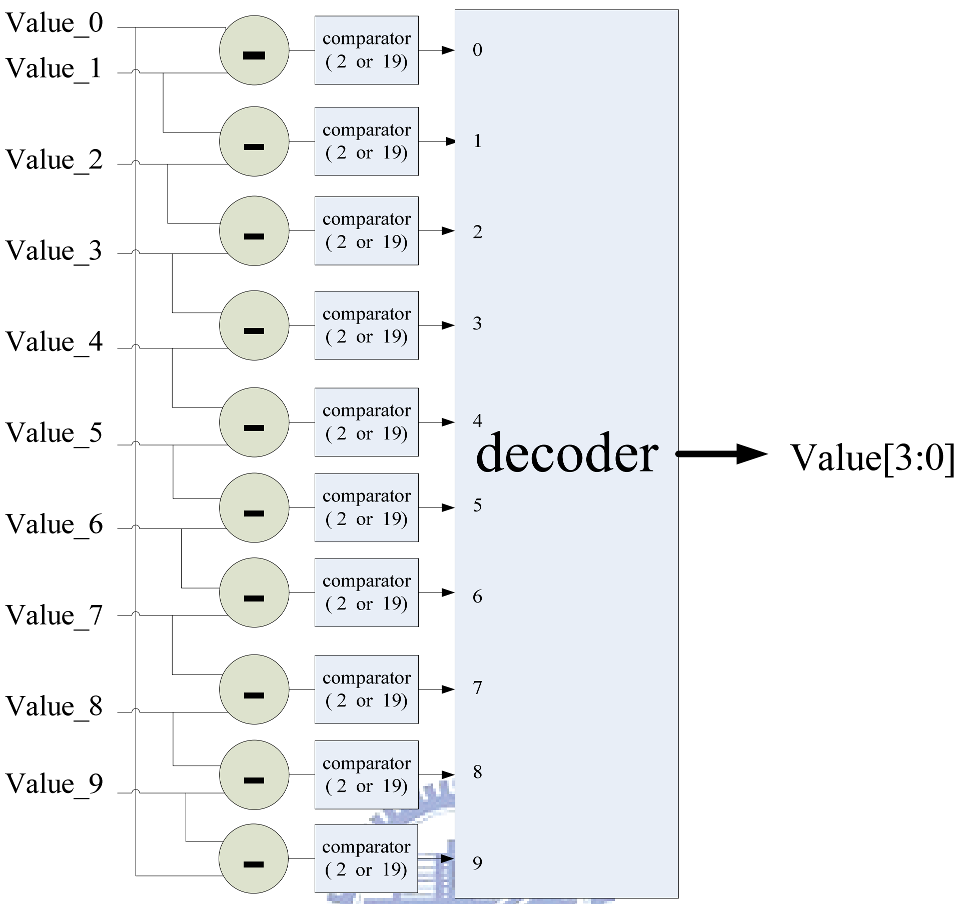


Fig. 3.15 The hardware architecture of differential decoder

3.6. CPU Bus

The connection of CPU is the instruction pins, the data pin and the WISHBONE control signals.

3.7. Digital Control Oscillator Interface

Fig. 3.16 and Fig 3.17 show the state diagram of Digital Control Oscillator Interface and pins of Digital Control Oscillator Interface. At the Coarse Frequency state, Digital Control Oscillator Interface outputs the control word to DCO so as to lock the frequency. At the Coarse Phase state, the control word is for phase locking. At the Coarse Transition state, the control word is the same as the coarse Frequency state.

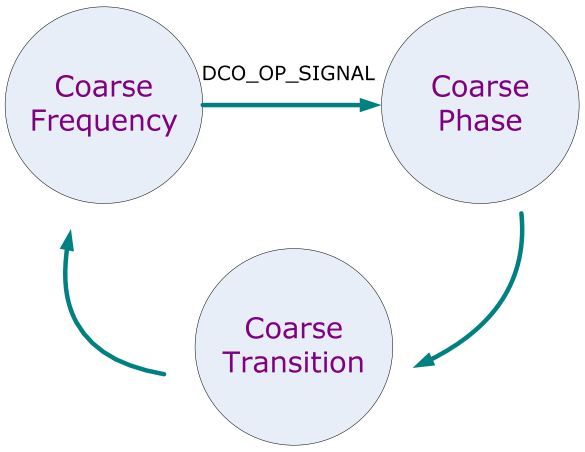


Fig. 3.16 The finite state machine of Digital Control Oscillator Interface

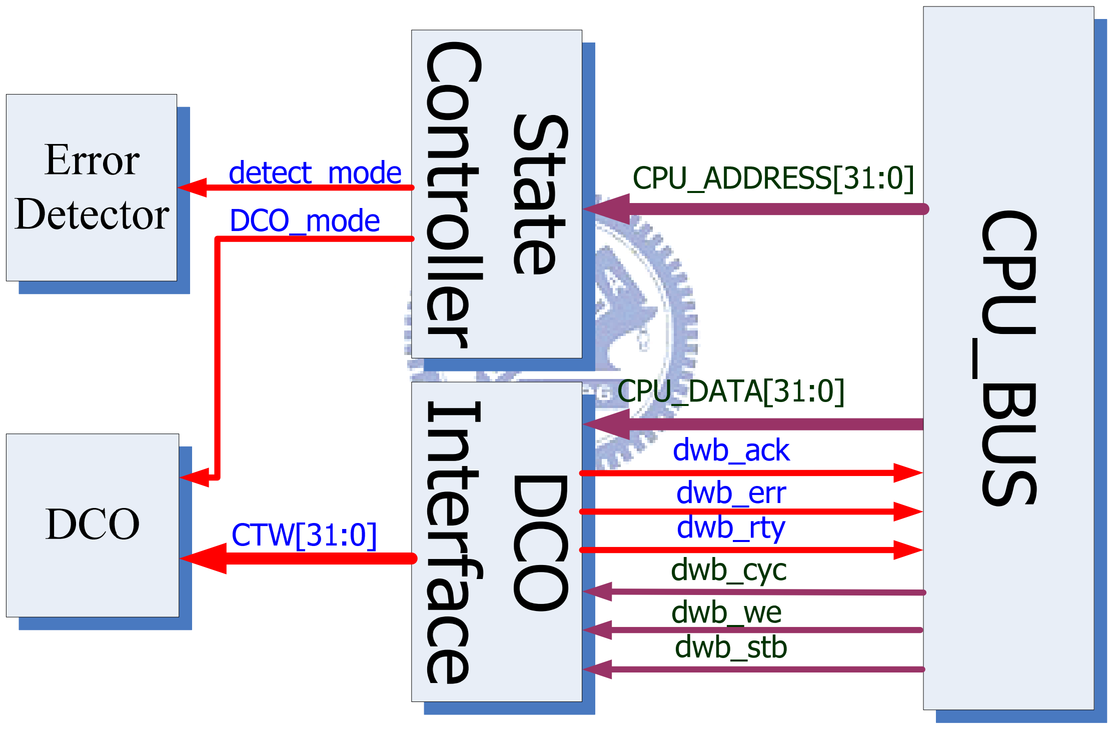


Fig. 3.17 Pins of Digital Control Oscillator Interface

The DCO interface has two different the clock system. One is the CPU clock. The CPU clock supplies blocks which communicate with CPU. The other is the DCO clock. The DCO clock supplies the finite state machine of the DCO interface because the control word needs to synchronize with the DCO clock.

3.8. Digital Control Oscillator

The proposed DCO is a behavior model. The basis frequency is 333 MHz. Each step of DCO is 10fs.

Chapter 4
The proposed SDPLL Software

4.1. OPENRISC ISA Overview

The OPENRISC instruction set which shows in Fig. 4.1 includes the following principal features. First, Simple and uniform-length instruction formats featuring five Instruction Subsets. Second, OPENRISC Basic Instruction Set (ORBIS32/64) with 32-bit wide instructions aligned on 32-bit boundaries in memory and operating on 32-bit and 64-bit data. Third, OPENRISC Vector/DSP eXtension (ORVDX64) with 32-bit wide instructions aligned on 32-bit boundaries in memory and operating on 8-, 16-, 32- and 64-bit data. Last, OPENRISC loating-Point eXtension (ORFPX32/64) with 32-bit wide instructions aligned on 32-bit boundaries in memory and operating on 32-bit and 64-bit data. The Table 4.1 shows difference between subsets. The proposed of SDPLL uses ORBIS32 to be instruction set.

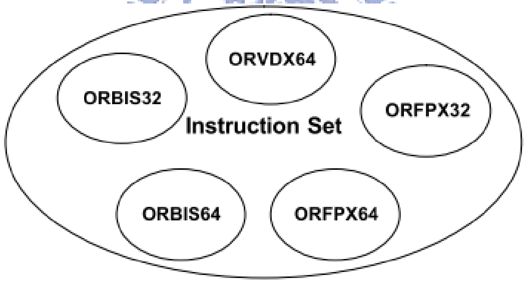


Fig. 4.1 OPENRISC instruction set

ORBIS32	32-bit instructions
ORBIS64	64-bit instructions
ORFPX32	Single-precision floating instruction
ORFPX64	Double-precision floating instruction
ORVDX64	Vector instruction

Table 4.1 Lists of RISC instruction set

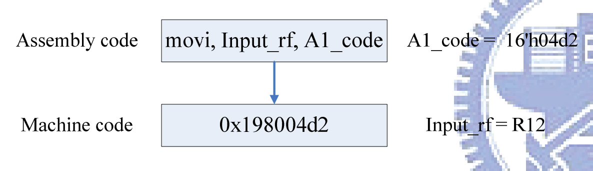
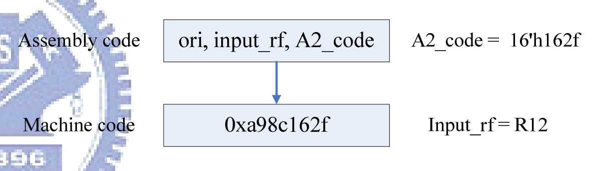
4.2. The Proposed ISA

The proposed ISA has three major ideas: the input instructions, the block jump instruction and the output instructions.

4.2.1. Input Instructions

The purpose of the input instructions is to pass the value to OPENRISC. We choose the ori instruction (Format: ori rD,rA,K. The immediate value is zero-extended and combined with the contents of general-purpose register rA in a bit-wise logical OR operation. The result is placed into general-purpose register rD) and the movi instruction (Format: movhi rD,K. The 16-bit immediate value is zero-extended, shifted left by 16 bits, and placed into general-purpose register rD.) to be input instruction.

Fig_4.2(a) and Fig. 4.2(b) show the machine code and the assembly code of the instruction. The memory stores the input instructions whose machine code are 0x198004d2 and 0xa98c162f. As the memory controller which is described in the section 3.3 reads instructions from memory, the memory controller detect the instruction whether the instruction is 0x198004d2 or 0xa98c162f. or not. If the instruction is 0x198004d2, the memory controller replaces the lower 16-bit part of the instruction to the higher 16-bit part of the instruction. Similarly, if the instruction is 0xa98c162f, the memory controller replaces the high 16-bit part of the instruction to the lower 16-bit part of the instruction. The above description shows in Fig. 4.3.

 Assembly code movi, Input_rf, A1_code A1_code = 16'h04d2 Machine code 0x198004d2 Input_rf = R12	 Assembly code ori, input_rf, A2_code A2_code = 16'h162f Machine code 0xa98c162f Input_rf = R12
Fig. 4.2(a) The input instruction of lower 16bits	Fig. 4.2(b) The input instruction of higher 16bits

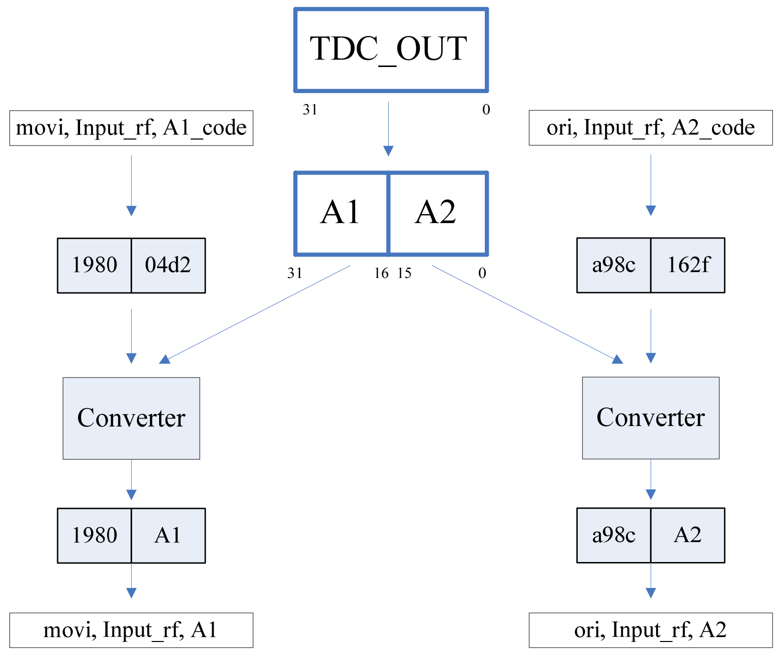


Fig. 4.3 The input instruction working flow

4.2.2. Block Jump Instruction

The instruction memory which is 256 x 32 bits is distributed into 16 blocks which consists of 16 32-bit instructions. We use the block to be a algorithm operation because most algorithm operation need less than 16 instructions.

The block jump instruction especially focuses on the clock phase issue whether REF_CLK is lead to DIV_CLK or not. Because the different issue has the different operation, we use the block jump instruction which shows in Fig. 4.4. If REF_CLK is lead to DIV_CLK and the memory controller reads the block jump instruction, the reading address is changed to the begin address of the num_block_read block. On the other hand, if REF_CLK is lag to DIV_CLK and the memory controller reads the block jump instruction, the reading address is changed to the begin address of the num_block_lag block. Fig. 4.5 shows the jumping process.

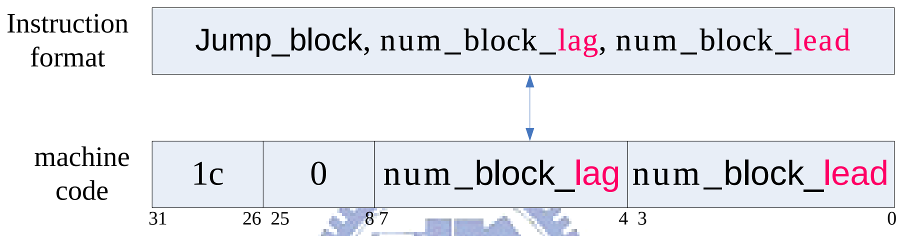
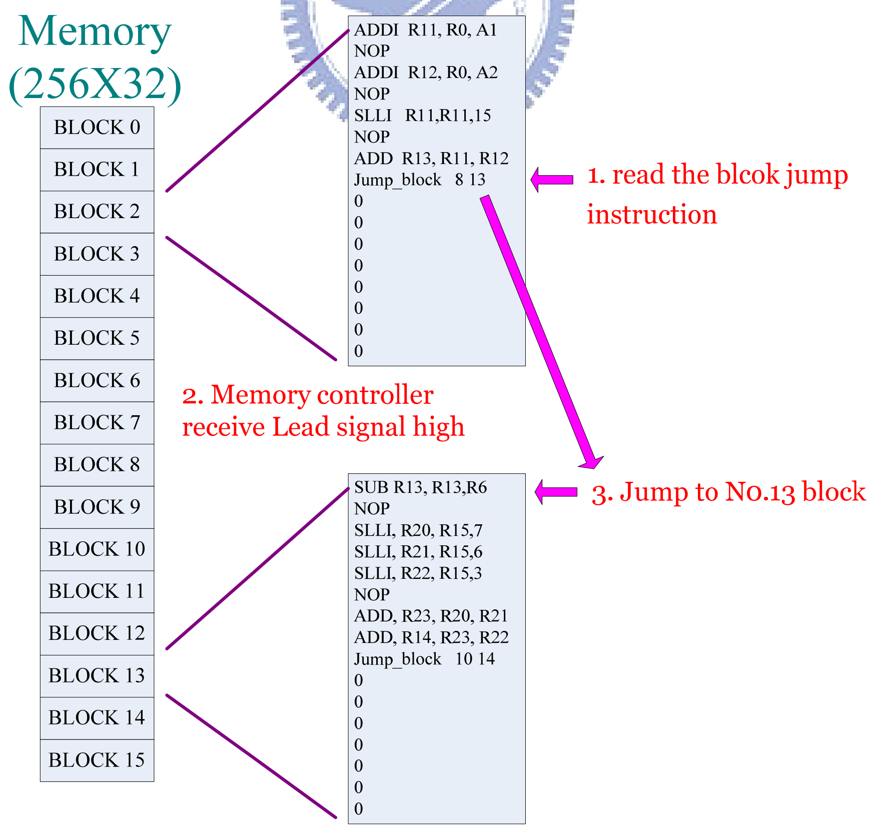


Fig.4.4 The block jump instruction



```
ADDI  R11, R0, A1
NOP
ADDI  R12, R0, A2
NOP
SLLI   R11,R11,15
NOP
ADD  R13, R11, R12
Jump_block   8 13
0
0
0
0
0
0
0
0
```

```
SUB R13, R13,R6
NOP
SLLI, R20, R15,7
SLLI, R21, R15,6
SLLI, R22, R15,3
NOP
ADD, R23, R20, R21
ADD, R14, R23, R22
Jump_block   10 14
0
0
0
0
0
0
0
```

Fig. 4.5 The working flow of the block jump instruciton

4.2.3. Output Instruction

The output instruction is the way of outputting the information of OPENRISC such as the DCO control word and the state control word. We choose the store instruction of ORBIS32. The store instruction has two parts to be information: data and address. Storing data is the DCO control word. Storing address is the state control word. Fig. 4.6 shows the machine code of the store instruction. Target Rf is a register which maintains the value of the DCO control word. Infor which is shown at Table 4.2 is the input of the state controller.

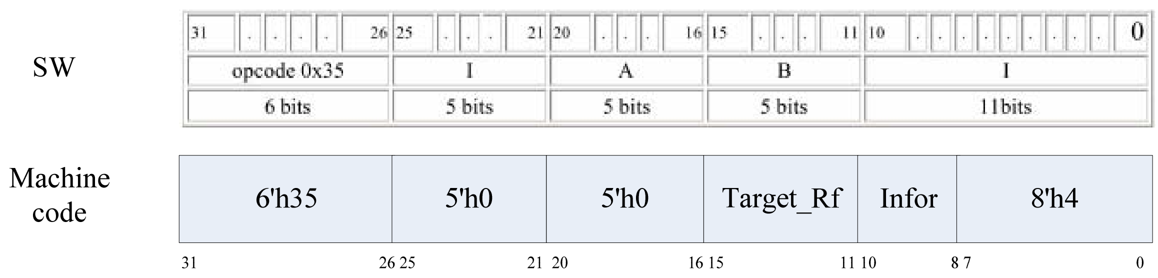


Fig. 4.6 The output instruction

Infor	Pin assignment	State	
[0]	DCO_mode	0	Frequency Lock operation
		1	Phase Lock operation
[1]	detect_mode	0	Frequency detection
		1	Phase Error detection
[2]	Tracking_mode	0	coarse tracking
		1	fine tracking

Table 4.2 The pin assignment of the state controller

4.3. Proposed SDPLL Algorithm

The proposed SDPLL has two basic states: the coarse frequency state and the coarse phase state. At the coarse frequency state, we use the TDC to get the half value of the period of Ref_Clk. OPENRISC converts the proper value because of the different resolution between TDC and DCO. At the coarse phase state, the action shows in Fig. 4.7(a) and Fig. 4.7(b). We use the TDC to get the value of the phase error between Ref_Clk and Div_Clk . The memory controller jumps to the proper block according to lead and lag signal. We assumes the period of Ref_clk as A and the phase error is E. In the lead case which is shown in Fig. 4.7(a), DCO changes the period of Div_clk to A+E in order to lock the phase. On other hand, in the lead case which is shown in Fig. 4.7(b), DCO changes the period of Div_clk to A-E.

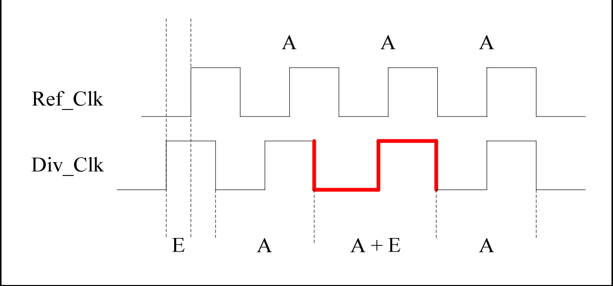
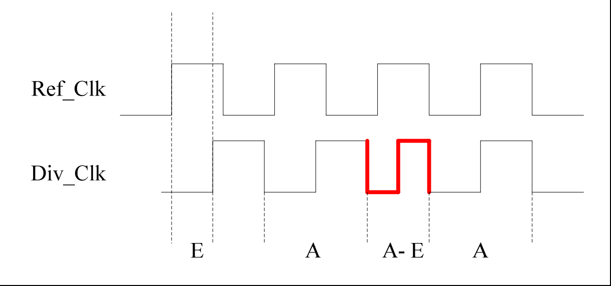
	

Fig. 4.7(a) When Div_Clk is lead to Ref_Clk	Fig. 4.7(b) When Div_Clk is lag to Ref_Clk

4.4. Simulation Result

Fig. 4.8 shows that the platform needs how many cycles to lock. The platform needs 3 cycles to lock frequency and 2.5 cycles to lock phase. Total cycle count to lock is 6. Fig. 4.9 (a) and Fig. 4.9 (b) show the waves of different clock period. Finally, we show comparison with other phase lock loops in Table 4.3.

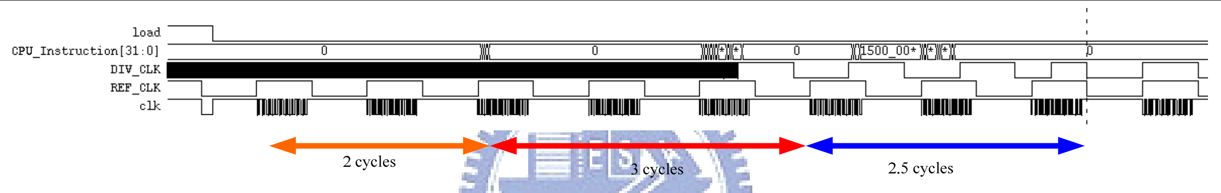


Fig. 4.8 The cycle count of platform to lock

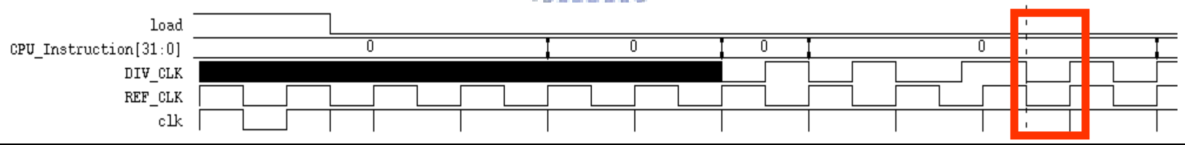


Fig. 4.9(a)The period of Ref_Clk is 6.3MHz

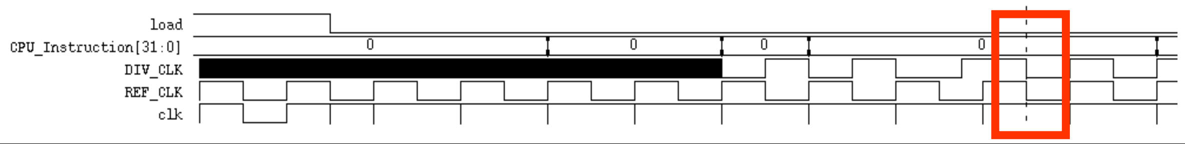


Fig. 4.9(b)The period of Ref_Clk is 50KHz

Performance Parameter	This work	06[2]	ISSCC’04[3]	JSSC’05[4]
Process	90nm CMOS	90nm CMOS	90nm CMOS	0.18um CMOS
Input Range	50KHz~6.3MHz	200KHz ~33MHz	30 KHz ~65 MHz	1KHz ~ 50MHz
Flexibility	Yes	No	No	No
Max Lock time	6	6	>150	<50

Table 4.3 Comparison with other phase lock loops

Chapter 5
Conclusion and Future Work

5.1. Conclusion

The proposed SDPLL has two levels for development. One is hardware system level which is comprised with ADPLL and OPENRISC. The other is the software level. The proposed SDPLL is flexible not only on the software level but also on hardware level. As the hardware upgrades, the proposed SDPLL just need to modify the software code. As a result, the proposed SDPLL can supply the flexible environment.

5.2. Future Work

The following topics to extend the work can be proposed.

I. The proposed SDPLL will combine the G.C.D (Greatest common divisor) application to recover NRZ (none return zero) clock signal.

II. The proposed SDPLL will lock the frequency-divided clock by more complicated instructions.

III. Enhancing the resolution of DCO and TDC is important issue.

References

[1] Terng-Yin Hsu, Bai-Jue Shieh, Chen-Yi Lee” An all-digital phase-locked loop(ADPLL)-based clock recovery circuit” Solid-State Circuits, IEEE Journal of Volume 34, Issue 8, Aug. 1999 Page(s):1063-1073

[2]. Li Jyun-Rong, Hsu Terng-Yin” The Study of All Digital Phase-Locked Loop (ADPLL) and its Applications” Thesis CS, NCTU 2006.

[3]. J. Lin, B. Haroun, T. Foo, J.-S. Wang, b. Helmick, S. Randall, T. Mayhugh, C. Barr and J. Kirkpartick, “A PVT Tolerant 0.18 MHz to 600 MHz Self-Calibrated Digital PLL in 90 nm CMOS Process, “ in Dig. Tech. Papers ISSCC’ 04, Feb. 2004, pp. 488-489.

[4] Ching-Che Chung, Chen-Yi Lee, “An all-digital phase-locked loop for high-speed clock generation” *IEEE Journal of Solid-State Circuits*, Vol38,pp.347-351, Feb.2003

[6] “*OpenRISC 1200 IP Core Specification*” Rev. 0.7, Sep 6, 2001

[7] “*OpenRISC 1000 Architecture Manual* “July 13, 2004