國 立 交 通 大 學

資訊科學與工程研究所

碩 士 論 文

以 ARM 為基礎之鎖相迴路軟體化平台

ARM-based Software-defined Phase-locked Loop Platform

研 究 生：蘇楷書

指導教授：許騰尹 教授

中 華 民 國 九 十 九 年 八 月

以 ARM 為 基 礎 之 鎖 相 迴 路 軟 體 化 平 台

ARM-based Software-defined Phase-locked Loop Platform

研　究　生：蘇楷書	Student：Kai-Shu Su
指導教授：許騰尹	Advisor：Terng-Yin Hsu

國 立 交 通 大 學

資 訊 科 學 與 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

Hsinchu, Taiwan, Republic of China

中華民國九十九年八月

摘要

本論文提出了以軟體控制之鎖相迴路平台。此平台透過 Advanced Microcontroller Bus Architecture (AMBA) 整合 AndesCore CPU 及誤差偵測器 (Error Detector)、數位控制振盪器 (Digital Controlled Oscillator, DCO)等矽智產 (IP)。使用 C/C++等高階語言實作「鎖相迴路演算法」，利用 CPU 的運算能力執行此演算法控制其他 IP。所有的 IP 皆以標準元件數位電路方式實作，並透過修改演算法之方式來更動平台之規格，以降低設計成本。

Abstract

This thesis is proposed to the ARM-based platform of software-defined phase-locked loop. This platform is combined of AndesCore CPU and silicon intellectual property (IP) such as Error Detector and Digital Controlled Oscillator (DCO), all IPs and CPU are integrated with Advanced Microcontroller Bus Architecture (AMBA). The proposed「phase-locked loop algorithm」is implemented by high level language such as C/C++ and executed by CPU to control the IPs, all IPs are implemented by standard-cell-based design. The specification of the platform can be changed by modifying the algorithm to reduce design cost.

誌謝

從碩士班入學到畢業的這段期間，最重要的就是有家人的包容與支持，才能讓我沒有顧慮地順利完成學業。而能完成這篇論文，首先要感謝我的指導教授 許騰尹老師對我的肯定。在老師的指導之下，讓我瞭解研究所學的不只是專業上的能力，更重要的是定義並解決問題的態度與方法，這些能力會跟著我一輩子。也感謝老師願意給我機會，讓我進入 ISIP 實驗室和大家一起學習、成長。

而在研究的過程中，最要感謝的是賴榮欽學長，無論是專業知識或是人生哲學，賴學長皆是我最佳的良師益友。也感謝學長在我畢業後不辭辛勞地為我尋求就業管道。此外，卉萱、鴻偉、阿德、建安、于萱、貴英以及 Kent 學弟，有這些同窗的陪伴，讓我知道原來念研究所也是可以很歡樂的，在交大這兩年因為有你們，使我的生活增色不少，謝謝你們。還有實驗室的其他學長們，小賢、Panda、f 龍、阿男、Jason、小班、冠傑、Dison，感謝你們在學業上的指導和生活上的照顧。

另外特別要感謝中山大學 ESLab 的夥伴們。靜昆、俊承、聖全、政隆、子銘、宗正、期財、俊宏、良弼、建宏、文綺、易廷、敬淵。和你們在中山一起渡過的日子，是我最難忘的回憶，感謝你們的照顧，並且在我低潮的時候給予我鼓勵，讓我在求學的旅途中找到新的方向。謝謝所有曾經幫助過我的人，讓我的碩士學位能順利完成。

蘇楷書 謹誌

民國九十九年八月十九日

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1. Thesis Motivation

In modern communication systems, the phase-locked loop (PLL) is widely used in synchronization applications, such as clock generator, frequency synthesizer, and clock recovery circuit. There are several categories of PLLs such as analog PLL (APLL), digital PLL (DPLL), All-digital PLL (ADPLL) [1]. Because of the process migration, portability, and design cycle issues, ADPLL is suitable for SoC implementation. However, it still spends much time to redesign a circuit with standard-cell-based IC design flow when the ADPLL control strategy is changed. Therefore, a novel type of ADPLL with silicon IPs which is flexible and reusable called software-defined phase-locked loop (SDPLL) platform is proposed. [2][3]

The SDPLL platform in [2][3] is composed of OpenRISC CPU and ADPLL related IPs, all of these components are integrated by WISHBONE bus. Since the CPU has powerful computing capability, the control strategy of ADPLL which called SDPLL algorithm that can be implemented with high level language such as C and C++. It is easier and faster than modify hardware circuit. The SDPLL algorithm defines the phase tracking and frequency search mechanism which are performed by CPU, the lock time of SDPLL is depends on the computing power of CPU. In OpenRISC-based SDPLL platform, it spends a long time to reach frequency maintain stage. It is not efficiency for stable clock generation. Thus, a more powerful architecture based on AndesCore CPU and AMBA bus system called ARM-based SDPLL platform is proposed in this thesis.

1.2. Thesis Contribution

The proposed ARM-based SDPLL platform can provide wide bandwidth and high resolution clock frequency. When the specification of system is changed, the designer can just modifies the software rather than redesign hardware to meet system specification. It can reduce design costs.

1.3. Thesis Organization

The organization of this thesis is as following. Chapter 1 introduces the phase-locked loop and the idea of flexibility of a system. Chapter 2 illustrates the basic concept of SDPLL. Chapter 3 shows the detail of the proposed SDPLL platform. Chapter 4 presents the implementation and simulation result. Chapter 5 is conclusion and future work.

Chapter 2
Overview of SDPLL

2.1. Introduction to ADPLL

There are three type of PLL. For hardware implement issue, all-digital approach is suitable and easier in this work. So ADPLL is chosen as basic PLL IPs. The function of the ADPLL is to generate frequency-locked and phase-locked clock. In order to reach the objective, there are several component included in the conventional ADPLL, such as phase frequency detector (PFD), time-to-digital converter (TDC), frequency divider, and digital controlled oscillator (DCO). PFD detects frequency or phase error between reference clock and divided-by-N clock. TDC converter error pulse received from PFD into digital data. If TDC output and DCO control tuning word is equal then digital date sends to DCO directly. But in practical, digital data output from TDC often need digital processing to calculate CTW. DCO generates clock signal depends on received CTW. Frequency divider divides the clock signal by user-defined number.

The basic ADPLL block diagram is shown in Fig. 2.1. The ADPLL working flow is as the following. First, PFD compares the incoming reference clock and feedback clock, and generates error pulse when frequency or phase error detected. Second, TDC receives the error pulse from PFD and converts it to digital values and output the values to the digital processing controller. Third, digital processing controller transforms the error value to DCO CTW and output the CTW to DCO. Fourth, DCO generates proper clock frequency with CTW. Fifth, Frequency divider divides the DCO output clock and feedbacks the divided clock to PFD. Repeat 1~5 until the reference and feedback clock are phase and frequency matched and DCO can output frequency-locked and phase-locked clock.

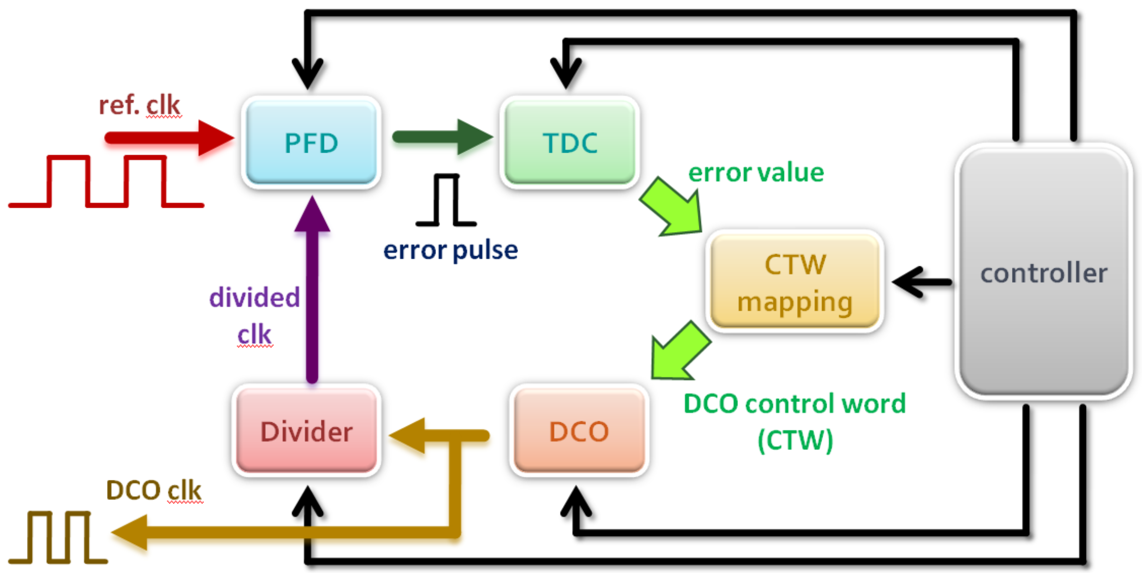


Fig. 2.1 Basic block diagram of ADPLL

However, it is not flexible for change transformation of CTW, tracking algorithm or control strategy in this conventional ADPLL architecture. For example, if the specification of the communication system is changed, the ADPLL control strategy must be modify for different application. Because of the hardware implementation issues, it means the designer must spend time to do RTL-simulation, logic synthesis, gate-level simulation, layout and verification to redesign circuit. It is time-consuming and hard work. Therefore, a SoC platform with software-controlled feature called SDPLL platform is proposed in this thesis.

2.2. Basic concept of SDPLL

The proposed SDPLL architecture has feature of software controllability and programmability by integrating CPU and ADPLL related IPs, it is flexible to the hardware architecture and the software operation. The basic SDPLL block diagram is shown in Fig. 2.2. The digital processing and controller part in Fig. 2.1 are replaced by CPU. It is more flexible and powerful than original processing. 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 AndesCore which will be described in section 2.2.1. At the software level, SDPLL only needs to modify the instructions which will be described in section 2.2.2, so as to supply different functions.

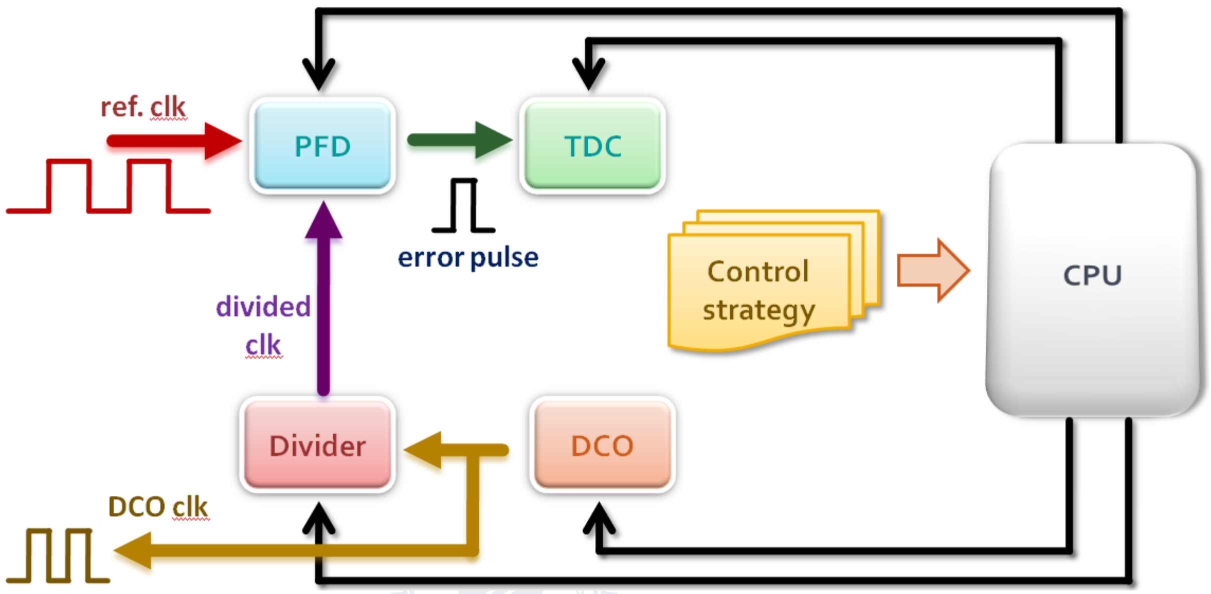


Fig. 2.2 SDPLL architecture

2.2.1. CPU

The selection of CPU is AndesCore N903-S provided by Andes Technology Corporation. The N903-S is a high performance, general purpose 32-bit RISC embedded processor designed for SoC applications. It is suitable for cost and power sensitive application which requires small footprint and manageable power consumption. The core is designed to allow customers to rapidly integrate their own IPs with a high performance RISC processor.

2.2.2. Software

The proposed SDPLL controls IP cores with software. In common, software can be developed with almost high level languages if there are corresponding compilers. In this work choose C/C++ as development language because C/C++ is one of the most popular programming languages. It is widely used on many different software platforms include AndesCore N-series CPU. Toolchain is part of AndeSight™, which is an integrated development environment for software development. Toolchain is mainly for compiling, assembling, and linking users' C/C++ and assembly programs and generating executable image. In Fig. 2.3 exhibit software flow of MIMO-SDPLL, it is common and used for a long while.

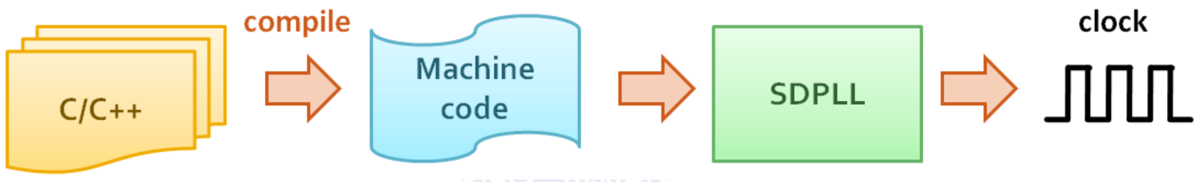


Fig. 2.3 Software flow of SDPLL

Chapter 3
ARM- based SDPLL Platform

3.1. System Overview

The proposed SDPLL architecture is shown in Fig. 3.1. CPU, PFD, TDC, frequency divider, DCO and other IP cores are connected by system bus. The memory stores the program of tracking algorithm and controlling strategy. Since the frequency of reference clock is too slow to be system clock, the semi-asynchronous clock generator (SACA) can provide faster system clock of SDPLL. The SACA output clock synchronous to the rising edge of reference clock and maintains low after the output clock count reaches the user-defined number. This can provide fast enough clock cycle for CPU computing. In Fig. 3.1, PFD, TDC and frequency divider generate digital data to digital processing. These three components are combined as error detector IP core. Note that these new IP cores need bus interface in order to connect to system bus. Since AMBA is selected as system bus architecture in this work, this SoC platform is called ARM-based SDPLL platform. The ARM-based SDPLL platform working flow is as the following. First, after system reset, CPU executes instructions from memory and initials all IP cores. Second, CPU start polling error detector until error detector detects error. Third, if the error detector detects error, it will convert the error pulse between reference clock and divided clock to error value and send the value to CPU. Fourth, When CPU receives the error value, it will calculate corresponding CTW and sends it to DCO. Fifth, DCO generates proper clock signal depends on received CTW and sends the clock signal to frequency divider. Sixth, Frequency divider divides the clock signal by user-defined number and feedbacks the divided clock to error detector. Repeat step 2~6 until reference and feedback clock are phase and frequency matched. The details of these IP cores will be illustrated in next section.

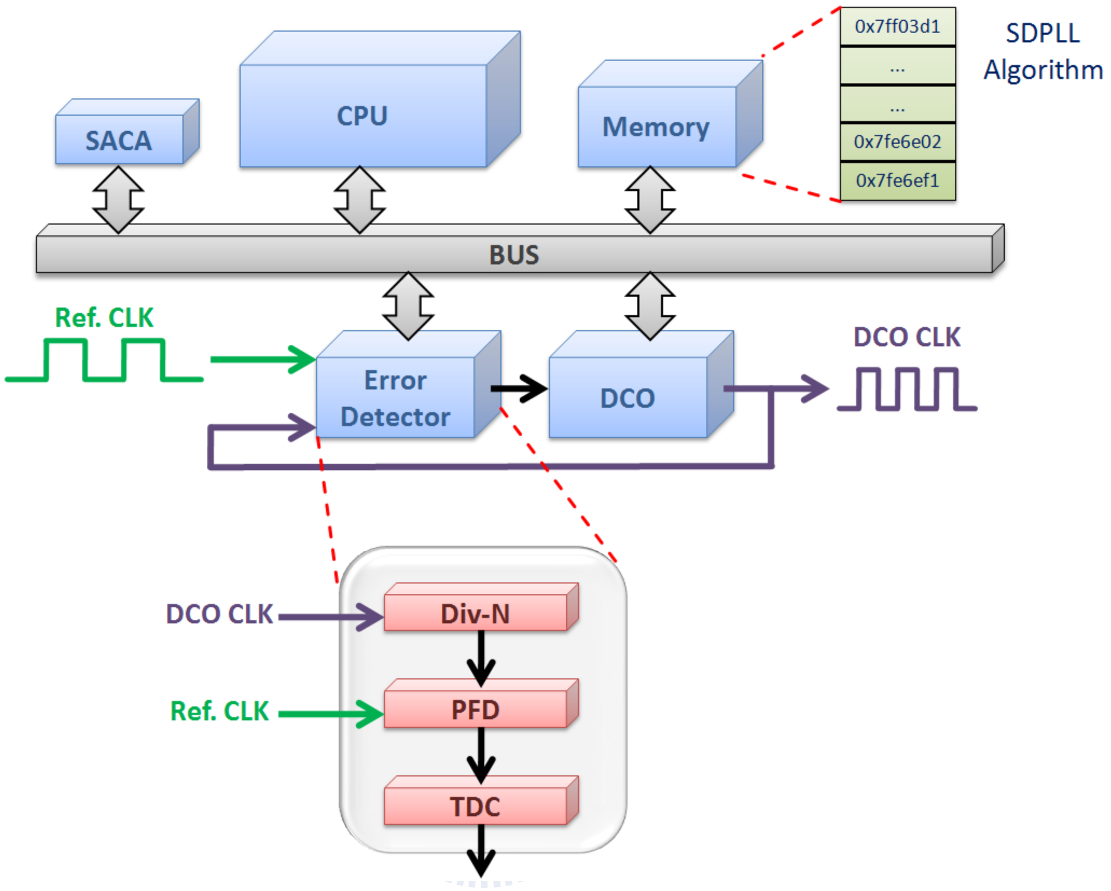


Fig. 3.1 ARM-based SDPLL platform architecture

3.2. System IP Core

3.2.1 N903-S

The selection of CPU in this work is AndesCore N903-S provided by Andes Technology Corporation [4] [5] [6]. N903-S which is a 32-bit RISC-based CPU with 5-stage pipeline provides AMBA AHB, AHB-Lite and APB interface for system bus. The N903-S block diagram is shown in Fig. 3.2. External Bus Interface is responsible for off-CPU memory access which includes system memory access and memory-mapped register access in devices. The N903-S supports AHB, AHB-lite, APB and AMI bus protocols. There is no constraint on the bus clock ratio between CPU core and AMBA bus clock. The local memory is to store those data and instructions that might be accessed frequently in a system such as service routine, system call, application data, etc. N903-S's local memory only can be configured as external since it target on a cost sensitive system. It provides external local memory interface to allow N903-S communicate with external memory. N903-S supports both instruction local memory and data local memory. The embedded debug module allows programmers perform debugging activities through a standard JTAG interface. It provides hardware breakpoint functionality, a dedicated interface to the target system bus, and the Debug Instruction Memory (DIM) to help the user debug the target software on the target hardware system.

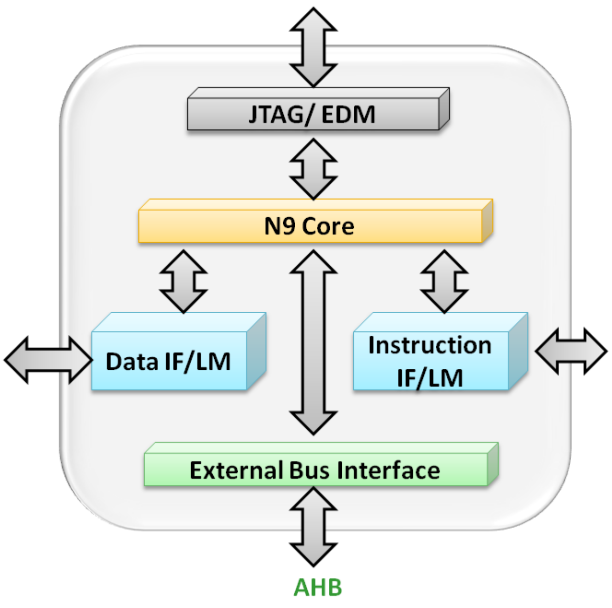


Fig. 3.2 Block diagram of N903-S

3.2.2 AMBA Bus

The AMBA specification defines an on-chip communications standard for the infrastructure of high-performance embedded systems [7]. There are three buses defined within the AMBA specification, one is the advanced high performance bus (AHB), another is the advanced system bus (ASB), and the other is the advanced peripheral bus (APB). Since the AHB provides a higher performance and is popular bus architecture for SoC design, it is used in this work. The typical AMBA AHB-based system is shown in Fig. 3.3. AHB supports the efficient connection of processor, on-chip memory and other IPs which has the requirement of high bandwidth transfer.

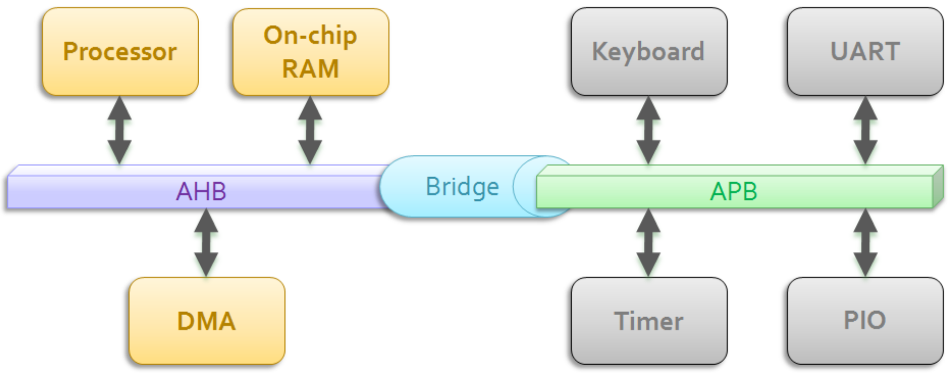


Fig. 3.3 AHB-based system

3.2.3 Error Detector

The error detector consists of three parts, the first part is PFD, the second par is TDC, and the third part is frequency divider. Fig. 3.4 shows the block diagram of error detector. TDC can not only measure the clock period of reference clock but also the phase error between reference clock and divided clock depending on detect mode. The basic concept of TDC is counting the pulse which is generated by the internal delay chain. Frequency divider divides the frequency of DCO clock. When the rising edge of DCO clock comes, the internal counter in frequency divider will increase one until the counter value equals the user-defined divided value. The phase frequency detector converts the timing difference between the rising edge of reference clock and divided clock to the pulse and to determine which clock signal is lead to another.

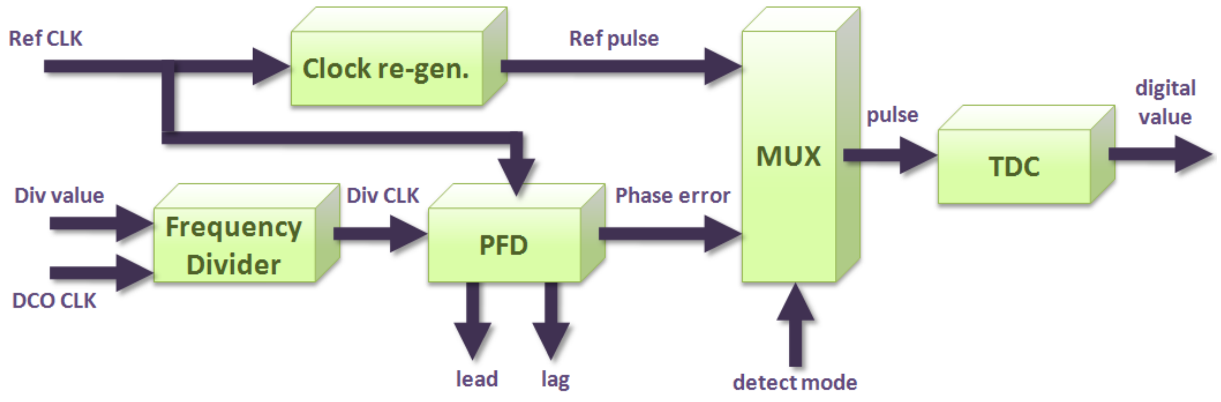


Fig. 3.4 Error detector block diagram

3.2.4 DCO

The DCO is high-resolution and wide frequency range is proposed at [8]. It can generate clock frequency by the control of digital signal. The range of frequency is from 0.66MHz to 460MHz.

3.2.5 SACA

Since there are several IPs driven by system clock and the reference clock is too slow in this work, the SACA clock generator module is proposed. SACA is a clock generator which synchronous to the rising edge of reference clock. And start trigger fixed number of cycles with specific period asynchronous to reference clock. The fixed number of cycles and clock period are defined by user. Fig. 3.5 shows the example of SACA with eight cycle count. The output frequency range of SACA is from 103MHz to 1231MHz.

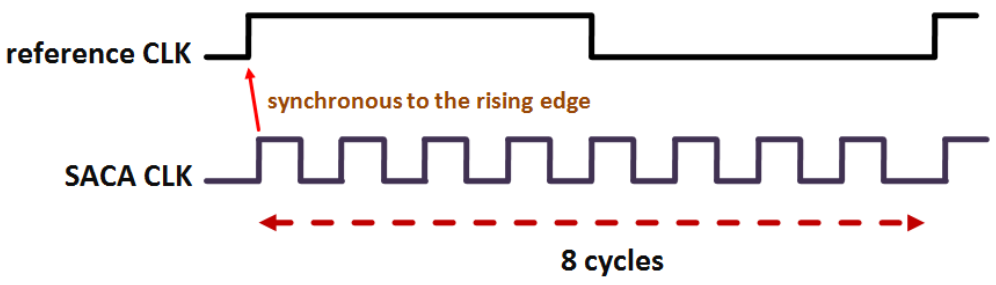


Fig. 3.5 The example of SACA

3.3. Tracking Algorithm

The SDPLL tracking algorithm is composed of three parts: frequency search, coarse tracking and fine tracking. In frequency search stage, the period of reference clock will be estimated by TDC and transfer into CTW to set DCO output clock frequency. After frequency search, the tracking algorithm does coarse tracking to fixing frequency and phase error until the phase error between reference clock and divided clock is small than TDC’s minimum detectable range, and the track algorithm enter the fine tracking stage. Since DCO has high resolution feature, more accurate tracking can do by DCO and PFD in fine tracking stage.

Chapter 4
Implementation and Simulation Result

In this section, the implementation of ARM-based SDPLL platform will be discussed in two parts, system integration and software programming. In system integration section, the implementation details of ARM-based SDPLL architecture are presented. In software programming section, the method of hardware control via software and software programming are presented. The hardware and software co-simulation result is showed at the end of this section.

4.1. System Integration

The system architecture of ARM-based SDPLL platform is shown in Fig. 4.1. All system IP cores such as CPU, SACA, memory, error detector, DCO are connected by AMBA AHB bus. The AHB interface is used for transformation of the signals between AHB bus and these IP cores. CPU executes instructions in memory which are compiled from C source code to controls other IP cores such as error detector and DCO. The SDPLL tracking algorithm is implemented by C source code and loaded to memory before system reset signal asserted. After system reset, CPU starts polling error detectors alternative. When reference clock and divided clock have phase error, the error detector will raise error flag signal. CPU then does tracking algorithm for the correspond device.

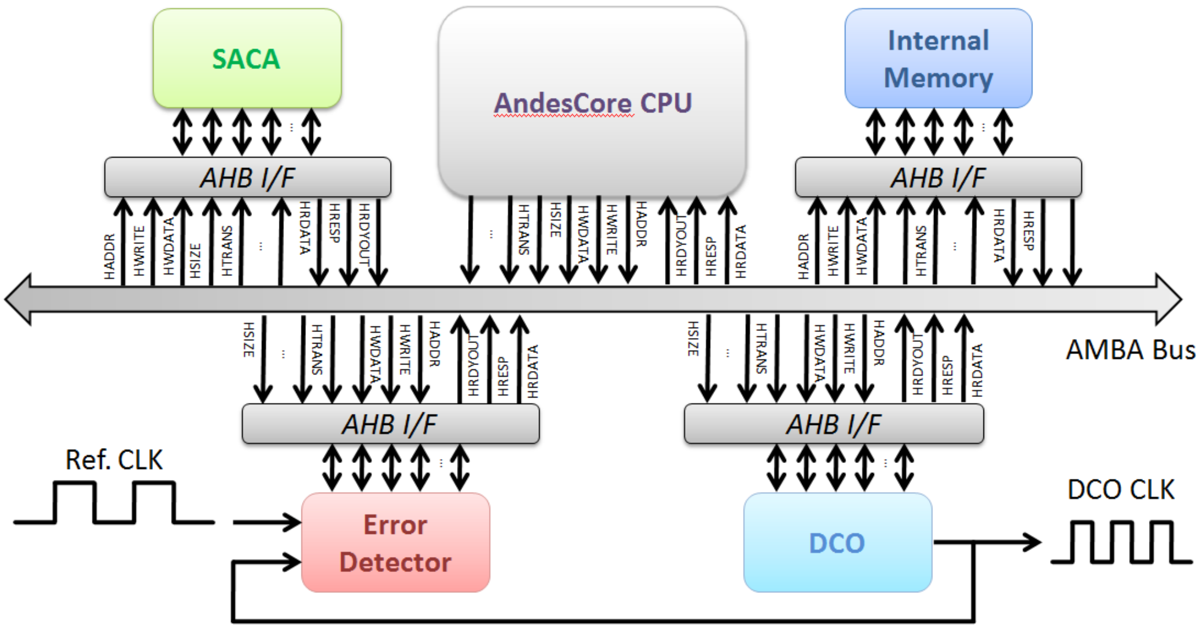


Fig. 4.1 ARM-based SDPLL architecture

4.1.1. Memory Mapping

In ARM-based SDPLL platform, the CPU needs to communicate with other IP cores. The popular way to do this job is memory-mapped I/O. With this method, each IP core is considered as an I/O peripheral and occupies specific address in the existing address space. CPU can access these IP cores by sending the specific address to memory location or registers in these IP cores. Since the software programmer can use pointer and data structure to communicate with hardware and reduce the complexity of hardware implementation, this method is helpful for hardware and software co-design.

4.1.2. N903-S Model

The selection of CPU is AndesCore N903-S. For HDL IP protection issue, it uses a behavior model for RTL simulation called AMP model. The N903-S AMP model is a binary model of released firmcore produced by Cadence IP Model Packager. AMP model communicates with simulators through the IEEE Std 1499 Open Model Interface (OMI) protocol or PLI through the OMI Adaptor. In this work we use OMI protocol for CPU model.

4.1.3. AHB Bus Protocol

For CPU compatibility and IP cores connection, the AMBA AHB bus is chosen for system bus. A typical AMBA AHB system design contains the following components, AHB master, AHB slave, AHB arbiter, and AHB decoder, these components are connected by a central multiplexer.

The AHB master sends address and control signals to slave to perform read and write operations. Only one master is allowed to use the bus at any one time. The maximum number of master in AHB bus is 16. The AHB slave receives address and control signals from AHB master and responds to a read or write operation. The AHB arbiter judges which master can use the bus and ensures that only one bus master at a time is allowed to initiate data transfers. The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer.

ALL bus masters send the address and control signals to indicate the transfer they wish to perform and the arbiter determines which master has its address and control signals routed to all of the slaves. After slave receives the address and control signals sent from master, the decoder is also required to control the read data and response signal multiplexor to select the appropriate signals from the slave. The central multiplexor interconnection scheme is shown in Fig. 4.2.

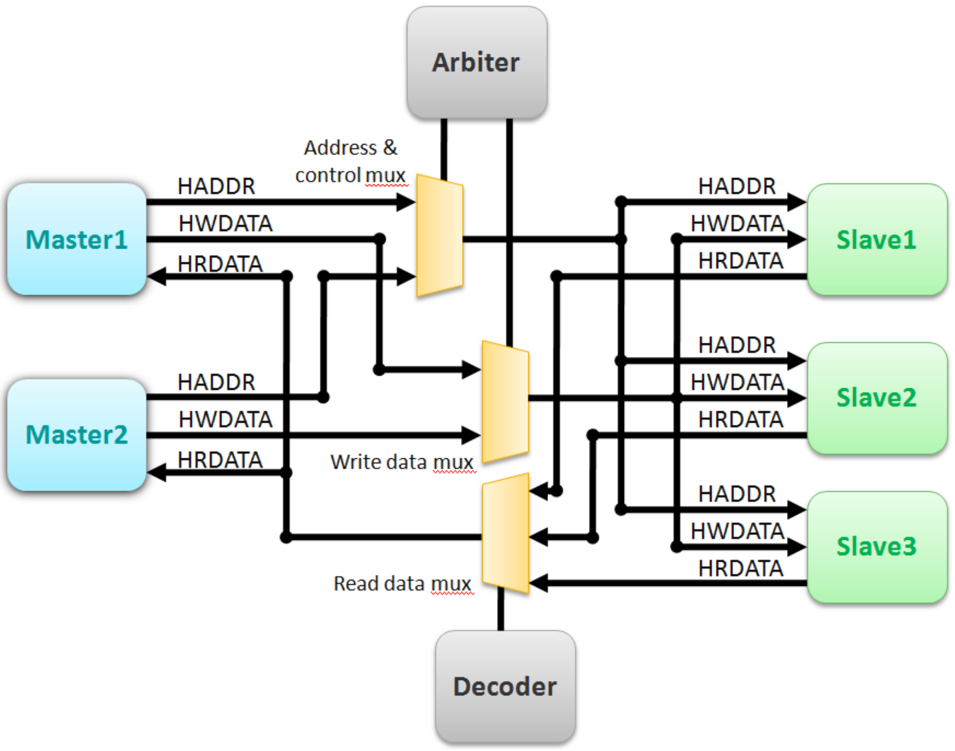


Fig. 4.2 AMBA central multiplexor interconnection

The AHB master must send HBUSREQ signal to AHB arbiter to grant the bus before performing a bus transfer. Then the arbiter will judge that which AHB master has higher priority to access the bus. Then the granted master sends the address and control signals to access the slaves. Each slave on the bus will receive the address signal and control signals but only the specific slave can be access and response to the master. The AHB simple transfer is shown in Fig 4.3. The AHB transfer consists of two parts, one is the address phase and the other is the data phase. The master send address and control signals to slave in address phase, and the slave response to master in data phase. Note that the data phase may be extended since the slave is not ready. The data phase can be extended by sending HREADY signal from slave.

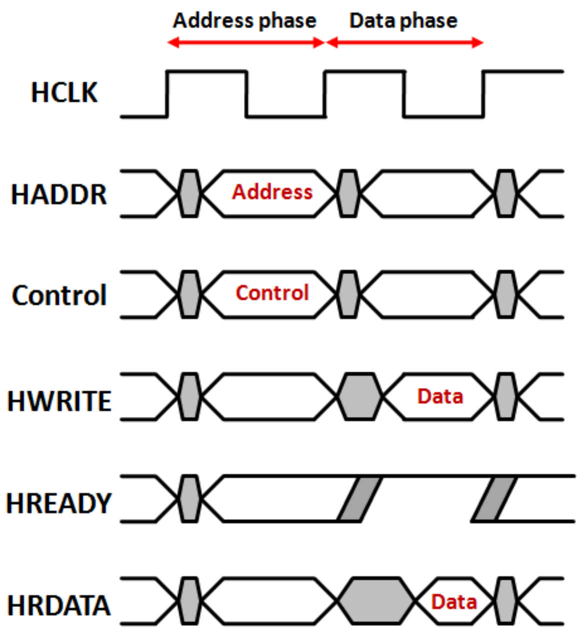


Fig. 4.3 AHB simple transfer

4.1.4. AHB Interface

To integrate DCO and error detector into the AMBA-based system, we need a bus interface to deal with the signal transformation between DCO/error detector and AHB bus. Fig. 4.4 shows the block diagram of bus interface. Depend on the proposed SDPLL algorithm, CPU sets the control signal to DCO and receives the error value from. The DCO write control part passes the DCO_CTW signal to DCO to generate proper frequency and determines the mode of DCO (frequency search stage / phase tracking stage). The error detector read control part passes the divide value to the frequency divider of error detector and the mode of error detector (frequency search stage / phase tracking stage). The error value which estimated by TDC and phase lead / lag signal also passed to bus through error detector read control. All of these read and write operations can be done in one cycle. Table 1 is the comparison of this work and OpenRISC-based bus interface. The increase of area is 13% since the complexity of AMBA protocol and the cycle of operation is equal to OpenRISC-based bus interface. By implementation of AHB interface, the DCO and error detector can be integrated into AMBA-based system easily.

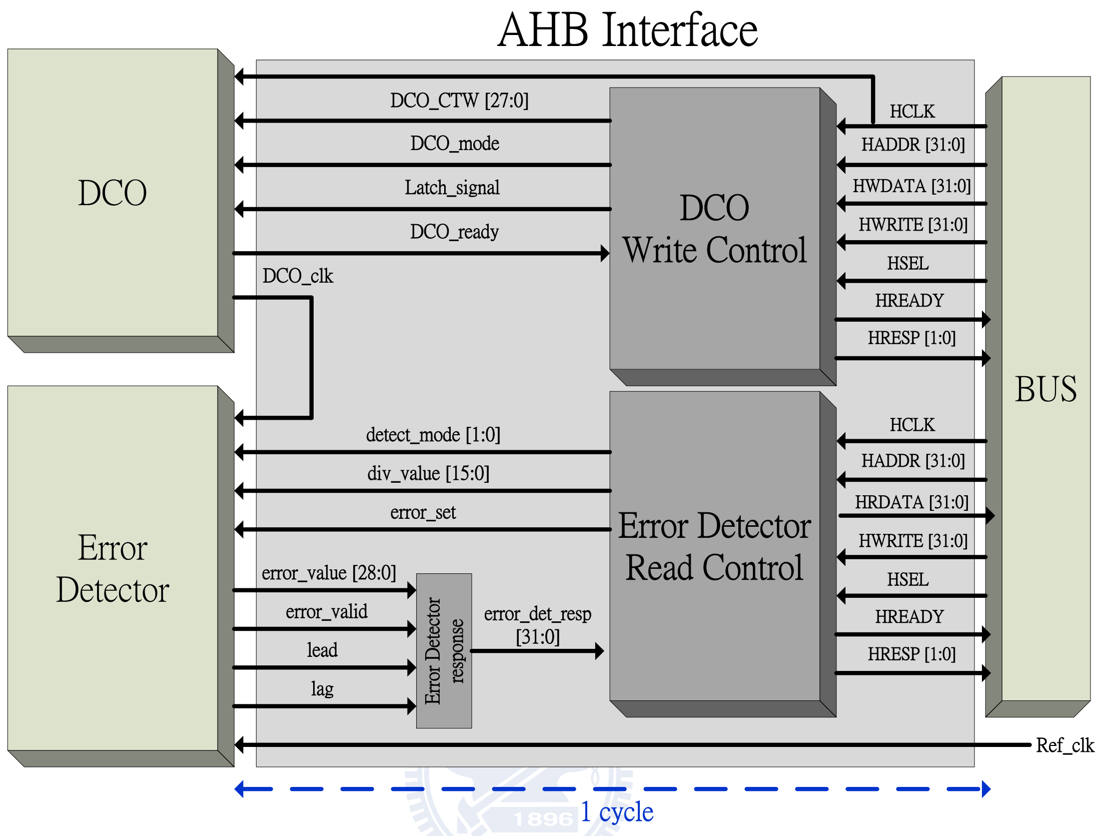


Fig. 4.4 Block diagram of AHB interface

	Set control word to DCO	Read from error detector	Bus protocol	Area*	Process
This work	1 cycle	1 cycle	AMBA2.0	2194um^2	90nm CMOS
OpenRISC-based	1 cycle	1 cycle	WISHBONE	1935um^2	90nm CMOS

Table 1 Comparison of the bus interface

4.1.5. AHB Interconnection

In this work, the bus interconnection is based on Example AMBA System (EASY) architecture. The EASY architecture provides a typical AMBA platform for SoC design. Since the EASY platform is integrated with ARM7 and ARM9 series CPU and several default IPs, it must be modified to meet the system requirement of SDPLL platform. Fig. 4.5 shows the bus interconnection of ARM-based SDPLL platform.

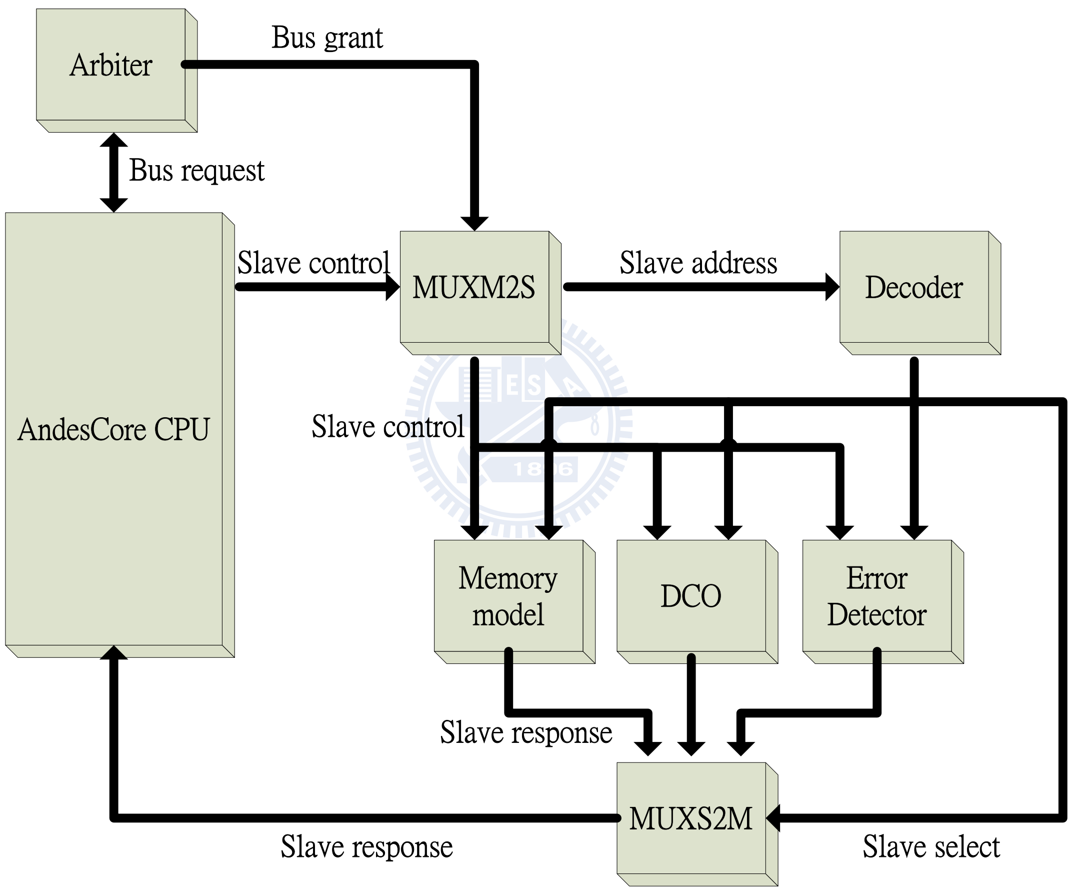
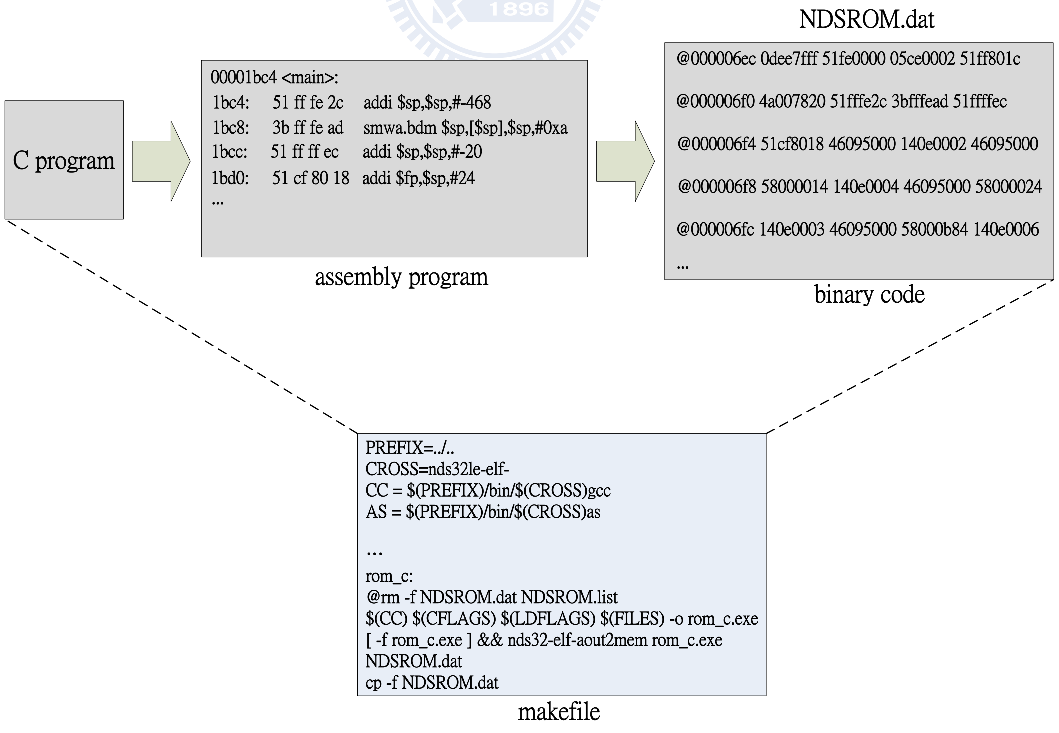


Fig. 4.5 Bus interconnection of ARM-based SDPLL platform

4.2. Software Programming

4.2.1. Software development flow and hardware simulation environment

For software development, software programmer needs toolchain to transform the source code into an executable program. Andes toolchain is built from GNU, thus the options of gcc, as, and ld are inherited [9] [10]. The cross-compiler compiles the C program, and the assembler and linker converts the assembly programs to a.out file. By default, N903-S starts fetching instructions from memory address 0x0, therefore the –Ttext=0 linker switch asks the linker to arrange the starting address of the final executable image to be at pc=0x0. For hardware / software co-simulation, it is necessary to build a development flow. The C program can be compiled to assembly program by nds32le-elf-gcc cross compiler. However, for hardware simulation the assembly program needs to be converted to a binary code which can be loaded into memory model. This conversion is performed through nds32-elf-aout2mem. Fig. 4.6 shows the conversion flow.



```
00001bc4 <main>:
 1bc4:     51 ff fe 2c    addi $sp,$sp,#-468
 1bc8:     3b ff fe ad    smwa.bdm $sp,[$sp],$sp,#0xa
 1bcc:     51 ff ff ec    addi $sp,$sp,#-20
 1bd0:     51 cf 80 18    addi $fp,$sp,#24
 ...
```

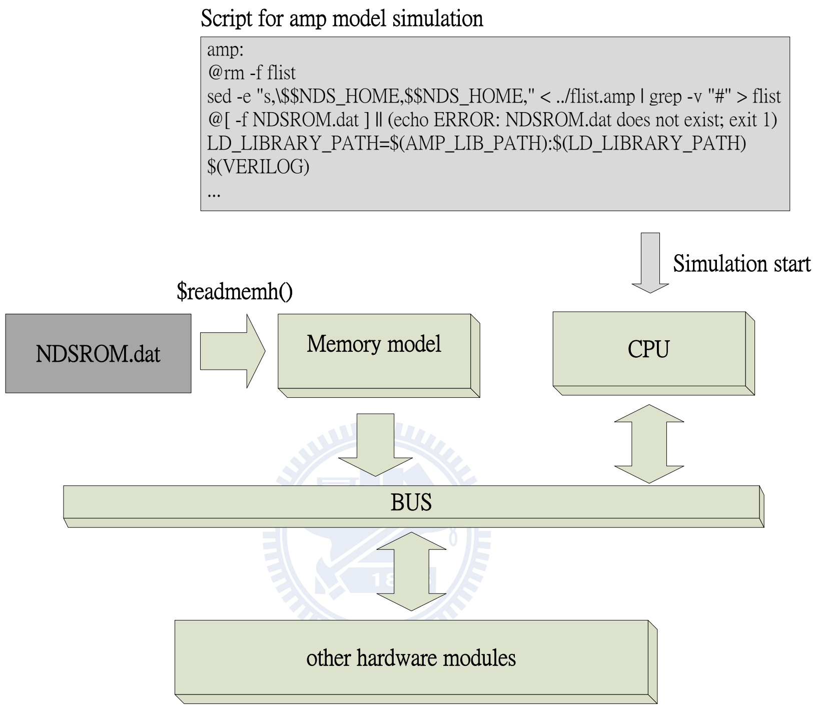
```
@000006ec 0dee7fff 51fe0000 05ce0002 51ff801c
@000006f0 4a007820 51fffe2c 3bfffead 51ffffec
@000006f4 51cf8018 46095000 140e0002 46095000
@000006f8 58000014 140e0004 46095000 58000024
@000006fc 140e0003 46095000 58000b84 140e0006
...
```

```
PREFIX=../..
CROSS=nds32le-elf-
CC = $(PREFIX)/bin/$(CROSS)gcc
AS = $(PREFIX)/bin/$(CROSS)as

…
rom_c:
@rm -f NDSROM.dat NDSROM.list
$(CC) $(CFLAGS) $(LDFLAGS) $(FILES) -o rom_c.exe
[ -f rom_c.exe ] && nds32-elf-aout2mem rom_c.exe
NDSROM.dat
cp -f NDSROM.dat
```

Fig. 4.6 C to binary code flow

The binary called NDSROM.dat can be loaded to memory model by Verilog $readmemh() system task. CPU fetches instruction from memory address 0x0 and jump to <c_star> routine and do SDPLL algorithm when simulation start. Fig. 4.7 shows the hardware simulation flow.



```
amp:
@rm -f flist
sed -e "s,\$$NDS_HOME,$$NDS_HOME," < ../flist.amp | grep -v "#" > flist
@[ -f NDSROM.dat ] || (echo ERROR: NDSROM.dat does not exist; exit 1)
LD_LIBRARY_PATH=$(AMP_LIB_PATH):$(LD_LIBRARY_PATH)
$(VERILOG)
...
```

Fig. 4.7 Hardware simulation flow

4.2.2. *Hardware Access*

Since SDPLL tracking algorithm need access error detector and DCO by setting value to control registers, the memory-mapped I/O mechanism is used in this work. The memory-mapped I/O control is as the following. First, define the device base address. Second, declare a pointer variable with the volatile keyword and assign base address value to the variable. Note that volatile qualifier must be used when reading the contents of a memory location whose value can change unknown to the current program. Third, this pointer can read or write IP cores register by software.

4.3. Simulation Result

In this section, the simulation result of SACA, error detector, DCO is presented as following. These IP are implemented with UMC 90nm standard cell library. Fig. 4.x is simulation waveform of SACA module. The user-defined divided value is 8. That means there are eight clock cycles which synchronous to the rising edge of reference clock will be generated in one reference clock period. The SACA can generate clock frequency with unbalanced duty cycle of the reference clock. Fig. 4.8 shows the SACA module works with 40% / 60% duty cycle of reference clock.

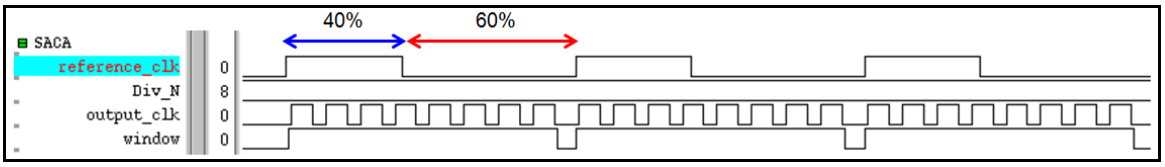


Fig. 4.8 Simulation of SACA with 10MHz reference clock and 83.87MHz output clock

The simulation result of error detector is shown in Fig. 4.9. After divided clock feedbacks to the error detector, the error detector generates a digital value for CTW mapping, and judges if the divided clock leads the reference clock. Part (a) of Fig. 4.9 shows that the TDC receives the phase error between reference clock and divided clock and outputs error value.

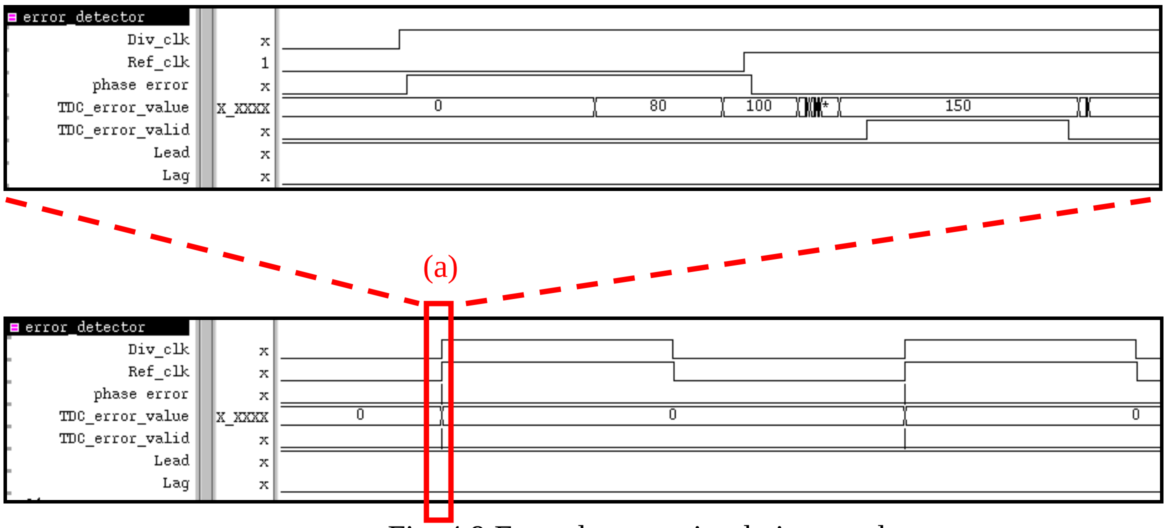


Fig. 4.9 Error detector simulation result

Fig. 4.10 shows the simulation result of DCO control. The DCO output clock period is changed by CTW which is sent from CPU. In part (b) of Fig. 4.10, HCLK, HADDR, HWDATA and HWRITE are come from AMBA AHB bus. The CPU sends CTW 0x8ff80000 to address 0x95000014, which means set CTW to DCO control register. Note that the address is point to DCO control register. After CPU send these control signals, the DCO control bit C1 will be set to 0x000001ff and the DCO output clock period will be changed.

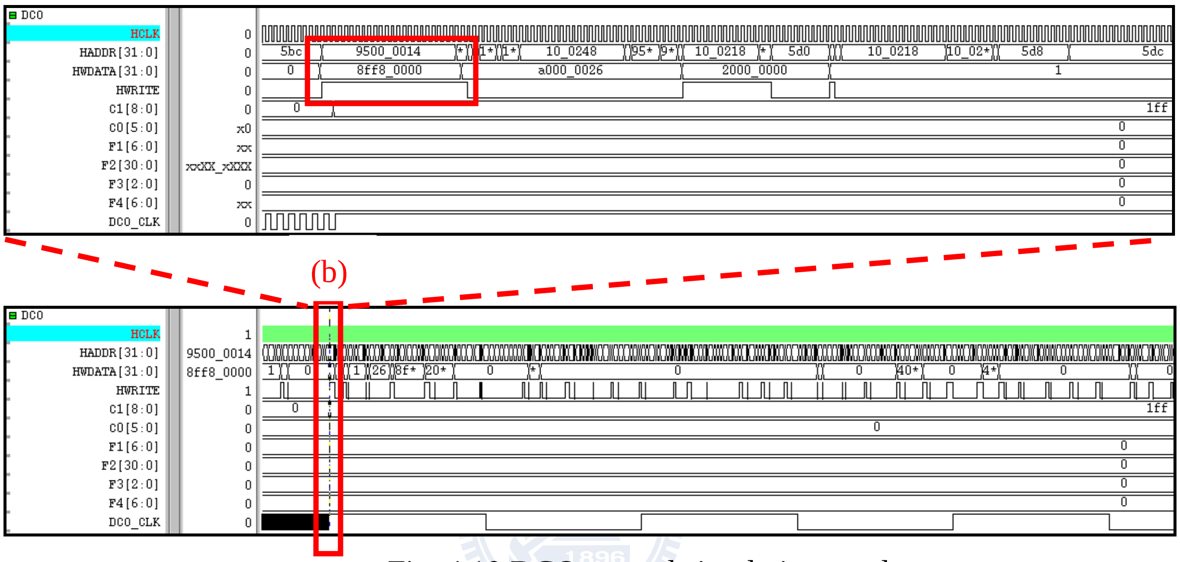


Fig. 4.10 DCO control simulation result

Chapter 5
Conclusion and Future Work

We have introduced basic SDPLL concept and the proposed ARM-based SDPLL platform in this thesis. The proposed SDPLL architecture has feature of software controllability and programmability by integrating CPU and silicon IPs. It is flexible for the hardware architecture and the software operation.

The following topics to extend the work can be proposed. The CPU and memory is behavior model. It must be implemented with gate-level logic to do system verification with precise timing information. And then we can do silicon implementation to verify this SoC system with real chip design.

Bibliography

[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] Chang-Ying Chuang, Terng-Yin Hsu" The study of Software-defined Phase-locked loop" Thesis CS, NCTU 2008

[3] Ze-Bin Huang, Terng-Yin Hsu" The study of MIMO Softwaredefined Phase-locked Loop" Thesis CS, NCTU 2009

[4] "*AndesCore N903-S Integration Guide*" IG0005-10, Oct. 2008

[5] "*AndesCore N903-S Verification Guide*" VG0005-10, Aug. 2008

[6] "*AndesCore N903-S Data Sheet*" DS0005-10, Nov. 2008

[7] *"AMBA Specification"* ARM IHI0011A, May 1999

[8] Jung-Chin Lai, Terng-Yin Hsu" The study of Wideband, Cell-based Digital Controlled Oscillator and its Implementation" Thesis CS, NCTU 2007.

[9] "*Andes Programming Guide*" PR002, Jun. 2009

[10] "*AndeSight User Manual*" UM017, Jun. 2009