應用於系統晶片設計之
自動化合成時序鎖定迴路
Automatic Synthesis of Timing-Locked Loops for SoC Designs

研 究 生 ：鍾 菁 哲

指導教授 ：李 鎮 宜

中華民國 九十二 年 十 月

應用於系統晶片設計之自動化合成時序鎖定迴路
Automatic Synthesis of Timing-Locked Loops for SoC Designs

研 究 生：鍾 菁 哲　　　Student ：Ching-Che Chung

指導教授：李 鎮 宜 博士　　　Advisor ：Dr. Chen-Yi Lee

國 立 交 通 大 學

電子工程學系電子研究所

博 士 論 文

A Dissertation
Submitted to Institute of Electronics
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Electronics Engineering
October 2003
Hsinchu, Taiwan, Republic of China.

中華民國 九十二 年 十 月

推 薦 函

主旨：推薦電子工程學系博士班研究生鍾菁哲，舉行博士班學位口試。

說明：本人所指導之博士班學生鍾菁哲，業已通過資格考試，並完成本校電子工程學系電子研究所博士班規定之學科課程及論文研究訓練。鍾君主要從事全數位鎖相迴路(ADPLL)及全數位延遲鎖相迴路(ADDLL)之研究工作，其論文題目「應用於系統晶片設計之自動化合成時序鎖定迴路」（Automatic Synthesis of Timing-Locked Loops for SoC Designs）針對 ADPLL/ADDLL 在不同應用或規格上的需求，從晶片性能、面積及功率消耗三方面著手，提出一系列的系統化設計方法和晶片實現方式，並透過矽晶片驗證所改善效能。相關研究成果如下：

[1] Jing-Jer Jong and Chen-Yi Lee, "A Novel Structure for Portable Digitally Controlled Oscillator, " in *IEEE International Symposium on Circuits and Systems*, Vol. 1, pp. 272-275, May 2001.

[2] Ching-Che Chung and Chen-Yi Lee, "An all-digital phase-locked loop for high-speed clock generation," in *IEEE International Symposium on Circuits and Systems*, Vol. 3, pp. 26-29, May 2002.

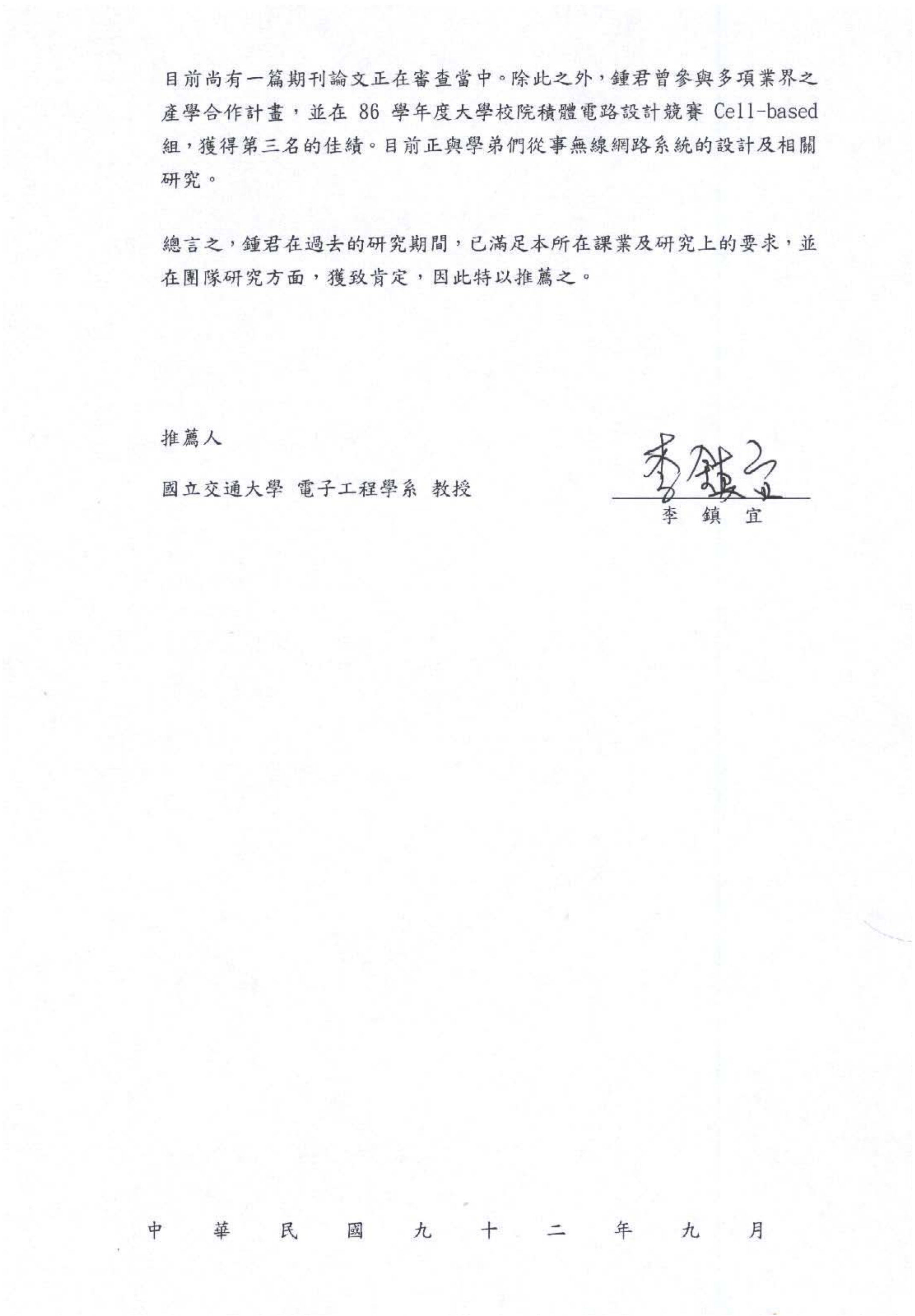
[3] Hsie-Chia Chang, Ching-Che Chung, Chien-Ching Lin, and Chen-Yi Lee, "A 300 MHz Reed-Solomon decoder chip using inverionless decomposed architecture for Euclidean algorithm, " in *28th European Solid-State Circuits Conf. (ESSCIRC)*, Florence, Italy, Sep. 2002.

[4] Ching-Che Chung and Chen-Yi Lee, "An all-digital phase-locked loop for high-speed clock generation," in *IEEE Journal of Solid-State Circuits*, Vol. 38, pp.347-351, Feb. 2003.

[5] Ching-Che Chung and Chen-Yi Lee, "A new DLL-based approach for all-digital multi-phase clock generation, " accepted by *IEEE Journal of Solid-State Circuits*.

投出待審之論文：

[6] Ching-Che Chung and Chen-Yi Lee, "An All-Digital Fast-Locking DLL for Wide-Range Clock Deskew Applications," submitted to *IEICE Transactions on Electronics*.



目前尚有一篇期刊論文正在審查當中。除此之外，鍾君曾參與多項業界之產學合作計畫，並在 86 學年度大學校院積體電路設計競賽 Cell-based 組，獲得第三名的佳績。目前正與學弟們從事無線網路系統的設計及相關研究。

總言之，鍾君在過去的研究期間，已滿足本所在課業及研究上的要求，並在團隊研究方面，獲致肯定，因此特以推薦之。

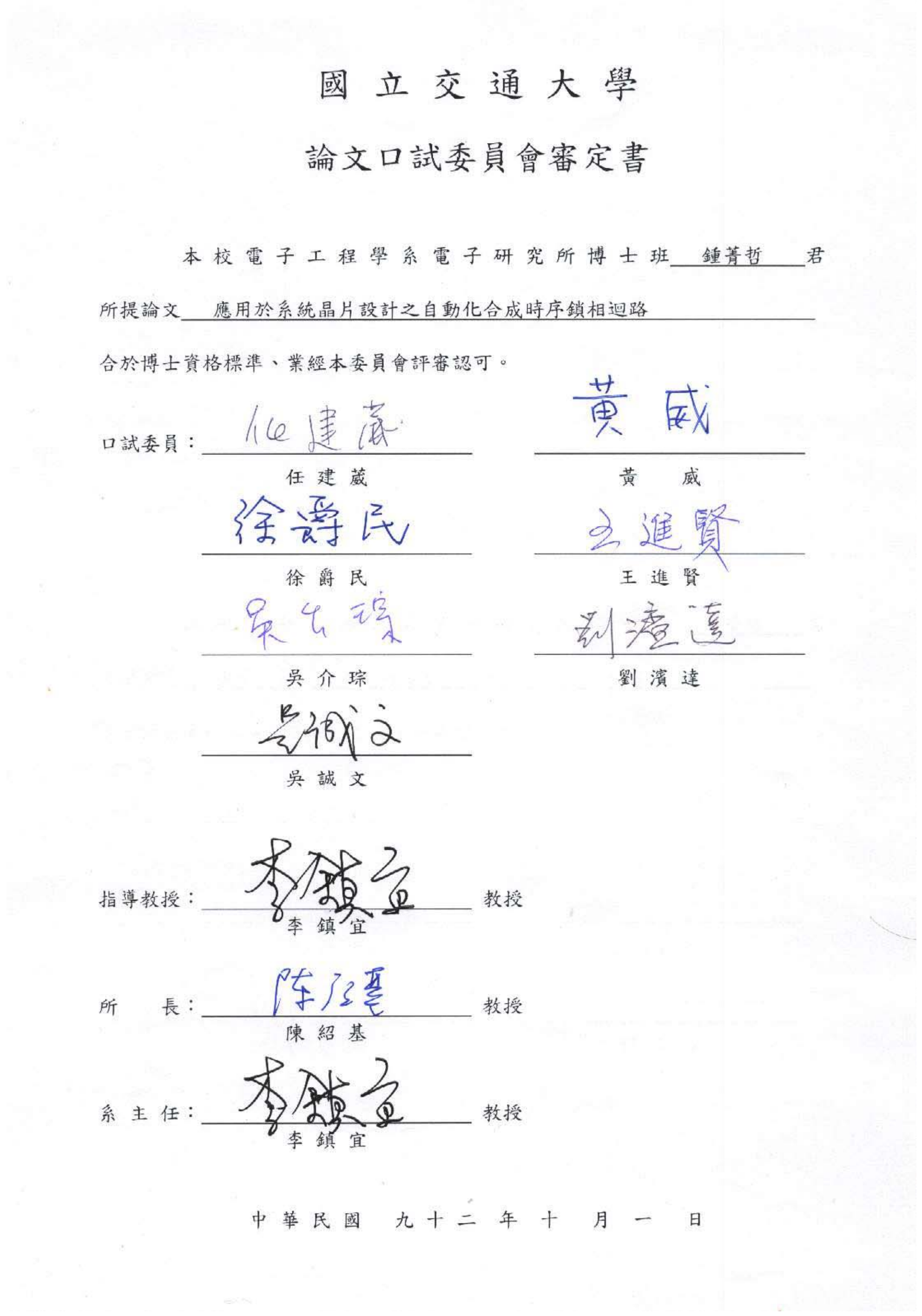
推薦人

國立交通大學 電子工程學系 教授

[illegible]

李 鎮 宜

中 華 民 國 九 十 二 年 九 月



國 立 交 通 大 學

論文口試委員會審定書

本校電子工程學系電子研究所博士班 鍾菁哲 君

所提論文 應用於系統晶片設計之自動化合成時序鎖相迴路

合於博士資格標準、業經本委員會評審認可。

口試委員：[illegible]	黃 威
任 建 葳	黃 威
徐爵民	[illegible]
徐 爵 民	王 進 賢
[illegible]	[illegible]
吳 介 琮	劉 濱 達
[illegible]	
吳 誠 文	

指導教授：[illegible] 教授
李 鎮 宜

所　長：[illegible] 教授
陳 紹 基

系 主 任：[illegible] 教授
李 鎮 宜

中 華 民 國　九 十 二 年 十 月 一 日

Department of Electronics Engineering
& Institute of Electronics
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.

Date : 2003-10-01

We have carefully read the dissertation entitled Automatic Synthesis of Timing-Locked Loops for SoC Designs

submitted by Ching-Che Chung in partial fulfillment of the requirements of

the degree of DOCTOR OF PHILOSOPHY and recommend its acceptance.

Chein-Wei Jen	Wei Hwang
Chein-Wei Jen	Wei Hwang
[illegible]	[illegible]
Jyuo-Min Shyu	Jin-Shyan Wang
[illegible]	[illegible]
Jieh-Tsorng Wu	Bin-Da Liu
[illegible]	
Cheng-Wen Wu	

Thesis Advisor : [illegible]
Chen-Yi Lee

Director : [illegible]
Sau-Gee Chen

Chairman : [illegible]
Chen-Yi Lee

博碩士論文電子檔案上網授權書

（提供授權人裝釘於紙本論文書名頁之次頁用）

本授權書所授權之論文為授權人在國立交通大學電子工程系 92 學年度第一學期取得博士學位之論文。

論文題目：應用於系統晶片設計之自動化合成時序鎖定迴路

指導教授：李鎮宜

授權人：鍾菁哲

簽名：鍾菁哲　中華民國 92 年 10 月 3 日

用於系統晶片設計之自動化合成時序鎖定迴路

研究生 ： 鍾菁哲　　　　　　　　　　指導教授 ： 李鎮宜博士

國立交通大學電子工程學系電子研究所

摘要

在此論文中，我們針對全數位鎖相迴路及全數位延遲鎖相迴路的設計，提出了有彈性的架構，以便可迅速配合不同應用領域進行修改。本論文所提出的架構可有效的縮減迴路鎖定所需的時間，並且在配合所提出的自動化合成技術下，使用數位積體電路技術和標準元件庫，可在短時間內快速的合成使用者所需求的全數位鎖相迴路或是全數位延遲鎖相迴路。因此本論文所提出的架構和自動化設計方法，非常適合應用於系統晶片使用。

此論文首先提出如何克服標準元件庫的限制來提高延遲元件的解析度和相位頻率比較器的靈敏度。接著本論文所提出的微調延遲元件和相位比較器便被應用到全數位鎖相迴路及全數位延遲鎖相迴路的設計。時間數位量測轉換器被大量的應用在所提出的全數位鎖相迴路及全數位延遲鎖相迴路的設計裡，以減少頻率鎖定和相位鎖定所需的時間。接著本論文提出以延遲鎖相迴路架構為基礎的全數位多相位時脈產生器，在此多相位時脈產生器的設計上，時間數位量測轉換器用來選定適當的延遲範圍，以避免產生傳統電路設計會鎖定到諧波的困擾。

最後本論文提出了合成全數位鎖相迴路、全數位延遲鎖相迴路及全數位多相位時脈產生器的方法。因而全數位鎖相迴路、全數位延遲鎖相迴路及全數位多相位時脈產生器的邏輯閘層次設計電路，將可以使用標準元件庫來自動產生。也因此針對全數位鎖相迴路、全數位延遲鎖相迴路及全數位多相位時脈產生器所需的設計時間和設計複雜度都可以因此大幅度的降低。

Automatic Synthesis of Timing-Locked Loops for SoC Designs

Student : Ching-Che Chung　　　　　　Advisor : Dr. Chen-Yi Lee

Department of Electronics Engineering
& Institute of Electronics
National Chiao Tung University

Abstract

In this dissertation, the designs for All-Digital Phase-Locked Loop (ADPLL) and All-Digital Delay-Locked Loop (ADDLL) are presented. The proposed flexible ADPLL/ADDLL architectures can easily be modified to fit different applications and achieve fast lock-in time. The proposed automated synthesis methodology uses both benefits of digital VSLI and cell-based design to build up user-specified ADPLL/ADDLL in a short time, making it very suitable for System-on-Chip (SoC) applications.

This dissertation first presents a scheme to overcome the limitations of standard cells and to build up high resolution delay cell and high sensitivity Phase and Frequency detector (PFD). Then the proposed fine-tuning delay cell and PFD are applied to ADPLL/ADDLL. Time-to-Digital Converter (TDC) is widely used in the proposed ADPLL/ADDLL architecture to reduce frequency/phase acquisition time. Then design for the proposed DLL-based All-Digital Multi-phase Clock Generator (ADMCG) is presented. In ADMCG design, TDC is used to choose a suitable delay range to avoid false-lock to harmonics.

Finally, a synthesis approach for ADPLL/ADDLL/ADMCG design is presented. As a result, the gate-level Hardware Description Language (HDL) codes of ADPLL/ADDLL/ADMCG can be automatically generated using standard cells from cell-library. Hence both design time and design complexity of ADPLL/ ADDLL/ADMCG is greatly reduced by the proposed methodology.

Acknowledgments

I would like to express my deepest gratitude to my advisor Prof. Chen-Yi Lee for his sophomore enthusiastic guidance and encouragement throughout the research, and wholeheartedly give him and his family my best wishes.

During my research, I would like to thank National Science Council (NSC) for supporting the project of My Ph. D work. The chip support from Chip Implementation Center (CIC) of NSC is acknowledged, too.

I want to thank my senior Si2 group mate, Dr. Terng-Yin Hsu and Mr. Terng-Ren Hsu for many valuable discussions and great help. Besides, I much appreciate my junior Mr. Hsuan-Yu Liu and Mr. Chien-Ching Lin for their comments and CAD supporting. I also want to thank all members of the Si2 group of NCTU for plenty of fruitful assistance in my graduated lives.

Finally, I give the greatest respect and love to my family and my girl friend, JoJo Ho, and I want to express my highest appreciation for their support and understanding. This thesis is dedicated to them for assisting me to achieve the most important stage in my life. I never let them down and hope them happy forever.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As VLSI technology grows up rapidly, the design trend goes toward system-level integration and single-chip solution. The system-level design considerations can lead to more cost-effective realizations, but it also results in more design complexity and design efforts. However, the design cycle remains the same or even shorter due to the time-to-market issue. Thus in System-On-Chip (SoC) designs, each module should better be reusable and process portable, so that total design time of SoC can be reduced. Unfortunately, some timing critical blocks, such as: Phase-Locked Loops (PLLs) and Delay-Locked Loops (DLLs) often are not reusable and process portable.

PLL and DLL are widely used in SoC designs. They are often applied to communication applications, such as: frequency synthesizer, clock multiplier, Clock and Data Recovery (CDR) circuit, and clock de-skew applications. They are very essential for current SoC designs, and for different modules, they may have different design specifications or requirements for PLLs and DLLs. As a result, how to design these PLLs/DLLs in a more efficient way becomes more and more important in these days.

For example, in single chip cable modem residential gateway [47], there are seven PLLs and over one hundred internally generated clocks used in this single chip gateway. Thus how to design these PLLs in an efficient way becomes very important for reducing system turn around time. And the importance to develop a systematic PLL/DLL design approach is required in current SoC era.

The design of PLL/DLL is a trade-off among jitter performance, frequency / phase resolution, lock-in time, area cost, power consumption, circuit complexity and design time. It is hard to design one PLL/DLL suitable for all applications. It often needs to redesign the PLL/DLL for target applications. If a wide-range PLL/DLL is designed for SoC applications, it can be used in more modules without modify it. But this scheme may waste unnecessary area cost and power consumption due to the requirement for wide-range operation. Thus a systematic design methodology to design user-specified PLLs/DLLs in a short time is necessarily for current SoC designs.

All-digital and cell-based approach is preferred for SoC applications. It can reduce both design time and design complexity for PLL/DLL. And this approach is also suitable for automated synthesis of All-Digital Phase-Locked Loop (ADPLL) and All-Digital Delay-Locked Loop (ADDLL).

However, due to the limitations of cell-based design, it is difficult to design a low-jitter, low-power, and high resolution ADPLL/ADDLL. Thus how to overcome the limitations of standard cells to build up a high resolution delay cell and high sensitivity frequency/phase detector (PFD) are the important design challenges for our research.

In this thesis, we propose the flexible ADPLL/ADDLL architectures for a truly portable and cost-effective ADPLL/ADDLL [35] solution. And we explain the proposed scheme to overcome the limitations of the standard cells and to build up high resolution delay cell and high sensitivity phase detector.

Time-to-Digital Converter (TDC) is widely used in the proposed ADPLL/ADDLL architecture to reduce frequency/phase acquisition time and achieve fast lock-in time. In all-digital multi-phase clock generator (ADMCG) [46], it is used to choose a suitable delay range to avoid false-lock to harmonics.

In this thesis, an automated synthesis methodology for ADPLL/ADDLL/ADMCG design is presented. Thus these modules now can be automatically generated using standard cells from cell-library. As a result, the proposed methodology uses both benefits of digital VLSI and cell-based design to build up user-specified ADPLL/ADDLL/ADMCG in a short time, and reduces design time and design complexity of ADPLL/ADDLL/ADMCG, making it very suitable for System-On-Chip (SoC) applications.

1.2 Thesis Organization

In this dissertation, we focus on algorithms, architectures, and circuits of high performance, low power, and area efficient ADPLL/ADDLL/ADMCG designs. And for SoC applications, we proposed an automated synthesis methodology to take design specifications and automatically generate the gate-level netlist for ADPLL/ADDLL/ADMCG. The proposed methodology can greatly reduce design time and design complexity of these modules in SoC era.

In chapter 2, we present the common timing critical modules design for ADPLL/ADDLL /ADMCG. The resolution of delay cell and the sensitivity of PFD have large influences on the jitter performance of the output clock, and they also have large effects on the final frequency error or phase error. In this chapter, we explain the proposed scheme to overcome the limitations of the standard cells and to build up high resolution delay cell and high sensitivity phase detector.

In chapter 3, the design for all-digital phase-locked loop (ADPLL) is presented. To reduce the system turn around time for SoC design, the ADPLL using the proposed high resolution delay cell and the proposed high sensitivity PFD is presented. The proposed ADPLL controller can achieve fast lock-in time, and the proposed delay cell can reduce both cost and design time for building a high resolution cell-based DCO. And the proposed PFD can improve the jitter performance and reduce the frequency error for the output clock. Moreover the flexible ADPLL architectures for different target applications are presented in this chapter for a truly portable and cost-effective ADPLL-based frequency synthesizer solution.

In chapter 4, the design for all-digital delay-locked loop (ADDLL) is presented. As the speed and the complexity of VLSI system increases rapidly, clock skew and clock jitter effects become more and more important now. It is difficult to design a DLL to overcome total effects caused by process, voltage, temperature, and loading (PVTL) variations. As in conventional DLL, the operation range of DLL is very limited. In this chapter, we propose an all-digital fast-locking DLL. The proposed DLL utilizes Time-to-Digital Converter (TDC) circuit and Digital-to-Time Converter (DTC) circuit to complete coarse-tuning in one clock cycle, resulting in less lock-in time. And the area cost for TDC and DTC can be shared with delay line. The proposed Digital-Controlled Delay Line (DCDL) architecture can turn off unused delay cells at high frequency operation, thus it is very suitable for wide-range clock deskew applications demanded in system-level integration.

In chapter 5, the design for an all-digital multi-phase clock generator (ADMCG) is presented. Multi-phase clocks are useful in many applications to process data streams at the bit rate higher than internal clock frequencies. But there are two problems of conventional DLLs. One is their limited phase capture range, and the other is restricted Voltage-Controlled Delay Line (VCDL) range to avoid false-lock to harmonics. Thus in this chapter, a new DLL-based approach for multi-phase clock generation is presented. The proposed ADMCG uses a TDC to choose a reasonable delay range rather than to use self-correcting circuit. Thus its operation is very robust and can avoid possible false-lock as in conventional designs. The lock-in time of the proposed ADMCG can also be reduced by adding TDC module. After TDC operation, a fixed step search scheme is used in the ADMCG to fine-tune the output phase accuracy.

In chapter 6, an automated synthesis design methodology for ADPLL/ADDLL /ADMCG is presented. In SoC design, the design time for each module is restricted. Thus each module should better be a reusable design so that the total design time for the SoC can be reduced. However, for different applications, the ADPLL/ADDLL/ADMCG may have different operating ranges or different lock-in time requirements, making it hard to design one ADPLL/ADDLL/ADMCG suitable for all applications. As a result, it often needs to redesign the ADPLL/ADDLL/ADMCG for target application and design phase becomes longer. Thus in this chapter, a proposed automated synthesis methodology uses both benefits of digital VLSI and cell-based design to build up user-specified ADPLL/ADDLL/ADMCG in a short time, making it very suitable for System-On-Chip (SoC) designs.

In chapter 7, we make conclusions and describe several design issues needed to be further explored in the near future.

Chapter 2

Timing Critical Modules in ADPLL and ADDLL Architecture

In this chapter, the design for delay cells and phase detectors are presented. They are the important modules in ADPLL/ADDLL design. The resolution of the delay cells and the sensitivity of the phase detector have large influences on the jitter performance of the output clock. They also have large effects on the final frequency error or phase error. As a result, they need to be designed carefully. And since we want to propose a cell-based ADPLL/ADDLL design, how to overcome the limitations of the standard cells to build up a high resolution delay cell and high sensitivity phase detector are the important design challenges for our research.

The delay cells are used to construct a ring oscillator and produce the desired output frequency in ADPLL design. In ADDLL design, delay cells are used to build a delay line and outputs the delayed version of the input clock. The delay cell must be easy to adjust its delay time, and the resolution of the delay cell should be sufficient enough to meet the requirements for the output clock.

The phase detector is used in ADPLL/ADDLL design to detect the phase error between the reference clock and output clock. The phase detector’s dead zone may cause the wrong operations in phase acquisition process. Thus how to reduce the dead zone of the phase detector is very important to improve the stability for the ADPLL/ADDLL and also minimize the static phase error.

The organization of this chapter is as follows. In section 2.1, the proposed high resolution delay cell is presented. And how to use the standard cells to build up this delay cells is also explained in this section. In section 2.2, the proposed high sensitivity phase and frequency detector (PFD) is presented. The proposed PFD is also constructed by standard cells. Finally, a brief summary is made in section 2.3.

2.1 Design of High Resolution Delay Cells

In analog delay cells, the delay cell is controlled by control voltage/current, and the output delay time is continuous over the controllable range. But in digital-controlled delay cell, the output delay time is quantized. And the resolution of the output delay time must be sufficient enough to meet the design specifications.

When two nearby digital control codes are applied to the delay cell, the output delay time difference is defined as the resolution of the delay cell. If an inverter-based (or buffer-based) delay line is used in delay cell design, since in this architecture, the delay cell produces different propagation delays by selecting different number of inverters (or buffers). Then the resolution of the delay cell is limited by the delay time of one inverter (or one buffer). And this resolution is often not sufficient to be used in ADPLL/ADDLL design.

Thus in previous designs, the phase blender is proposed in [19] to provide a better resolution by phase interpolation. And the delay matrix, which uses parallel tri-state buffers to enhance the resolution of the delay cell, is proposed in [8]. However, the intrinsic delay time of the phase blender is too large to be used in high-speed applications. This is because that in phase blender architecture, high resolution means the increasing of phase blender stages. And the area cost and the power consumption for the delay matrix is too large to be used in a low cost and low power design.

In this section, a low cost and high resolution delay cell is presented. The schematic of the proposed delay cell is shown in Fig 2.1. The function of this delay cell acts as an inverter regardless of (A, B) input value. And the value of (A, B) determines the delay time of the proposed delay cell.

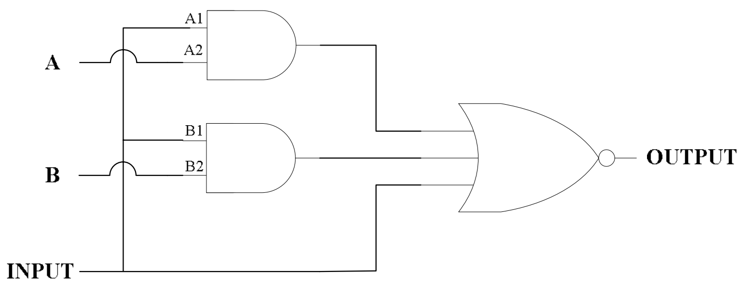


Fig. 2.1: The schematic of the proposed delay cell.

The CMOS circuit realization of the proposed delay cell is shown in Fig. 2.2.

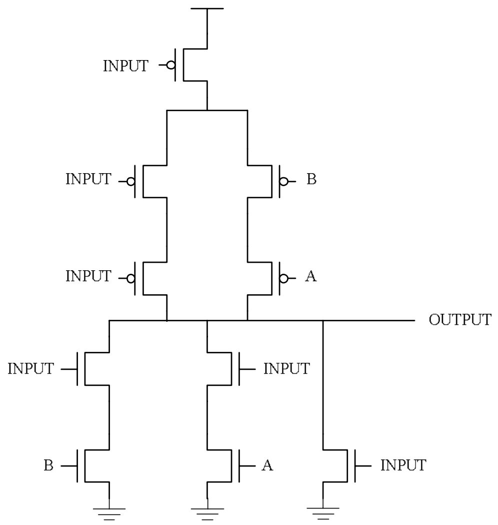


Fig. 2.2: The CMOS circuit of the proposed delay cell.

The delay time when the input changes from 0 to 1 and the output changes from 1 to 0 is defined as T_{PHL}. Oppositely, the delay time when the input changes from 1 to 0 and the output changes from 0 to 1 is defined as T_{PLH}.

In Fig. 2.2, when (A, B) is equal to (1, 1), the T_{PHL} of the delay cell is always smaller than the cases: (A, B) = (0, 1) or (1, 0). And when (A, B) = (1, 0) or (0, 1), the T_{PHL} of the delay cell is always smaller than the case: (A, B) = (0, 0). This is because the change of (A, B) value determines that how many pull-down paths will be turned on/off. So the resolution enhancement can be achieved by using the proposed delay cell.

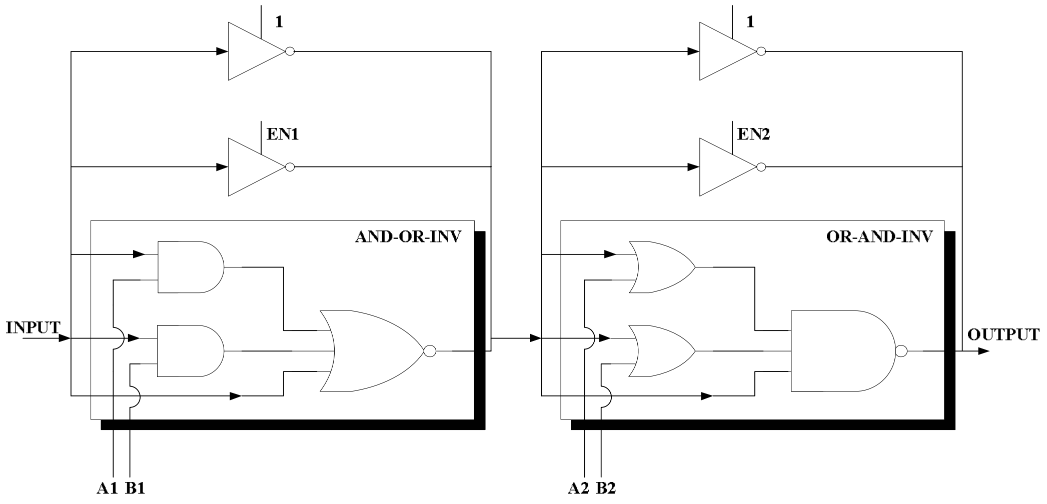


Fig. 2.3: The proposed high resolution delay cell.

In Fig. 2.1, this delay cell is named as an AOI (AND-OR-INVERSE) type delay cell, and it is a basic cell and can be found in standard cell library. The control signals can be used to adjust the T_{PHL} of the AOI type delay cell. Oppositely, the OAI (OR-AND-INVERSE) type cell can be controlled to adjust the T_{PLH} of the delay cell.

If those two type delay cells are cascaded, the resolution of the delay cell can be further enhanced. Fig. 2.3 shows the proposed high resolution delay cell, and it can be used to perform fine-tuning in ADPLL/ADDLL design. Both AOI type delay cell and OAI type delay cell are shunted with two tri-state buffers. And shunted tri-state buffers are used to increase the controllable range of the proposed high resolution delay cell.

In the proposed high resolution delay cell, in total six bits: (EN1, A1, B1, EN2, A2, B2) can be used to control the delay cell. Thus, in total 64 (=2^6) different delays can be provided. To design the proposed delay cell, first, the suitable standard cells must be selected from the cell library. And the SPICE circuit simulation for the proposed delay cell must be performed to measure the delay time (T_{PHL} and T_{PLH}) of the delay cell for all possible 64 cases. After SPICE simulation, the lookup table for mapping the binary control code versus delay cell control code can be created.

Table 2.1: CONTROL CODE VS. DELAY TIME OF THE PROPOSED DELAY CELL TIME UNIT: (PS), CODE: (EN1 A1 B1 EN2 A2 B2)

$T_{PHL}+T_{PLH}$	CODE	$T_{PHL}+T_{PLH}$	CODE	$T_{PHL}+T_{PLH}$	CODE	$T_{PHL}+T_{PLH}$	CODE
0802.7026	000 011	0674.2368	000 100	0622.3394	010 111	0580.2200	111 001
0777.6740	000 010	0672.7012	001 111	0622.0947	101 001	0578.2402	110 000
0759.8178	000 001	0670.2695	100 010	0620.1069	011 111	0577.0692	100 101
0759.3041	001 011	0667.4821	101 011	0613.8635	101 000	0572.2359	111 000
0751.3155	000 000	0666.1494	010 001	0611.3577	100 111	0568.2607	100 100
0736.0601	001 010	0663.9600	011 001	0605.6821	110 010	0563.3331	101 110
0718.9363	001 001	0658.9083	010 000	0604.3907	010 110	0549.1761	110 111
0714.0263	000 111	0657.2062	011 000	0602.3159	011 110	0549.1071	101 101
0711.8740	001 000	0655.3770	001 110	0599.4157	111 010	0543.4591	111 111
0707.5820	010 011	0650.9609	100 001	0591.4753	100 110	0540.5546	101 100
0704.4837	011 011	0642.2144	001 101	0591.0464	010 101	0529.1033	110 110
0697.3336	100 011	0642.0410	100 000	0589.1943	011 101	0523.2402	111 110
0695.9196	000 110	0641.0289	101 010	0586.3467	110 001	0514.6414	110 101
0683.6460	010 010	0634.8068	001 100	0583.3829	010 100	0508.8459	111 101
0682.2107	000 101	0632.4963	110 011	0583.0358	101 111	0506.1100	110 100
0681.0558	011 010	0626.2559	111 011	0581.7245	011 100	0500.2978	111 100

Table 2.1 lists the simulated delay time of the proposed high resolution delay cell versus delay cell control codes. In this simulation, a standard 0.35μm 1P4M CMOS process cell library is used to construct the delay cell.

If the proposed delay cell is applied to be the fine-tuning stage of the DCO. Then the value of ($T_{PHL}+T_{PLH}$) means the change of the output clock period. The resolution of the DCO can be improved to the resolution of the proposed delay cell. And in this simulation, the average resolution of the proposed delay cell is about 5ps = ((802.7026 ps - 500.2978 ps)/64).

Since in the proposed delay cell, there only six standard cells are used. Thus its area cost and power consumption is very low, and its resolution is also sufficient to be used in ADPLL/ADDLL design.

2.2 High Sensitivity Phase and Frequency Detector Deign

The phase detector (PD) can detect the phase error between the reference clock and the output clock. And it generates the up or down pulse to control the ADPLL/ADDLL. In analog phase-locked loop or analog delay-locked loop, the pulse width of the up/down pulse, which means the amount of the phase error, controls the charge/discharge time for the charge pump capacitor. However, in ADPLL and ADDLL, only the polarity information (i.e. lead or lag) is taken from the phase detector, and thus the pulse width information is not used. This is because the ADPLL/ADDLL controller is a cycle-based finite state machine. Thus it is not possible to use the pulse width information for the ADPLL/ADDLL controller unless some circuits likes the time-to-digital converter (TDC) is used [39].

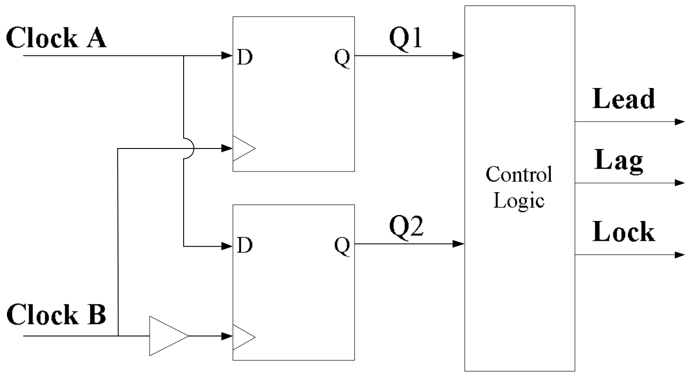


Fig. 2.4: The sample-based phase detector.

Since the phase detector used in ADPLL/ADPLL design only needs to detect the polarity information. The sample-based phase detector is often used as in previous designs [18]. In those phase detectors, they determine lead/lag information by using output clock as the sampling clock to sample the reference clock.

The sample-based phase detector is shown in Fig. 2.4. Clock B is output clock and clock A is the reference clock. Both clock B and its delayed one create a sampling window for detecting the transition of clock A. If the sampling values (Q1, Q2) are different, the transition of the clock A is detected. And then the control logic can use the sampled values to generate the lead or lag information for the ADPLL/ADDLL controller.

But when the reference clock edge (clock A) is very close to the output clock edge (clock B), the phase detector may produce wrong lead/lag information for the ADPLL/ADDLL controller. The minimum phase error, which can be detected by the phase detector, is called the dead zone of the phase detector. If standard cells are used to construct the phase detector, the resulting dead zone is often too large to be used in ADPLL/ADDLL design. The limitations of this type phase detector often comes from the timing requirements (such as: setup time and hold time) for the D-Flip/Flop.

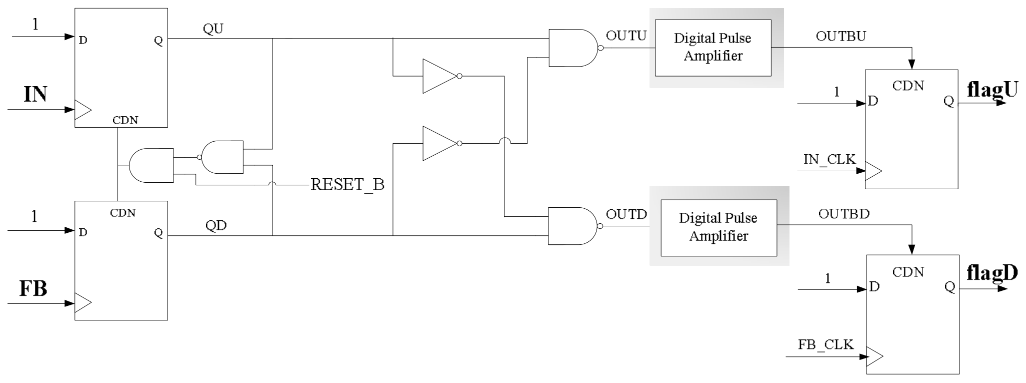


Fig. 2.5: The proposed high sensitivity phase and frequency detector.

To overcome the disadvantages and limitations of the traditional sample-based phase detector, a new phase and frequency detector (PFD) is presented. Fig. 2.5 shows the schematic of the proposed high sensitivity PFD. The proposed PFD consists of one three-state phase detector and two digital pulse amplifiers. The simulation waveform of the proposed high sensitivity PFD is shown in Fig. 2.6. When output clock (FB) leads reference clock (IN), flagD produces a low pulse and flagU remains at high. Oppositely, when output clock (FB) lags reference clock (IN), flagU produces a low pulse and flagD remains at high. And those flagU and flagD signals are used to control the operation of the ADPLL/ADDLL controller.

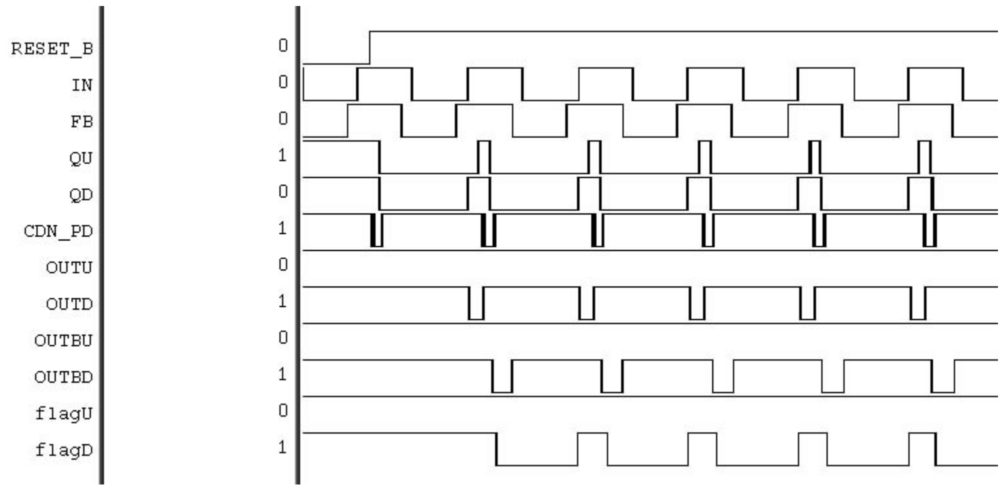


Fig. 2.6: The simulation waveform of the proposed PFD.

In Fig. 2.5, when the output clock (FB) is very close to the reference clock (IN), the pulse width of three-state phase detector's output (i.e. OUTU or OUTD) becomes very small. This narrow low pulse (OUTU or OUTD) can not correctly clear the D-Flip/Flop's output (flagU or flagD). Thus the dead zone of the proposed PFD is limited by this minimum pulse width requirement for the D-Flip/Flop's clear pin (CDN). This timing requirement is often about several hundred pS in typical standard cell library.

To improve the sensitivity of the proposed PFD, two digital pulse amplifiers are connected at the output of three-state phase detector (i.e. OUTU or OUTD). The schematic of the digital pulse amplifier is shown in Fig. 2.7. It uses the cascaded two-input ANDs architecture to increase the low pulse width of OUTU and OUTD. The output pulses are named as OUTBU and OUTBD. The digital pulse amplifier enlarges the phase error between reference clock (IN) and output clock (FB), thus the following D-Flip/Flops can detect it.

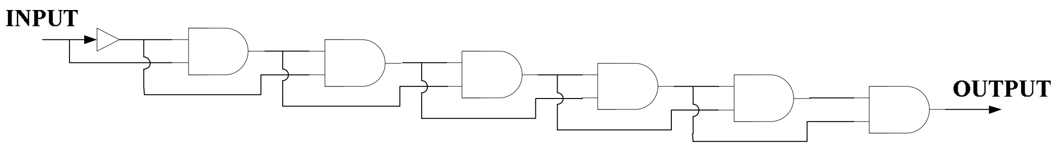


Fig. 2.7: The digital pulse amplifier.

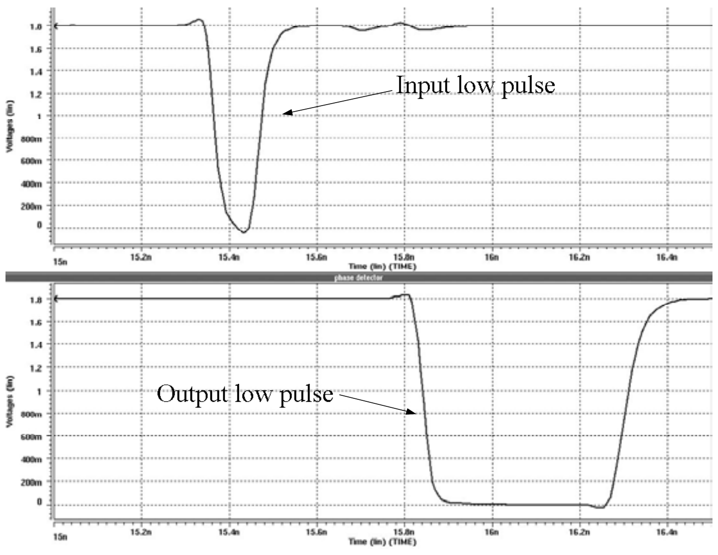


Fig. 2.8: The SPICE simulation waveform of the digital pulse amplifier.

Fig. 2.8 shows the SPICE circuit simulation waveform of the digital pulse amplifier. The input is delayed and is “AND” with itself, thus the pulse width of the low pulse is extended. Ideally, increasing the number of “AND” stages can increase the sensitivity of the proposed PFD. But when the phase error between reference clock and output clock becomes very small, the low pulse at OUTU and OUTD may be disappeared or may be not full swing voltage. In this case, the digital amplifier becomes useless. So the real sensitivity of the proposed PFD can be determined after SPICE simulation.

In the simulation which is shown in Fig. 2.8, a standard 0.18μm 1P6M CMOS process cell library is used to construct the proposed high sensitivity PFD. From SPICE simulation, the input low signal whose pulse width is larger than 20ps will be increased to more than 600ps to meet the minimum pulse width requirement of the D-Flip/Flop's clear pin. As a result, the sensitivity of the proposed PFD can be improved to ±20ps, and this high sensitivity ability is very suitable to be used in ADPLL/ADDLL design.

Besides detecting the phase error, the proposed PFD also has the ability to detect the frequency difference between reference clock and output clock. Both flagU and flagD will be clear to high after the next rising edge of reference clock and output clock occurs respectively. Thus if the frequency of the output clock is higher than reference clock, the flagD will have more numbers of low pulse than flagU, and this will cause the ADPLL controller to slow down the DCO's output frequency. So the proposed PFD is also suitable to be used in ADPLL design to detect frequency error.

2.3 Summary

In this chapter, the timing critical modules for ADPLL and ADDLL are presented. The proposed delay cell architecture overcomes the limitations of using the standard cells to construct the high resolution delay cell, and it also has lower cost and lower power consumption than the traditional designs. Thus it is very suitable to be used in ADPLL and ADDLL. The proposed high sensitivity PFD is also presented in this chapter. The proposed PFD uses the digital pulse amplifier to improve the sensitivity of the traditional three-state phase and frequency detector, and thus it is also very suitable to be used in the ADPLL and ADDLL design.

Chapter 3

All-Digital Phase-Locked Loop Design

In this chapter, the design for all-digital phase-locked loop (ADPLL) is presented. The phase-locked loops (PLL) are widely used for many communication applications, such as frequency synthesizer, clock multiplier, clock and data recovery circuit, or input clock jitter filtering. It had become an essential function block for current System-On-Chip (SoC) design. And how to design the PLL in a more efficient way is a very important design issue for chip makers.

Due to high integration of VLSI system, PLL often operates in a very noisy environment. The jitter less than ±4% of the clock cycle time is typically needed to avoid functional failures in a microprocessor [1]. However, the digital switching noise coupled through power supply and substrate induces considerable noise into noise-sensitive analog circuits [1,3-7].

Traditionally, many analog approaches are proposed to improve the jitter performance of PLL, such as: choosing a narrow bandwidth or using a low gain Voltage-Controlled Oscillator (VCO) [3]. However, those analog approaches often result in long lock-in time and increasing design complexity of PLL.

In recently years, ADPLL became more attractive since they yield better testability, programmability, stability and portability over different processes [8,9]. And they can reduce the system turn around time. But the jitter performance and the frequency resolution of the traditional ADPLLs are not as well as analog PLLs. This is because it is difficult to design the high resolution digital-controlled oscillator (DCO) and high sensitivity phase and frequency detector (PFD).

However, the ADPLL has the ability to achieve fast-locking in a short period, and it is very suitable for fast-locking applications. If the jitter performance of the ADPLL can be improved, then the ADPLL can be used in more applications. Thus how to design a fast-locking ADPLL with low-jitter performance in a short design time is the goal for this research.

In this chapter, an ADPLL using the proposed high resolution delay cell and the proposed high sensitivity PFD is presented. The design for the delay cell and the PFD was already discussed in chapter 2. The proposed delay cell can reduce both cost and design time for building a high resolution cell-based DCO. And the proposed PFD can improve the jitter performance and reduce the frequency error for the output clock. Moreover the flexible ADPLL architectures for different target applications are presented in this chapter for a truly portable and cost-effective ADPLL-based frequency synthesizer solution.

The organization of this chapter is as follows. In section 3.1, the overview of PLL is discussed. In section 3.2, the design trade-off between different PLL architectures is discussed. In section 3.3, the proposed ADPLL architecture is presented. In section 3.4, the test chip which implements of the proposed ADPLL using standard 0.35μm CMOS process is presented. In section 3.5, the simulation and measurement results of the test chip are presented. Finally, a brief summary is made in section 3.6.

3.1 Overview of Phase-Locked Loop (PLL)

The general PLL architecture is shown in Fig. 3.1. The input of PLL is reference clock and programmable divide ratio M, and output clock outputs from the internal oscillator. The PLL keeps tracking the frequency and the phase of reference clock, and it speeds up or slows down the internal oscillator to minimize frequency and phase error between divided output clock (Out_divM) and reference clock. After PLL is locked, the frequency of output clock is M times higher than reference clock, and the phase of output clock is synchronized with reference clock.

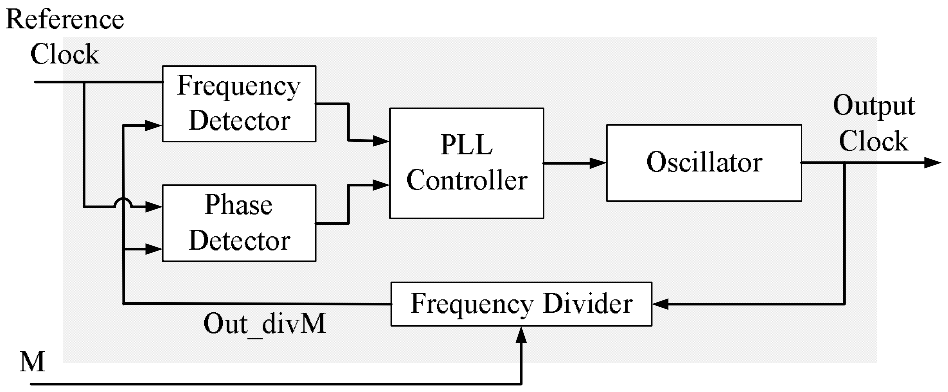


Fig. 3.1: The general PLL architecture.

The PLL lock-in process is separated into frequency acquisition and phase acquisition. The frequency detector detects the frequency difference between reference clock (Reference Clock) and divided output clock (Out_divM). And the PLL controller controls the internal oscillator’s output frequency to minimize frequency error. Thus high resolution oscillator is needed to generate accurate frequency output. After frequency acquisition is completed, the PLL turns into phase acquisition and phase maintaining mode.

The lock-in time of PLL is mainly determined by the frequency acquisition time, thus how to reduce frequency acquisition time is very important to a fast-locking PLL design. In previous designs, the adaptive gain control PLL [2,37] or the TDC-based ADPLL [39] are proposed to speed up the frequency acquisition process.

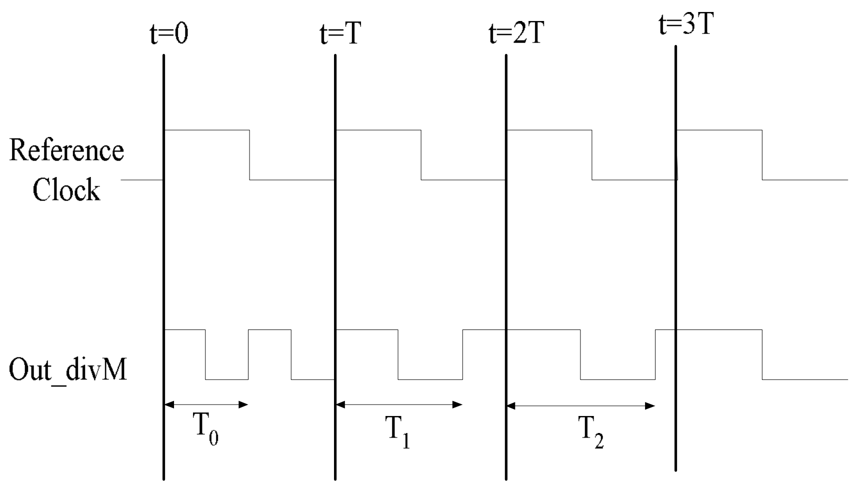


Fig. 3.2: The frequency acquisition process of PLL.

Fig. 3.2 illustrates the frequency acquisition process of PLL. If we assume the oscillator is reset to high at every rising edge of Reference Clock, and the output period of Out_divM is F(t), where t is time. And we assume at time=0, the frequency detector finds that output frequency of oscillator is higher than Reference Clock. The PLL controller will control the internal oscillator to slow down.

If step size for frequency acquisition is F_STEP, then the output period of Out_divM at t=T becomes $F(T)=T_0$ + F_STEP, and the output period of Out_divM at t=2T becomes F(2T)= T_1 + F_STEP = T_0 + 2*F_STEP. The general form to determine the output period of Out_divM at time t can be expressed as Eq. 3.1.

$$\begin{aligned} F(nT) = &\ F((n-1)T) - F\_STEP, \text{ if } F((n-1)T) > T, \\ &\ F((n-1)T) + F\_STEP, \text{ if } F((n-1)T) < T \\ &\ F((n-1)T), \text{ if } F((n-1)T) = T \qquad \text{(Eq. 3.1)} \end{aligned}$$

When F(t) becomes T or the frequency detector can not distinguish the frequency difference between Reference Clock and Out_divM, the frequency acquisition is done.

After frequency acquisition is completed, the PLL starts to trace the phase of reference clock. Fig. 3.3 illustrates the phase acquisition process of PLL. If we assume the frequency error between Reference Clock and Out_divM is zero after frequency acquisition, and the phase error between Reference Clock and Out_divM is P(t), where t is time. The initial phase error is P(0). And we assume at time=0, the phase detector finds that Out_divM lags behind Reference Clock. The PLL controller will control the internal oscillator to speed up.

If step size for phase acquisition is P_STEP, then the phase error at t=T becomes P(T) = P(0) + T_0 – T= P(0) – P_STEP. And the phase error at t=2T is P(2T) = P(T) + T_1 – T = P(0) – P_STEP – 2*P_STEP. The general form of phase error at time t can be expressed as Eq. 3.2.

$$P(nT) = P(0) - \sum_n n * P\_STEP \qquad \text{(Eq. 3.2)}$$

where we assume that the Out_divM still lags behind the Reference Clock after several updates of the oscillator frequency. When P(t) becomes negative or zero, the phase acquisition process is completed.

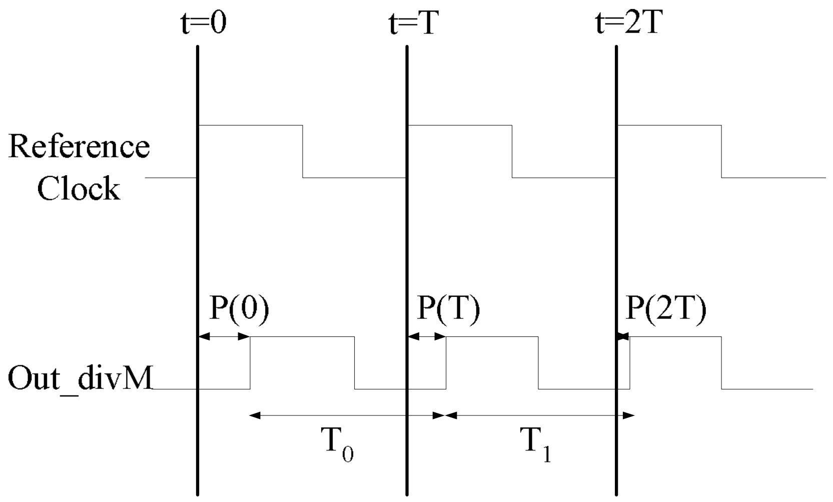


Fig. 3.3: The phase acquisition process of PLL.

After phase error is eliminated, phase acquisition process is completed. PLL controller will restore oscillator’s frequency back to the baseline frequency which is determined by frequency acquisition process, and the PLL turns into phase maintaining mode. In this phase maintaining mode, the PLL keeps tracking the phase of reference clock by fine-tuning oscillator’s output frequency.

In phase acquisition process, phase detector must provide correct phase relationship information about reference clock and divided output clock. The dead zone of phase detector will increase phase acquisition time and final phase error. Thus how to design a high sensitivity phase detector is a design challenge for PLL design. Since the operating range and output frequency accuracy is determined by the internal oscillator, the design for high resolution oscillator is also an important design challenge for PLL design.

3.2 Design Trade-Off in Different PLL Architectures

PLL design is a trade-off among jitter performance, lock-in time, area cost power consumption, circuit complexity and design time. Thus it is hard to design one PLL suitable for all applications. For fast-locking frequency synthesizer applications, such as a frequency hopping multiple access systems, the lock-in time is the most critical design issue. And for portable or mobile applications, lock-in time is also very important since the PLL must support fast entry and exit from power management techniques [9].

In traditional analog PLL designs, fast acquisition requires tuning of the Voltage-Controlled Oscillator (VCO) free-running frequency near the desired frequency in advance or to increase loop bandwidth. But increasing the loop bandwidth degrades jitter performance, and the exact VCO tuning range is not easy to be achieved since there always has process variations, voltage variations, and temperature variations (PVT variations).

Thus a Digital Frequency-Difference Detector (DFDD) is proposed in [2] to convert the frequency difference directly to the digital value, and then change the gain for VCO control adaptively. The dual-loops PLL architecture [37] uses one loop for fast tracking the suitable frequency range and the other loop fine-tuning the output. Both of them proposed a concept that for fast lock-in time, the VCO gain or the loop bandwidth should be increased during the frequency acquisition process. And after frequency is locked, it should return to its normal value to preserve the low jitter performance. But the circuit complexity is increased due to this adaptive gain control ability.

A different way to achieve fast lock-in time is proposed in [38]. It uses a digital hybrid PLL (DH-PLL) with Digital Look-up Table (DLT) to shorten settling time and achieve fast switching speed at every frequency synthesis. This design uses a DLT to directly adjust VCO output to the desired frequency, and then uses a traditional analog PLL to fine-tune the output. However, this digital look-up table is still dependent on PVT variations. As a result, acquisition time increases in proportion to the initial frequency difference.

From the previous PLL architectures [2,37,38], the methods for fast-locking PLL design can be classified into two types: one uses an adaptive gain control for the frequency acquisition process, and the other uses a look-up table to speed up the frequency acquisition process.

To further speed up the lock-in time, an all-digital phase-locked loop (ADPLL), which uses a Time-to-Digital Converter (TDC) circuit to quantize the reference clock period into multiples of inverter delay times, is proposed in [39]. This PLL replaces the DLT [38] by TDC to against PVT variations and speeds up the frequency acquisition process. Since the TDC and the DCO are suffered from the same PVT variations, the TDC measured value is more accurately than the DLT [38], and the lock-in time of the PLL can be further reduced. But the area cost for the TDC digital processing unit is a problem if a small chip area is required.

For clock multiplier applications, the phase error between reference clock and output clock is very important. Since the ring oscillator has jitter accumulation problem, it is not easy to minimize the phase error. In ADPLL [9], the anchor register is used to store the baseline frequency, thus the ADPLL controller can keep tracking the phase of the reference clock. But the phase error of the PLL-based clock multiplier may become worst when the input jitter from the reference clock exists or the multiplication ratio is increased.

The DLL-based clock multiplier [40-42], which generates the output clock from the delayed version of reference clock, can efficiently reduce the phase error. But it is not suitable for a programmable design since the multi-phase delay line is not a scalable design. So the PLL-based clock multiplier is still more flexible than the DLL-based clock multiplier.

The Digital Controlled PLL (DCPLL) [2] has been proposed to achieve fast lock-in time. But due to low sensitivity of frequency detector and resolution limitation of D/A converter, its jitter performance is worse than analog designs. An ADPLL proposed in [9] can achieve fine resolution and fast lock-in time. However its DCO needs to be full-custom designed, making it difficult for porting to different processes as design requests. A complete cell-based ADPLL is proposed in [8], where fine-search delay matrix architecture is developed to improve DCO's resolution. Also two DCOs are exploited to reduce output clock jitter effectively. However the proposed fine-search delay matrix occupies large silicon area and has high power consumption.

From the above discussions, a better ADPLL architecture should be easily modified to fit different applications. Thus in the next section, a flexible ADPLL architecture is proposed for most applications. The proposed ADPLL architecture takes the advantages of the TDC-based ADPLL [39] and the portability of the cell-based ADPLL [8] to build up a low-jitter, low-cost, fast-locking and cell-based ADPLL.

3.3 The proposed ADPLL Architecture

3.3.1 ADPLL Architecture Overview

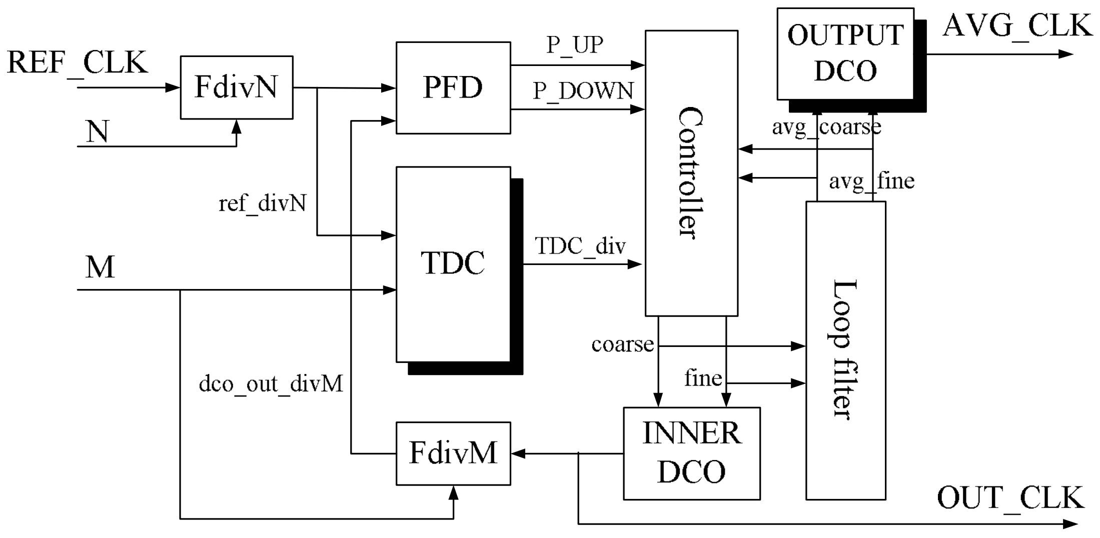


Fig. 3.4: The proposed ADPLL architecture.

The proposed ADPLL architecture is shown in Fig. 3.4. The ADPLL consists of Phase and Frequency Detector (PFD), Time-to-Digital Converter (TDC), ADPLL Controller, INNER DCO, OUTPUT DCO, loop filter, input frequency divider (FdivN) and feedback frequency divider (FdivM). M and N inputs are used for programming frequency divider and input divider respectively.

The PFD detects the frequency difference and phase error between divided reference clock (ref_divN) and divided INNER DCO's output clock (dco_out_divM), and it outputs up (P_UP) and down (P_DOWN) signal to indicate that the INNER DCO should speed up or slow down respectively. The ADPLL controller takes those control signals from the PFD and performs update of the DCO control code (coarse, fine). This DCO control code is also sent to the loop filter. After ADPLL is locked, the DCO control code is converged to the fine-tuning range. And then both frequency acquisition and phase acquisition are achieved.

The loop filter takes DCO control code from the ADPLL controller, and it detects the variations range of the DCO control code after ADPLL is locked, and outputs average control code (avg_coarse, avg_fine) for OUTPUT DCO. After ADPLL is locked, every time when the PFD’s output changes from up to down or vice versa, the ADPLL controller restores this average DCO control code from the loop filter for phase acquisition and phase maintaining.

In the proposed ADPLL architecture, the TDC is an optional module and is only used for fast-locking ADPLL. If the lock-in time is not a critical design issue or a smaller chip area is required, this module is removed from the architecture, and then the binary search ADPLL controller is used for frequency acquisition.

For frequency synthesis application, the loop filter can filter out DCO control code variations and controls OUTPUT DCO to provide a low-jitter clock output (AVG_CLK). For clock multiplier applications, OUTPUT DCO is removed from the architecture, and in-phase output clock directly outputs from the OUT_CLK.

3.3.2 Design for Binary Search ADPLL Controller

In binary search ADPLL controller, a binary search scheme is used when it searches for the target frequency. Fig 3.5 illustrates the frequency acquisition process. The frequency acquisition starts from middle frequency band of the DCO. If DCO can provide “n” different frequencies, the search step is “n/4” in the initial state. When output frequency is lower than target frequency, ADPLL controller adds current search step to DCO control code, and this increases the output frequency of DCO. Oppositely, when output frequency is higher than target frequency, ADPLL controller subtracts the DCO control code to lower the output frequency of DCO.

Whenever the PFD’s output changes from up (P_UP) to down (P_DOWN) or vice versa, the search step is divided by 2. And after the search step reduces to 1 (i.e. one fine-tuning step of the INNER DCO), the frequency acquisition is done. Then the ADPLL controller enters phase acquisition and phase maintaining mode. In this mode, the ADPLL controller adjusts the fine-tuning control code of the INNER DCO to eliminate the phase error between divided reference clock (ref_divN) and divided INNER DCO's output clock (dco_out_divM) whenever it receives the up (P_UP) or down (P_DOWN) from PFD.

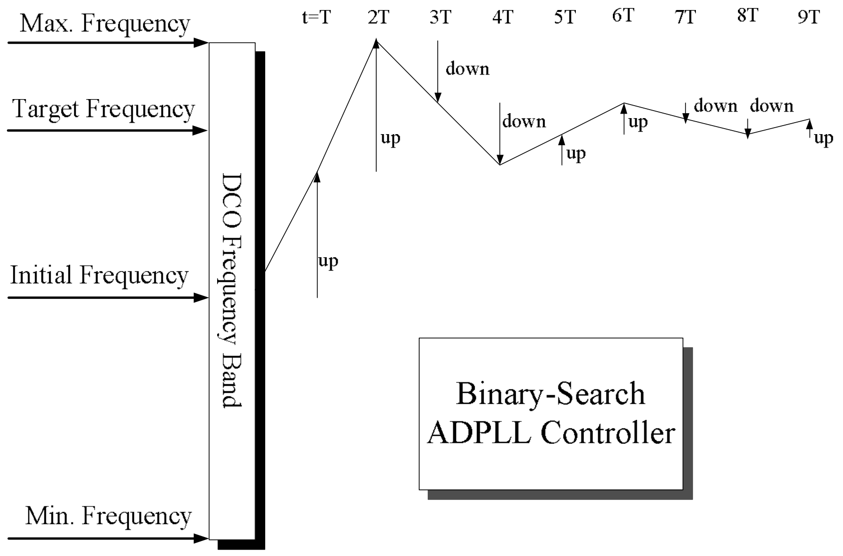


Fig. 3.5: The Binary search ADPLL controller.

The ADPLL's closed-loop response time is determined by the response time of DCO, delay time of ADPLL controller and frequency divider. Therefore DCO's control code can only be updated at every "m" cycles, instead of every reference clock cycle. Here "m" is determined by closed-loop's response time. In Fig. 3.5, it shows that the update period (T) for DCO control code is "m" reference clock cycles. Hence, the worst-case lock-in time for this frequency acquisition algorithm, in term of reference clock cycles can be express as Eq. 3.3.

$$T(n) = m*(1 + 2*\log_2(n/2)) = m*(2*\log_2(n) - 1) \qquad \text{(Eq. 3.3)}$$

3.3.3 Design for TDC-Based Fast-Locking ADPLL Controller

For fast-locking applications, lock-in time is the most critical design issue. Thus in the proposed architecture, TDC is used to quickly calculate the nearest control code for DCO to produce the desired frequency. TDC can convert the reference clock's period information to multiples of delay cell's delay time. Hence, ADPLL controller can use this information to quickly jump to the desired frequency band. And then ADPLL performs fine-tuning to reduce the residual frequency error and phase error. As a result, the lock-in time can be reduced by adding TDC module.

Fig. 3.6 shows the architecture of the proposed TDC for fast-locking ADPLL. To make sure that the TDC measured value can be directly applied to DCO, the ring delay line of TDC has a copied structure from the DCO with some reductions, and the difference is that only three coarse-tuning delay cells are used in TDC path selector. The detail DCO structure is discussed in section 3.4.1.

After system reset and with the first rising edge of reference clock, TDC is turned on (TDC_enable=1), and ring delay line of the TDC begins to oscillate. Output clock of the ring delay line triggers TDC counter to count up until the second rising edge of reference clock comes. Then TDC is turned off (TDC_enable=0).

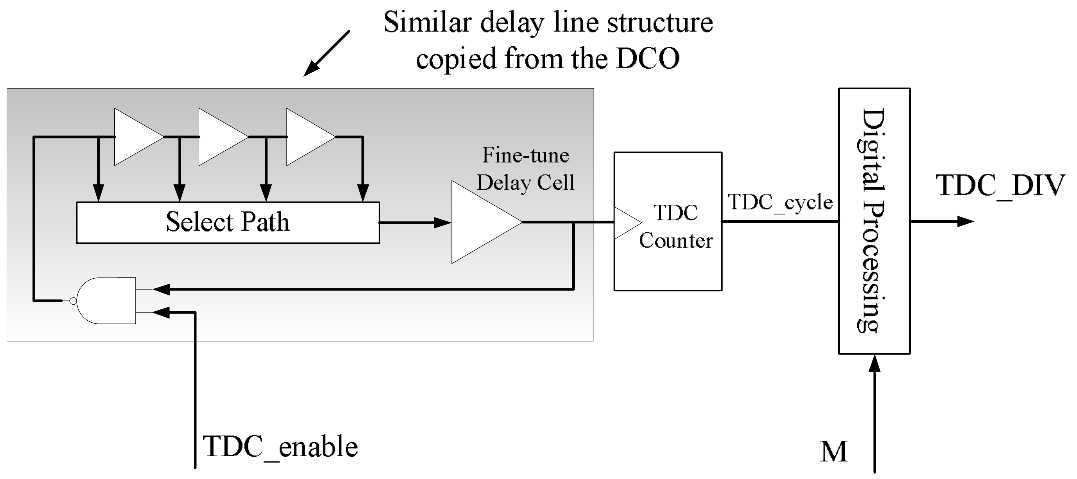


Fig. 3.6: The structure of the proposed TDC for fast-locking ADPLL

The TDC counted value: TDC_cycle means that the reference clock's period can be quantized as the multiples of $T_{TDC\text{-}ring}$, where $T_{TDC\text{-}ring}$ means the oscillation period of the TDC ring delay line. The value of $T_{TDC\text{-}ring}$ can be expressed as Eq. 3.4.

$$T_{TDC\text{-}ring} = 3*COARSE\_UNIT + T_{select\text{-}path} + T_{Fine\text{-}tune} + T_{RESET} \qquad \text{(Eq. 3.4)}$$

where $T_{select\text{-}path}$ is delay time of path selector and $T_{Fine\text{-}tune}$ is the delay time of fine-tuning delay cell, and T_{RESET} is the delay time for reset stage.

If n coarse-tuning delay cells are needed in DCO's coarse-tuning delay stage to produce the desired output frequency, output clock period of the DCO can be express as Eq. 3.5.

$$T_{DCO} = n*COARSE\_UNIT + T_{select\text{-}path} + T_{Fine\text{-}tune} + T_{RESET} \qquad \text{(Eq. 3.5)}$$

If we let $T_Z = T_{select\text{-}path} + T_{Fine\text{-}tune} + T_{RESET}$, and the desired output frequency should be M/N times of the reference clock frequency, and Eq. 3.6 must be satisfied.

$$M*T_{DCO} = T_{TDC\text{-}ring}*(TDC\_cycle/N)$$

$$\Rightarrow M*N*(n*COARSE\_UNIT + T_Z) = (3*COARSE\_UNIT + T_Z) * TDC\_cycle$$

$$\Rightarrow n = 3*(TDC\_cycle/M*N) + (T_Z/COARSE\_UNIT)*((TDC\_cycle-1)/M*N) \qquad \text{(Eq. 3.6)}$$

To reduce the circuit complexity for TDC, TDC_cycle-1 is reduced to TDC_cycle, and since T_Z contains five gate delays, T_Z/COARSE_UNIT is replaced by 5. Therefore the equation can be further reduced and expressed as Eq. 3.7.

$$n \cong 8*TDC\_cycle/(M*N) \qquad \text{(Eq. 3.7)}$$

Thus digital processing unit takes the counted value (TDC_cycle) from TDC counter and performs the calculation of Eq. 3.7, and then outputs the DCO control code (TDC_DIV) to ADPLL controller. The ADPLL controller takes this value as the initial DCO control code. After that, it fine-tunes the output by up (P_UP) and down (P_DOWN) control signals from PFD.

The bit width of TDC counter and TDC digital processing unit are determined by the maximum reference clock period (i.e the lowest reference clock rate). After user specifies the reference clock range, the bit width must be large enough to avoid overflow in TDC counter.

Since in Eq. 3.7, one divider is needed for calculation of the suitable DCO control code. Thus when TDC module is used for the ADPLL design, the area cost is increased, but the frequency acquisition time can be reduced.

The area cost for binary search ADPLL controller is much lower than TDC-based fast-locking ADPLL controller. Hence binary search ADPLL controller is very suitable for a low-cost ADPLL design and still has faster lock-in time than traditional analog PLLs. And for fast-locking applications, the TDC-based ADPLL controller is preferred.

3.4 The ADPLL Circuit Design

In the proposed ADPLL architecture, all functional blocks are implemented with standard cells. Thus design time for the ADPLL is reduced by the proposed cell-based architecture. And the limitations of the cell-based design are overcome by using the proposed high resolution delay cell and high sensitivity PFD.

3.4.1 Design for Digital-Controlled Oscillator

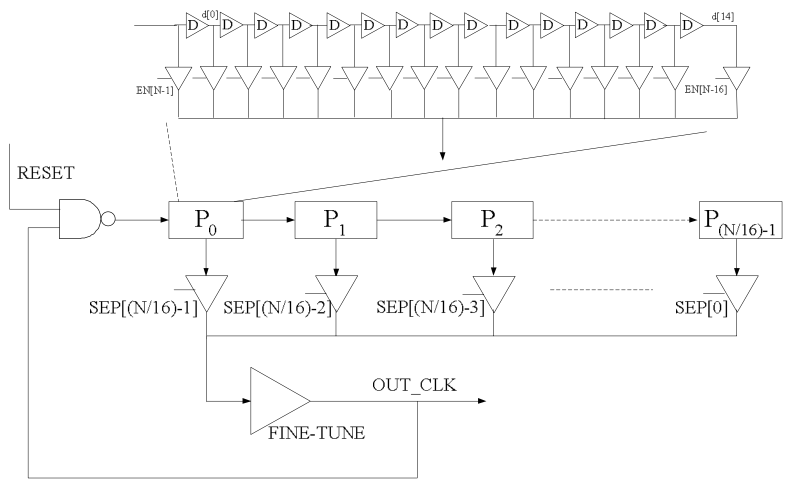


Fig. 3.7: The proposed cell-based DCO architecture.

The proposed cell-based DCO architecture is shown in Fig. 3.7. In the test chip, the DCO is implemented with TSMC 0.35μm 1P4M CMOS standard cell library. It is separated into two stages: coarse-tuning stage and fine-tuning stage.

In coarse-tuning stage, the coarse-tuning delay chain with 64-to-1 path selector architecture is used to provide different delays for coarse-tuning. The 64-to-1 path selector architecture is implemented with tri-state buffers. The DCO coarse-tuning encoder encodes 6 ($=\log_2(64)$) bits coarse-tuning control code into 64-bit one-hot path selection control signals. This architecture has advantage of minimum intrinsic delay time in path selector to improve maximum operating frequency of the DCO. And it can be easily modified to meet different specifications for different applications.

To avoid large loading capacitance appearing in the path selector's output, the path selector is partitioned into two stages. In the first stage, every sixteen coarse-tuning delay blocks will select a partial output. And the second stage path selector will select the final output. The delay time difference between two neighbor paths is determined by one coarse-tuning delay cell. The ($T_{\mathbf{PHL}} + T_{\mathbf{PLH}}$) of one coarse-tuning delay cell is about 300ps in the target process. Thus when DCO's coarse-tuning control code increases one or decreases one, the amount of output clock's period will be changed by ±300ps.

To increase frequency resolution of the DCO, fine-tuning delay cell is added after coarse-tuning stage. The circuit of fine-tuning delay cell is show in Fig. 2.3. And the detail information about how to design the fine-tuning delay cell is discussed in section 2.1.

The controllable range of fine-tuning delay cell should cover one coarse-tuning step (i.e. 300ps). And the DCO resolution can be improved to averagely 5ps by adding fine-tuning delay cell. The maximum output frequency of DCO is 545MHz (1.833ns) and minimum output frequency of DCO is 41MHz (24.261ns) by SPICE circuit simulation.

3.4.2 Design for Phase and Frequency Detector

The circuit of cell-based PFD is shown in Fig. 2.5, and the detail information about how to design PFD is discussed in section 2.2. After using the digital pulse amplifier to increase the sensitivity of PFD, phase error up to ±50ps can be detected in the target process. When phase error is less than the dead zone of PFD, there will have no trigger signals sent to ADPLL controller which remains unchanged in its previous state.

3.4.3 Loop Filter Design

After ADPLL has finished frequency acquisition and phase acquisition, INNER DCO's control code becomes converged to a fine-tuning range. However the control code may have small variations due to the following factors: PFD's dead zone, DCO's finite resolution and reference clock jitter. To further improve jitter performance of the APDLL for frequency synthesizer applications, loop filter is used to filter out the resultant noise into OUTPUT DCO.

Thus the loop filter detects the maximum INNER DCO control code and minimum INNER DCO control code within 512 reference clock cycles and then outputs (DCO control code $_{\textbf{(maximum)}}$ + DCO control code $_{\textbf{(minimum)}}$)/2 as the average DCO control code for the OUTPUT DCO. As a result, the jitter performance of the output clock can be improved.

But since phase relationship between OUTPUT DCO and INPUT DCO is unknown. Thus this two DCO structure can only be used in frequency synthesizer applications, where only accurate frequency output is needed. And for phase acquisition applications, OUTPUT DCO is removed from the structure, and the output clock directly achieved from INNER DCO.

The proposed loop filter circuit is very simple. It only needs two registers and one adder. But it can greatly reduce the noise effects and reference clock jitter effects.

3.4.4 A Systematic Approach for ADPLL Design

A systematic way is provided to design the ADPLL with specified standard cell library. Firstly, SPICE circuit simulation of the fine-tuning delay cell should be performed to construct lookup table for mapping fine-tuning control code.

When the controllable range of fine-tuning cell is determined, a suitable coarse-tuning cell, whose delay time ($T_{PHL}+T_{PLH}$) is less than or equal to the controllable range of fine-tuning delay cell, can be selected from cell library. And the specifications of output clock range determine the number of select paths in coarse-tuning stage.

Hardware Description Language (HDL) is used to describe ADPLL controller, frequency divider, and loop filter. We use logic synthesizer to synthesize those modules to gate-level circuits with TSMC 0.35μm 1P4M CMOS cell library. Thus design time and complexity for ADPLL can be reduced. And the proposed ADPLL architecture can easily be ported to different processes in a short time.

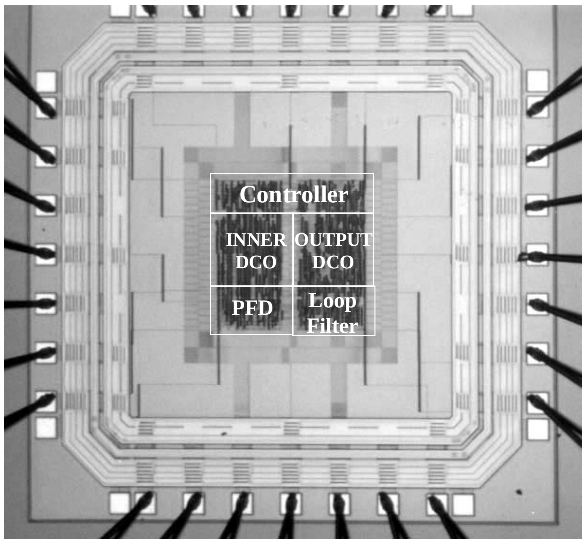


Fig. 3.8: Microphotograph of the ADPLL (TSMC 0.35um).

Fig. 3.8 shows microphotograph of the ADPLL chip. We use Auto Placement and Routing (APR) tools to generate the layout. Since different interconnection delays may result in mismatches between INNER DCO and OUTPUT DCO, we use APR tools to generate one DCO layout first and then duplicate this DCO layout in final APR process. And both DCO and PFD should have maximum occupied area constraints to minimize the wire-loading effects during APR. Gate count of the ADPLL is 4800. The core area of the ADPLL is 840μm x 840μm.

3.5 Experimental Results for the ADPLL test chip

Fig. 3.9 shows the transient response of the binary search ADPLL, where the reference clock is 5MHz, and the division ratio M is 40. Thus the output frequency is 200MHz (=5MHz*40). The code[11:0], which means {coarse[5:0], fine[5:0]}, is converged to a fine-tuning range of the INNER DCO in a short time. By using binary search step in frequency acquisition, the ADPLL can finish frequency acquisition in 46 (= $2*(2*\log_2(2^{12})-1)$) reference clock cycles.

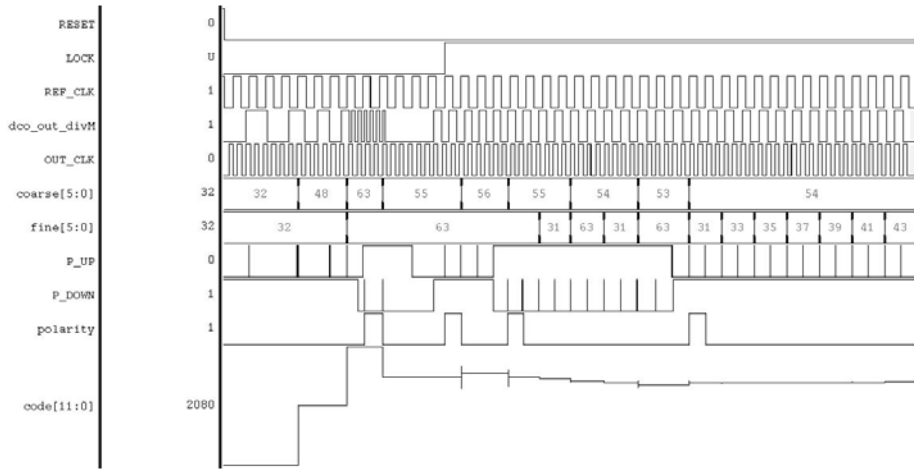


Fig. 3.9: Transient response of the binary search ADPLL (@200MHz).

Fig. 3.10 shows the transient response of TDC-based fast-locking ADPLL at 300MHz. In this figure, the reference clock is 10MHz and M=30, N=1. After system reset, the TDC calculates the nearest DCO control code and then outputs it to the ADPLL controller in the 1st reference clock cycle. In the 2nd reference clock cycle, the ADPLL controller updates the DCO control code to the initial DCO control code (TDC_DIV).

During the 1st to 3rd reference clock cycles, the DCO and the frequency divider are reset and waiting for the update of the DCO control code is completed. In the 4th reference clock cycle, the DCO starts to oscillate and the frequency divider also starts to work. With this reset control, there has no initial phase error between DCO's output and reference clock, and it can further reduce the time for phase acquisition. As a result, both frequency acquisition and phase acquisition is completed at the rising edge of 4th reference clock.

After 4th reference clock, the ADPLL takes the control signals from the PFD to fine-tune the output frequency and also keeps maintaining the phase of output clock.

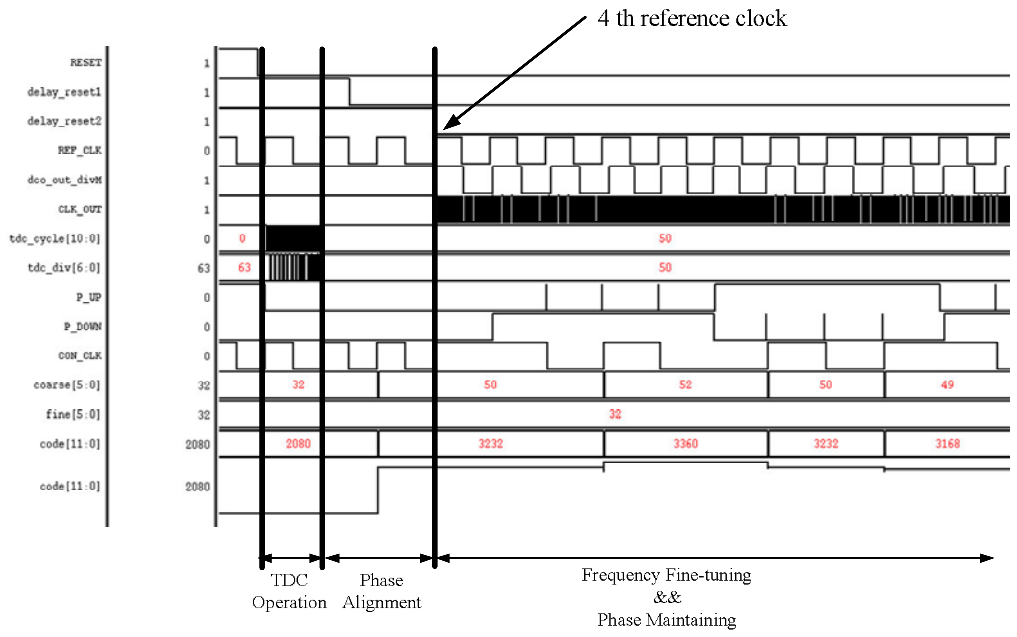


Fig. 3.10: Transient response of TDC-based fast-locking ADPLL (@300MHz).

Fig. 3.11 compares two types of the proposed ADPLL. One uses the TDC-based fast-locking scheme and the other use the binary search for frequency acquisition process. From Fig. 3.11, it shows that the DCO control code for the TDC-based ADPLL has a faster convergence rate than the binary search ADPLL. Thus the proposed TDC-based ADPLL is very suitable for fast lock-in time applications.

Fig. 3.12 shows the measured output waveform of the ADPLL with noisy digital circuitry (≈ 600mVpp supply noise) at 45MHz and 450MHz respectively. Due to the speed limitation of I/O PAD, the output frequency must be lowered for testing. The signal at Channel 2 shows the OUT_CLK signal divided by two and the signal at Channel D shows the long-term P_k-P_k jitter histogram over 200,000 sweeps. We use LeCory LC584A to measure output signal. Rms jitter and P_k-P_k jitter at 45MHz is 7ps and 20ps respectively. And rms jitter and P_k-P_k jitter at 450MHz is 22ps and 70ps respectively.

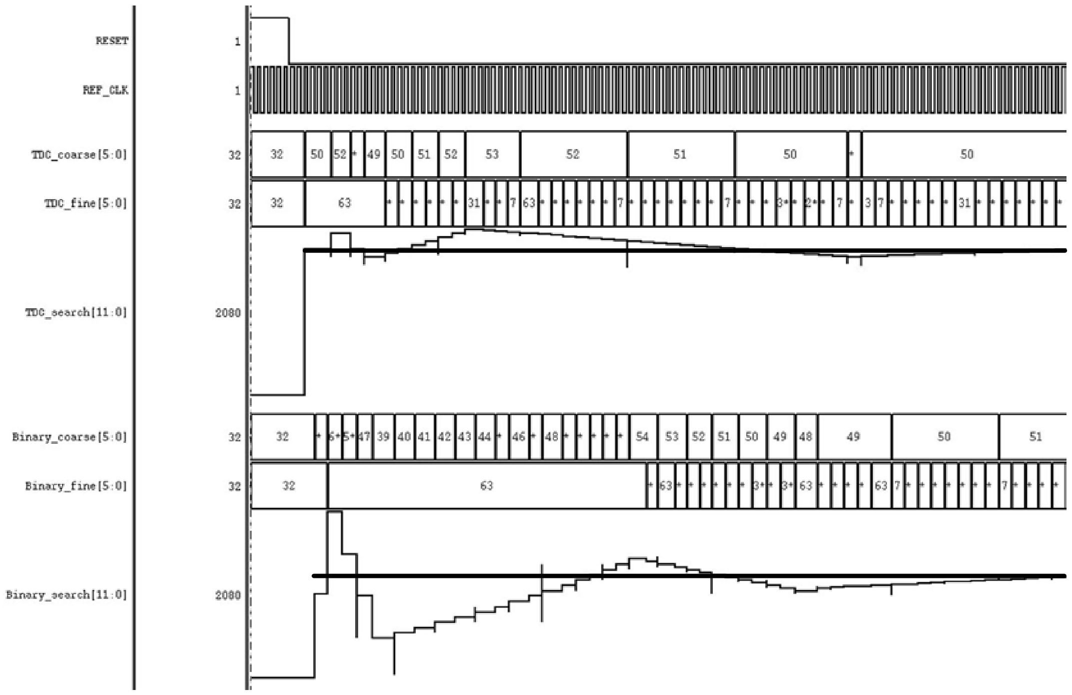
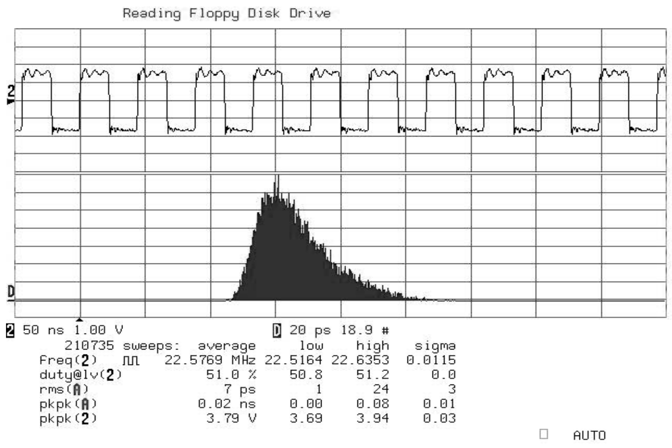


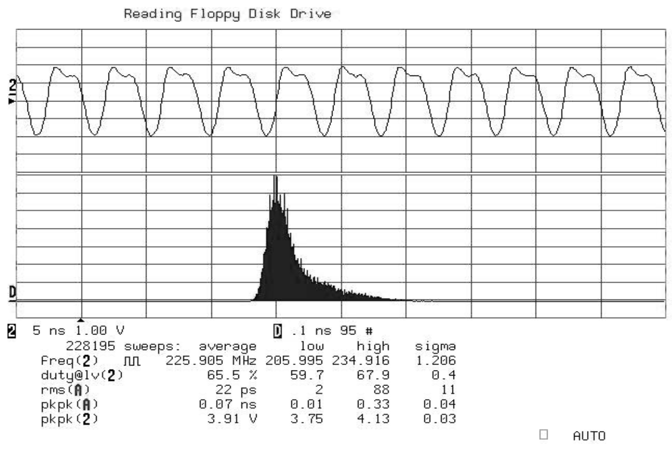
Fig. 3.11: Compared TDC-based ADPLL to binary search ADPLL (@300MHz).

Table 3.1 lists the comparisons among different PLLs. The proposed ADPLL has shorter lock-in time and better jitter performance than analog PLL [1] and ADPLL [9]. Although DCPLL [2] can achieve fast locking, its jitter performance is worse than the proposed design. And the proposed ADPLL also has smaller area and lower power consumption than the cell-based ADPLL [8]. The proposed binary search ADPLL can achieve fast locking in 46 reference clock cycles, and the proposed TDC-based ADPLL can achieve fast locking within 4 reference clock cycles, and the proposed architecture has the best portability than the other designs.



```
2 50 ns 1.00 V                  D 20 ps 18.9 #
     210735 sweeps:  average      low     high    sigma
   Freq(2)   ⎍⎍   22.5769 MHz 22.5164 22.6353  0.0115
   duty@lv(2)           51.0 %    50.8    51.2      0.0
   rms(A)                7 ps        1      24        3
   pkpk(A)            0.02 ns     0.00    0.08     0.01
   pkpk(2)             3.79 V     3.69    3.94     0.03
```

(a)



```
2  5 ns 1.00 V                  D .1 ns 95 #
     228195 sweeps:  average      low     high    sigma
   Freq(2)   ⎍⎍   225.905 MHz 205.995 234.916    1.206
   duty@lv(2)           65.5 %    59.7    67.9      0.4
   rms(A)               22 ps        2      88       11
   pkpk(A)            0.07 ns     0.01    0.33     0.04
   pkpk(2)             3.91 V     3.75    4.13     0.03
```

(b)

Fig. 3.12: Jitter histogram of the ADPLL (a) at 45MHz (b) at 450MHz.

Table 3.1: PLL PERFORMANCE COMPARISONS.

Performance Parameter	Proposed ADPLL	[1]	[2]	[8]	[9]
Process	0.35μm CMOS	0.25μm CMOS	0.60μm CMOS	0.6μm CMOS	0.5μm CMOS
Area	0.71mm^2	0.09mm^2	0.83mm^2	2.75mm^2	0.71 mm^2
Approach	All-Digital Cell-Based	Analog	Semi-Digital	All-Digital Cell-Based	All-Digital
Power dissipation	100mW (@500MHz)	25mW	105mW (@400MHz)	315mW @(800MHz)	39.6mW @(100MHz)
Max. Lock time	< 46 cycles	< 720 cycles	< 16 cycles	< 25 cycles	< 50 cycles
Min. Frequency	45MHz	8.5MHz	300MHz	360MHz	50MHz
Max. Frequency	510MHz	660MHz	800MHz	800MHz	550MHz
Supply voltage	3.3V	1.9V	3.3V	3.3V	3.3V
Output jitter (P_k-P_k)	70ps	80ps	149ps	60ps	125ps

3.6 Summary

In this chapter, an all-digital phase-locked loop is presented. The ADPLL can be implemented with standard cells. And it has good portability over different processes. The ADPLL implemented in a TSMC 0.35μm 1P4M CMOS standard cell library, can operate from 45MHz to 510MHz. The P_k-P_k jitter of output clock is less than 70ps, and the rms jitter of output clock is less than 22ps. A systematic way to design ADPLL is also presented in this chapter. The proposed ADPLL can reduce design time and circuit complexity. Therefore it is very suitable for SoC applications.

Chapter 4

All-Digital Delay-Locked Loop Design

In this chapter, the design for all-digital delay-locked loop (ADDLL) is presented. As the speed and the complexity of VLSI system increases rapidly, clock skew and clock jitter effects become more and more important now. Thus how to distribute the clock for large clock loading nets and minimize the clock skew among all modules has become a design challenge for high-speed integrated circuits.

Wide operation frequency for the clock deskew circuit is typically needed for many applications, such as I/O interface circuit and on-chip clock deskew circuit. And for portable or mobile applications, lock-in time is very important since the clock deskew circuit must support fast entry and exit from power management techniques. Thus how to design a fast-locking wide-range clock deskew circuit with low-jitter performance in a short period is the goal for this research.

Both Phase-Locked Loops (PLL's) and Delay-Locked Loops (DLL's) can be used to solve the clock skew problems in microprocessors and high-speed I/O interfaces. However, PLL accumulates phase error or clock jitter, and makes its jitter performance worse than DLL. Since DLL tracks the reference clock cycle by cycle and doesn't have this accumulative effect, it is a good alternative in clock deskew applications.

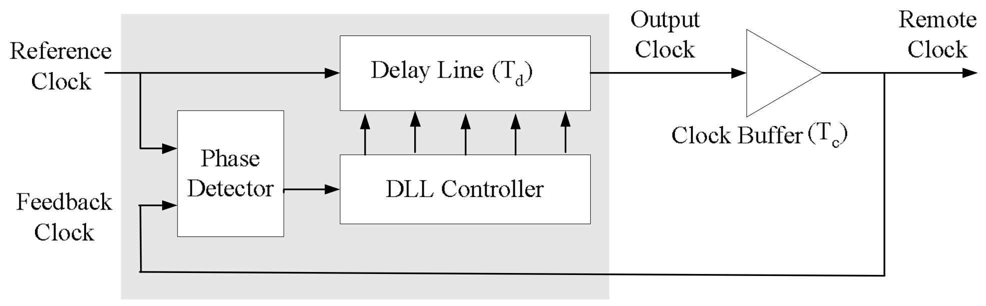


Fig. 4.1: Concept of Clock Deskew.

Fig. 4.1 shows the general architecture of conventional DLL's. The DLL adaptively inserts a delay between reference clock and remote clock. It selects an optimal delay (T_d) to compensate the phase error between reference clock and remote clock. After DLL is locked, both remote clock and feedback clock will synchronize with reference clock. Then clock buffer delay (T_c) can be ignored.

The lock condition for the DLL can be expressed as Eq. 4.1.

$$T_{loop}=T_{delay\text{-}line}+T_{clock\text{-}buffer} \qquad \text{(Eq. 4.1)}$$

where $T_{delay\text{-}line}$ and $T_{clock\text{-}buffer}$ denote the delay time of delay line and clock buffer respectively. T_{loop} means the total delay time between reference clock and remote clock (or feedback clock). After DLL is locked, T_{loop} becomes integer multiple of reference clock's period, thus there has no phase error between remote clock and reference clock.

It is difficult to design a DLL to overcome total effects caused by process, voltage, temperature, and loading (PVTL) variations. As in conventional DLL, the operation range of DLL is very limited. Thus different architectures have been proposed for different applications. It needs to trade off phase error, clock jitter, power consumption, area cost, portability, and lock-in time when designing a certain DLL.

In this chapter, we propose an all-digital fast-locking DLL. The proposed DLL utilizes

Time-to-Digital Converter (TDC) circuit and Digital-to-Time Converter (DTC) circuit to complete coarse-tuning in one clock cycle, resulting in less lock-in time. And the area cost for TDC and DTC can be shared with delay line. The proposed Digital-Controlled Delay Line (DCDL) architecture can turn off unused delay cells at high frequency operation, thus it is very suitable for wide-range clock deskew applications demanded in system-level integration.

This chapter is arranged as follows: section 4.1 the overview of DLL is discussed. Section 4.2 discusses the design trade-off between different DLL architectures. In section 4.3, the proposed fast-locking DLL architecture is presented and the performance of the proposed fast-locking algorithm is analyzed. In section 4.4, the implementation of the proposed DLL in a 0.35 μm 1P4M CMOS process with standard cells is presented. Section 4.5 shows simulation results of the proto-type chip and experimental results are also presented and discussed. Finally, a brief summary is made in section 4.6.

4.1 Overview of Delay-Locked Loop (DLL)

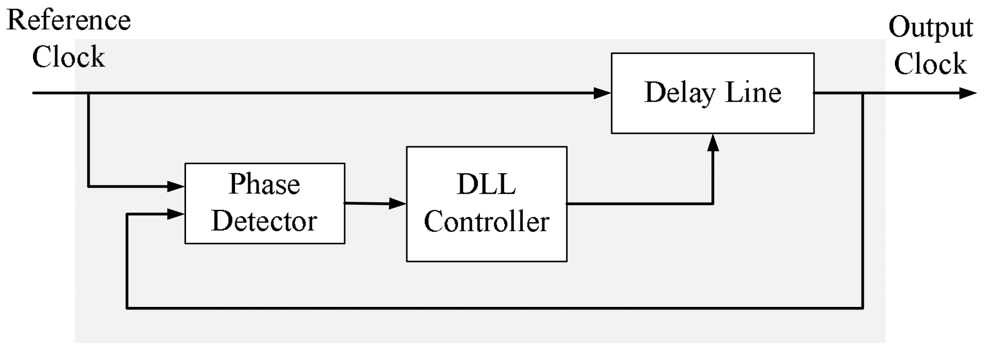


Fig. 4.2: The general DLL architecture.

The general DLL architecture is shown in Fig. 4.2. The input of DLL is reference clock, and output clock is a delayed version of reference clock outputs from the delay line. The DLL keeps tracking the phase of reference clock, and DLL controller increases or decreases the delay time of the delay line to minimize phase error between output clock and reference clock. After DLL is locked, the phase of output clock is synchronized with reference clock.

The DLL is often used to eliminate clock skew. In this application, the fixed delay exists at delay line output. And DLL eliminates the phase error between output clock and reference clock by adjusting the delay time of delay line. After DLL is locked, the phase of output clock is aligned with reference clock, and this fixed delay (i.e. clock skew) is removed.

DLL only needs to perform phase acquisition. Thus the lock-in time of DLL is dependent on how to quickly estimate the phase error and uses the delay line to compensate it. Thus the phase acquisition process is divided into coarse-tuning phase acquisition and fine-tuning phase acquisition. In the coarse-tuning phase acquisition, the synchronous mirror delay (SMD) type DLL [11,20,24] is often used to achieve fast lock-in time. And then in the fine-tuning phase acquisition, the phase detector detects the residual phase error and controls the delay time of delay line to eliminate the phase error.

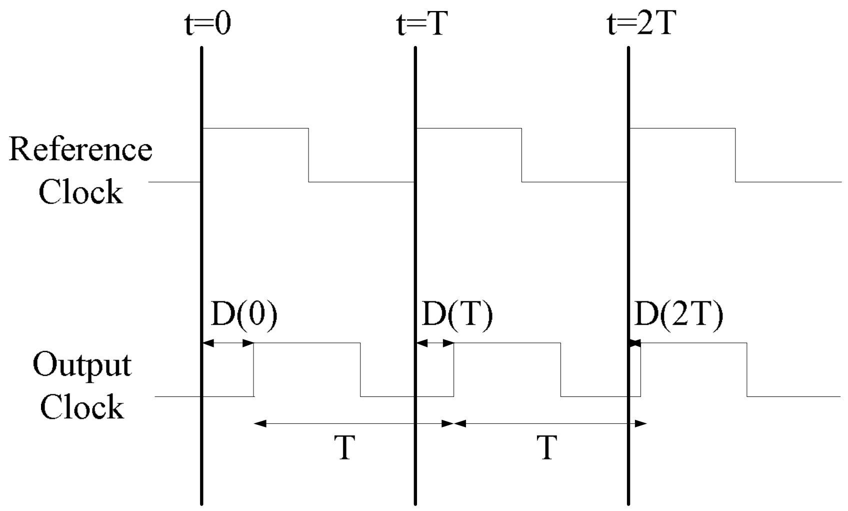


Fig. 4.3: The phase acquisition process of DLL.

Fig. 4.3 illustrates the phase acquisition process of DLL. Since output clock is a delayed version of reference clock, the period of reference clock (Reference Clock) and output clock (Output Clock) is the same. And since the delay line is open-loop architecture, the output clock doesn’t accumulate the previous phase error. Every rising edge of the output clock is a delayed version from current rising edge of the reference clock.

Thus if we express the phase error between Reference Clock and Output Clock as D(t), where t is time. The initial phase error is D(0). And we assume at time=0, the phase detector finds that Output Clock lags behind Reference Clock. The DLL controller will control the delay line to reduce the delay time. If the step size for phase acquisition is D_STEP, then the phase error at t=T becomes to D(T) = D(0) - D_STEP. And the phase error at t=2T is D(2T) = D(T) – P_STEP = D(0) – 2*D_STEP. The general form of phase error at time t can be expressed as Eq. 4.2.

$$D(t) = D(0) - n * D\_STEP \qquad \text{(Eq. 4.2)}$$

where we assume that the Output Clock still lags behind the Reference Clock after several updates of delay line’s delay time. When D(t) becomes negative or zero, the phase acquisition process is completed. And the DLL turns into phase maintaining mode. In this phase maintaining mode, the DLL keeps tracking the phase of reference clock by fine-tuning the delay line’s delay time.

Similarly to the PLL in phase acquisition process, the phase detector must provide correct phase relationship information about reference clock and output clock. Thus how to design a high sensitivity phase detector is also a design challenge for DLL design. And since the operating range and output phase accuracy is determined by the delay line, the design for high resolution delay line is also an important design challenge for DLL design.

4.2 Design Trade-Off in Different DLL Architectures

DLL architectures can be classified into three categories: analog DLL [12-15], digital DLL [11,16-19], and mixed-mode DLL [20-24]. Analog DLL often has smaller phase error than digital DLL. But when some fluctuations appear in voltage-controlled circuit, the total delay variation of whole delay line becomes very large and induces large jitter at output clock. To avoid this situation, the loop filter in analog DLL must have narrow bandwidth, leading to the fact that analog DLL often takes a long time to achieve lock.

Digital DLL has good portability over different fabrication processes, and it can achieve lock in a short time. But since the delay time is quantized, phase error and clock jitter will increase when continuously changing the quantized delay time with supply noise and reference clock's jitter. So the major design challenge of digital DLL is to improve the resolution of delay line while maintaining an acceptable short lock-in time and power consumption.

In mixed-mode DLL [20-24], it combines analog DLL and digital DLL and separates the locking scheme into coarse-tuning stage and fine-tuning stage. Coarse-tuning stage is adjusted by digital DLL. After digital DLL is locked, the phase selection of digital DLL remains unchanged, and analog DLL with phase interpolator will interpolate the fine-tuning stage's output and the coarse-tuning stage's output to produce the final output. Both fast locking and phase error minimization can be achieved by mixed-mode DLL. However, the portability of mixed-mode DLL is less than digital DLL since analog circuits depend on target process. And its design complexity is also higher than digital DLL.

For system-level integration or realization of System-On-Chip (SoC) design, it's better to implement the DLL with digital circuits because of less design complexity, higher portability, and lower supply voltage than analog circuit. However, large phase error and clock jitter are the major drawbacks of traditional digital DLL. Thus in SAR DLL [18], MOS capacitor is used to achieve high phase resolution in low supply voltage. And binary search algorithm is applied to reduce the lock-in time. But the delay line architecture proposed in [18] has large intrinsic delay, and when DLL operates at high frequency, most portions of delay line are not used. That means some unnecessary power dissipations are consumed at high frequency operation. So the DLL architecture proposed in [18] is not suitable for wide-range fast locking operation.

To reduce the intrinsic delay of delay line, the digital DLL [19] uses both reference clock and inversion reference clock to reduce half of delay line length. Moreover phase blenders are used to improve the phase resolution of delay line. But this DLL architecture needs extra duty cycle correctors to correct the duty cycle of reference clock and output clock to exactly 50%. Otherwise there will have large phase error between reference clock and output clock. Since the duty cycle corrector is an analog circuit, it will depend on process, so the portability of this DLL is less than the all digital DLL's proposed in [16-18].

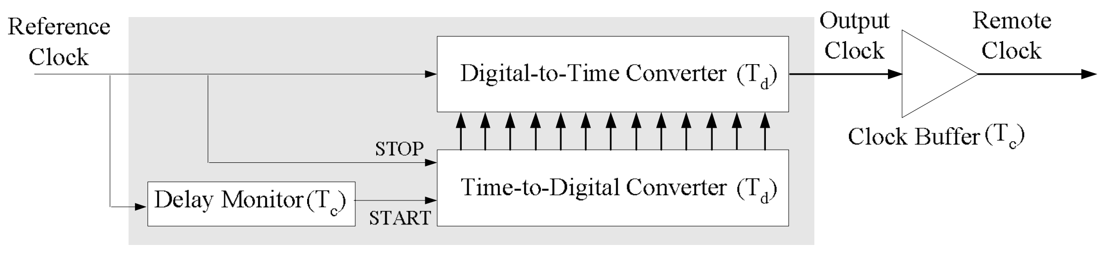


Fig. 4.4: Fast-Locking DLL based on TDC-DTC architecture.

To further reduce the lock-in time of DLL, TDC-DTC architecture is used [11,20, 24]. In Fig. 4.4, TDC and DTC are used to estimate the phase error quickly, and immediately compensate the phase error between reference clock and remote clock. However the resolution limitation of TDC and DTC makes it have large phase error after the DLL is locked. Thus it often needs to have further compensation to minimize the residue phase error. And the TDC circuit and the DTC circuit will increase the area cost for DLL design.

From the above discussions, a better ADDLL architecture should have a fast-locking time and also have small phase error and low power consumption for wide-range operation. Thus in the next section, an all-digital fast-locking DLL architecture is proposed. The proposed ADDLL architecture takes the advantages of the SAR DLL [18] and the TDC-DTC based DLL [11,20,24] to build up a low-jitter, low-cost, fast-locking and cell-based ADDLL.

4.3 The proposed ADDLL Architecture

Fig. 4.5 shows the proposed DLL architecture. The DLL consists of several parts, namely: phase detector, initial phase error estimator, Time-to-Digital Converter (TDC), DLL controller, and Digital Controlled Delay Line (DCDL). The DLL controller receives the UP and DOWN signals from the phase detector, and then it decreases or increases the delay time of DCDL respectively. The DLL needs to insert an optimal delay between reference clock and remote clock. After DLL is locked, the remote clock and feedback clock will synchronize with the reference clock.

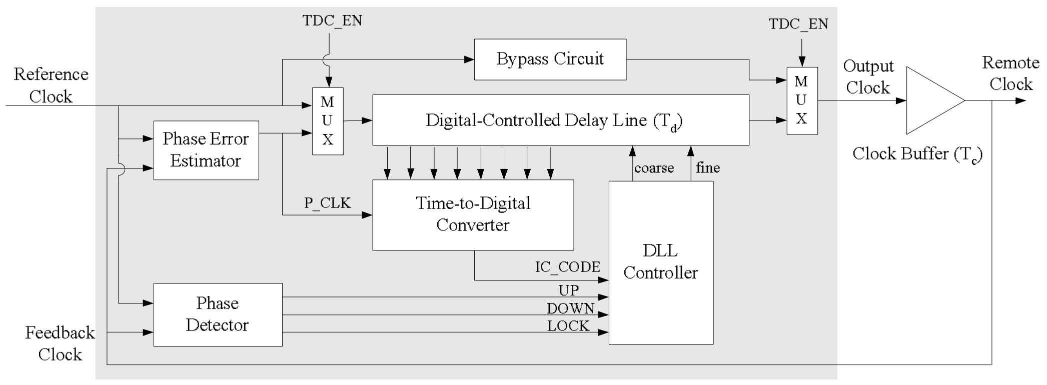


Fig. 4.5: The proposed fast-locking wide-range DLL.

To achieve fast locking, the phase error between reference clock and feedback clock should be compensated in a more efficient way. Fig. 4.6 shows the initial phase error between reference clock and feedback clock after system reset. The initial phase error estimator, which is also shown in Fig. 4.6, generates the “P_CLK” pulse for the next TDC stage. The pulse width of “P_CLK” indicates the initial phase error needed to be compensated.

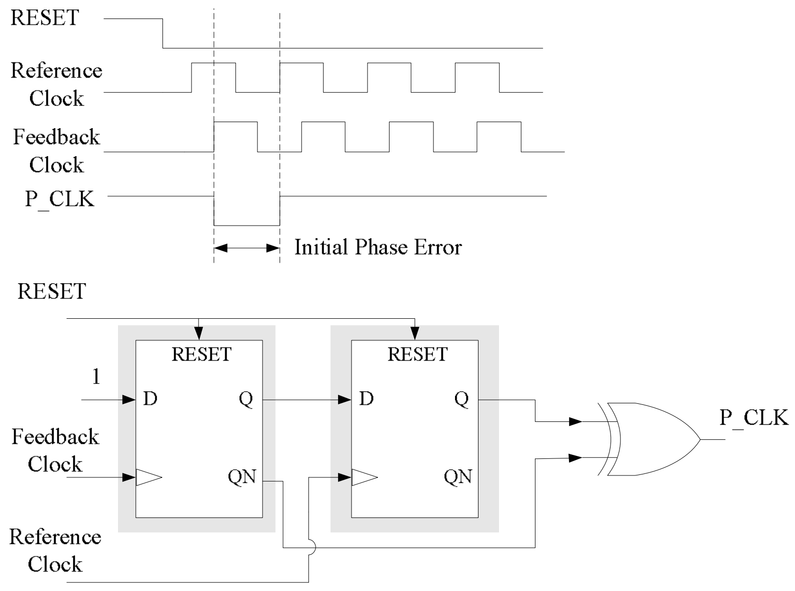


Fig. 4.6: Initial phase error estimator.

To speed up lock-in time, TDC is applied to the proposed DLL. Fig. 4.7 shows the architecture of the TDC module. In Fig. 4.7, “P_CLK” is set to high after system reset, thus the output of each coarse-tuning delay unit (CDU) will be initially set to high. Then when a low pulse is applied to “P_CLK”, this low signal will propagate through CDUs. And when the rising edge of “P_CLK” signal comes, implying the end of the pulse, the D-Flip/Flops will sample the current state of each CDU’s output. Thus the pulse width of “P_CLK” can be converted to multiples of CDU’s delay time. And the coarse-tuning control command for DCDL can be set to this initial value (IC_CODE). Then the feedback clock becomes very close to reference clock in a few clock cycles.

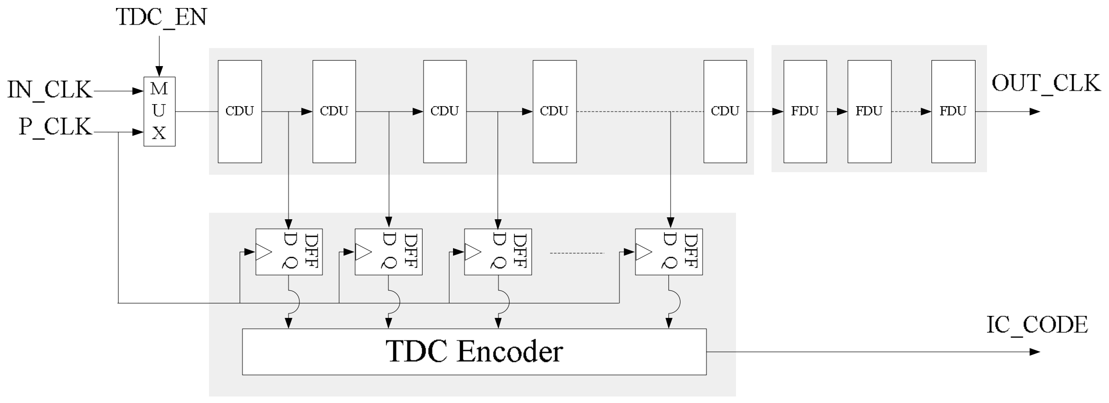


Fig. 4.7: The architecture of Time-to-Digital Converter for fast-locking ADDLL design.

The lock-in time for the proposed DLL can be expressed in term of reference clock cycles as T(n)=1+m+n, where m, n denote the delay line response time, and clock buffer delay time respectively. One extra clock cycle is needed for TDC calculation. Thus for the case, both delay line response time and clock buffer delay time are one clock cycle delay, t becomes 3 (=1+1+1) clock cycles. But since the resolution of TDC is limited by the delay time of CDU or the timing requirements of D-Flip/Flops, the lock-in time will be increased when further update of the delay line control command is needed.

For wide-range operation, the DCDL of DLL should have large controllable range to overcome the possible phase error in low frequency operation. But when DLL operates at high frequency, the DLL only needs a smaller controllable range than in low frequency operation. Thus the DCDL should have control mechanisms to turn off unused delay cells in high frequency operation to reduce power consumption.

In the proposed DLL, the DCDL is partitioned into coarse-tuning stage and fine-tuning stage. And its structure is shown in Fig. 4.8. In the coarse-tuning stage, totally (N-1) CDUs are used. The delay line controller will encode $\log_2(N)$ bits coarse-tuning control command into N paths selection control signals. Only one path will be selected corresponding to one specific coarse-tuning control command. And every four CDUs will have one CDU with disable control. Thus if the coarse-tuning control command is applied, those CDUs which are not used can be turned off to minimize power consumption. The DLL controller also controls those disable control signals.

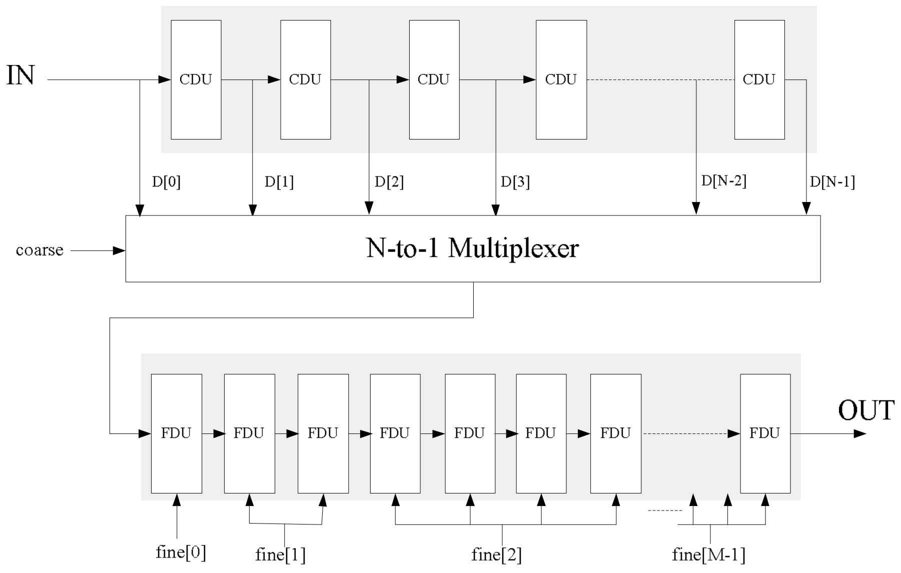


Fig. 4.8: The proposed digital-controlled delay line.

In the coarse-tuning stage, totally N different delays are achieved, and the step size of the coarse-tuning stage is determined by the delay time of CDU. Thus the totally delay controllable range of coarse-tuning stage is $T_{Coarse}=T_{CDU}*(N-1)$.

To increase the phase resolution of DCDL, fine-tuning stage is added after the coarse-tuning stage. The fine-tuning stage consists of (2^M-1) fine-tuning delay units (FDU), and this stage is controlled by (M) bits fine-tuning control command. Each FDU has two different delays: fast delay and slow delay. The delay time difference between fast delay and slow delay is denoted as T_{FDU}. The fine-tuning stage is binary-weighted controlled, leading to totally (2^M) different delays. The step size of fine-tuning stage is determined by T_{FDU}. Hence the total delay controllable range of fine-tuning stage becomes $T_{Fine}=T_{FDU}*(2^M-1)$.

It is important to keep the value of T_{Fine} larger than or equal to T_{CDU}. Otherwise when the coarse-tuning control command is changed, the output of the DCDL will have a large phase jump. And properly choosing the fine-tuning range can also minimize the possibility to change the coarse-tuning stage, and hence the possible glitches at path selector’s output can be avoided.

To overcome phase error at low frequency operation, the controllable range of DCDL should be larger than the maximum clock period of reference clock. That means T_{Coarse} should be larger than the maximum period of reference clock under PVTL variations.

The proposed DCDL architecture uses the path selector to minimize the intrinsic delay of delay line. The fine-tuning stage, which is connected after the output of coarse-tuning stage, can improve the phase resolution of the delay line and avoid possible glitches at path selector’s output.

In Fig. 4.5, multiplexers (MUXs) and bypass circuit are used to share the area cost for TDC circuit and DTC circuit with delay line. After system reset, the output clock is switched to the output of bypass circuit, and the input of delay line is switched to “P_CLK” signal. Thus TDC can use the CDUs of delay line to estimate the initial phase error between reference clock and feedback clock.

When the initial phase error has been estimated, the input of delay line is switched back to reference clock, and then the output clock is switched to the output of delay line. Thus the area cost of TDC circuit and DTC circuit can be reduced. In bypass circuit, it mirrors the fastest coarse-tuning stage delay and fine-tuning stage delay. Thus the measured initial coarse-tuning control command (IC_CODE) can be directly applied to the coarse-tuning stage.

4.4 The ADDLL Circuit Design

For high-speed I/O interface and on-chip clock de-skew applications, embedded DLL's are demanded to achieve high-speed data transfer rate. The required operation range of DLL may be different for different applications. To make sure that the proposed DLL can be applied to those applications, a test chip targeting to a wide operation range (65MHz to 500MHz) is designed.

The proto-type chip of the proposed DLL has been fabricated in a standard 0.35μm 1P4M CMOS process with standard cells. And we use Auto Placement and Routing (APR) tools to generate the layout of DLL. But some critical modules, such as DCDL or phase detector, should meet area constraints to minimize wire-loading effects during APR. Thus the proposed DLL can easily be ported to different processes in a short time with supported standard cells.

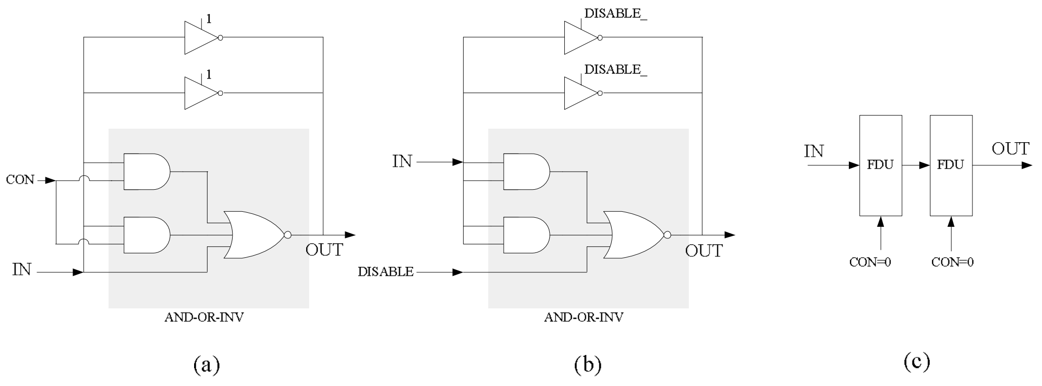


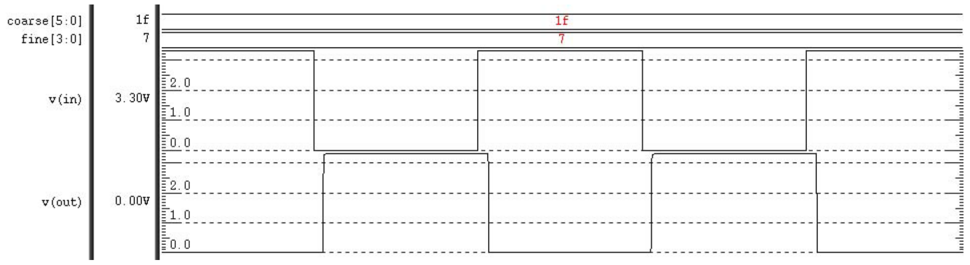
Fig. 4.9: (a) Fine-tuning delay cell (b) Fine-tuning delay cell with disable control (c) Coarse-tuning delay cell.

Fig. 4.9(a) shows the circuit of fine-tuning delay unit (FDU). The detail information about this fine-tuning delay cell is discussed in section 2.1. But it has some modifications for ADDLL applications. The FDU is constructed by one AND-OR-INV (AOI) cell and two shunted tri-state buffers which are always turned on. Tri-state buffers are used to reduce the transition time of FDU to improve operation frequency of the DCDL.

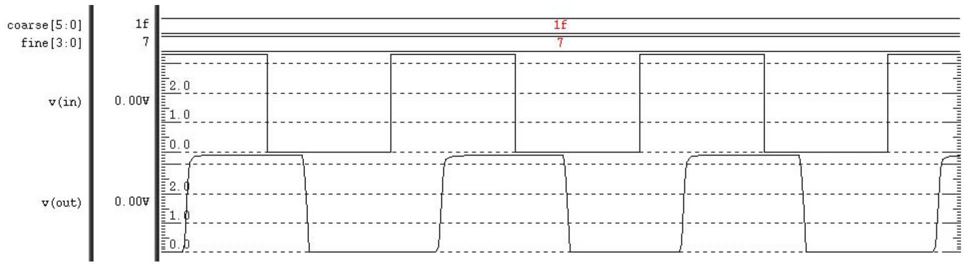
The FDU is an inverter with delay time control. The delay time from “IN” to “OUT” can be controlled by “CON”, where “CON”=1 is for fast delay and “CON”=0 is for slow delay. The delay time difference between those two cases is 20ps in the target process. Thus the step size of fine-tuning stage is 20ps (T_{FDU}).

Fig. 4.9(b) shows the FDU with disable control. If “DISABLE”=1, the “OUT” remains at low regardless of “IN”. This delay cell is used in coarse-tuning delay stage to turn off redundant delay cells after this delay cell. Thus the power consumption of DCDL in high frequency operation can be reduced. The input capacitance of this delay cell is the same as the FDU without disable control to prevent the delay time difference between normal delay cells and these delay cells.

The coarse-tuning delay unit (CDU) is constructed by cascading two FDUs and is shown in Fig. 4.9(c). The delay time from “IN” to “OUT” is 300ps, thus the step size of coarse-tuning stage is 300ps (T_{CDU}). Since the controllable range of fine-tuning stage should be larger or equal to the step size of coarse-tuning stage, totally 15 (=2^4-1) FDUs are used (M=4), and the total delay controllable range of fine-tuning stage is about 300ps (=20ps*(2^4-1)). The maximum period of reference clock is about 15.4ns (at 65MHz). Thus the total controllable range of DCDL (T_{Coarse}) should be larger than this value. Hence 63 CDUs are used (N=64), and the total controllable range of DCDL is about 18.9ns (=300ps*(64-1)).



(a)



(b)

Fig. 4.10: Simulation waveform of DCDL (a) at 50MHz clock input (b) at 500MHz clock input

Fig. 4.10 shows the simulation waveform of the proposed DCDL for different frequencies input. In the proposed DCDL, since each CDU consists of two FDUs, the output's rise time and fall time is balanced in coarse-tuning stage. And in fine-tuning stage, the asymmetry between output's rise time and fall time comes from the FDU, which is controlled by the LSB of fine-tuning control command. As a result the output's rise time and fall time of the proposed DCDL is balanced, and the duty cycle of output clock is almost 50% duty cycle.

In the ADDLL test chip, two phase detectors are used. One is coarse-tuning phase detector, and the other is fine-tuning phase detector, where the latter has higher sensitivity.

The sample-based phase detector is used as the coarse-tuning phase detector, and it is discussed in section 2.2. The circuit of the coarse-tuning phase detector is shown in Fig. 2.4.

When the “UP” signal is occurred, the delay time of the DCDL needs to be increased. Oppositely, when the “DOWN” signal is occurred, the delay time of the DCDL needs to be decreased.

The dead zone of coarse-tuning phase detector is limited by the timing requirements for the D-Flip/Flop. And in the target process, the sensitivity of the coarse-tuning phase detector is about ±300ps, which is not sufficiently precise for the DLL design. After the coarse-tuning phase detector is locked, the fine-tuning phase detector is turned on to further minimize the phase error.

The detail discussions for the fine-tuning phase detector were presented in section 2.2. And the schematic for the fine-tuning phase detector is shown in Fig. 2.5. The digital pulse amplifier is used to improve the sensitivity of fine-tuning phase detector. Thus the sensitivity of the fine-tuning phase detector can be improved to ±50ps by adding the digital pulse amplifies.

The DLL controller in the proposed DLL is described with Hardware Description Language (HDL) and then synthesized to the final gate-level circuit. The gate count of the proposed DLL is 5400.

4.5 Experimental Results for the ADDLL test chip

Fig. 4.11 shows the transition response of the proposed ADDLL. After system reset, the DLL starts to eliminate the phase error between reference clock (S_IN_CLK) and feedback clock (S_FB_CLK). The initial phase error estimator generates P_CLK pulse and sends this pulse to the TDC. The TDC will calculate the initial coase-tuning control code (IC_CODE) for the DCDL. The DLL controller takes the IC_CODE as start value and counts up or down the DCDL control code by the UP or DOWN signals from coarse-tuning or fine-tuning phase detectors.

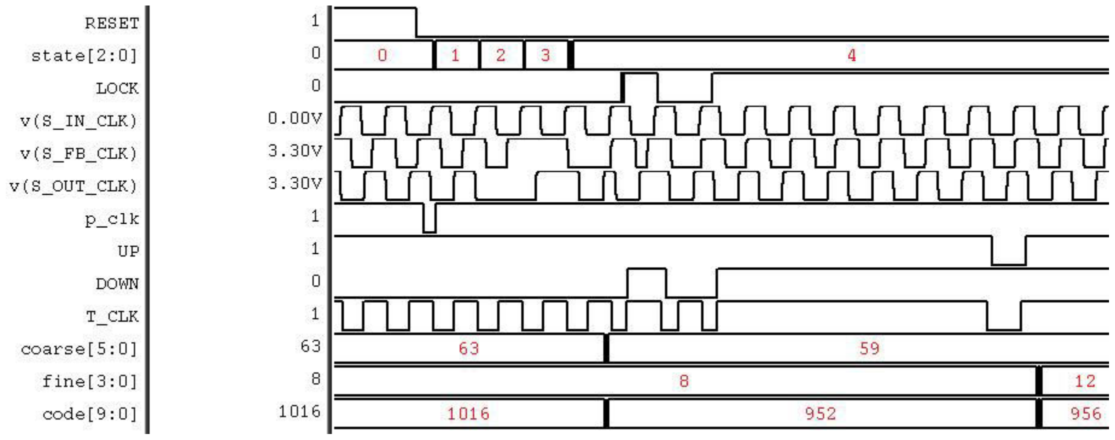


Fig. 4.11: Transient response of the proposed ADDLL.

The maximum operating speed of the proposed DLL is limited by the maximum operating speed of the phase detector and the DCDL. From chip measurement results, the proposed DLL can operate from 65MHz to 487MHz. And the power consumption of the proposed DLL is 210mW at 487MHz, and 50mW at 65MHz.

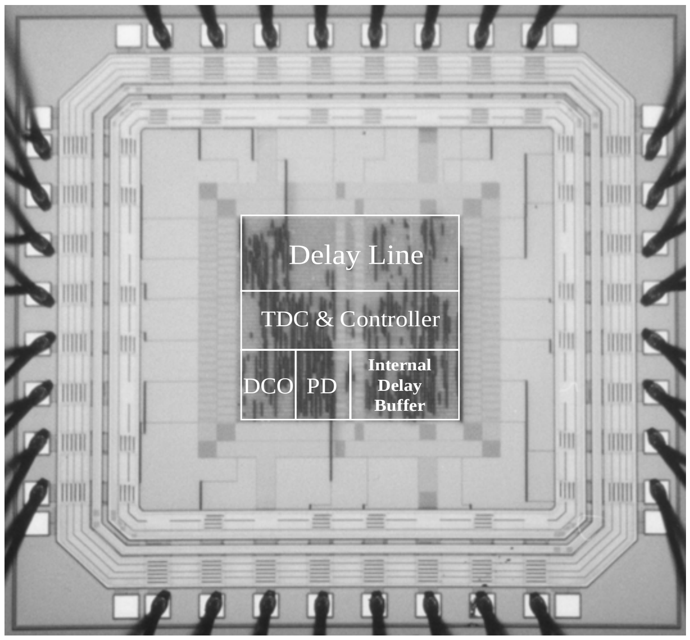
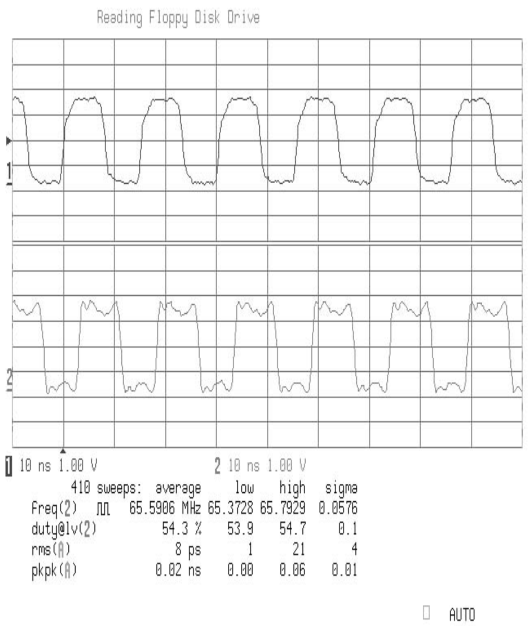


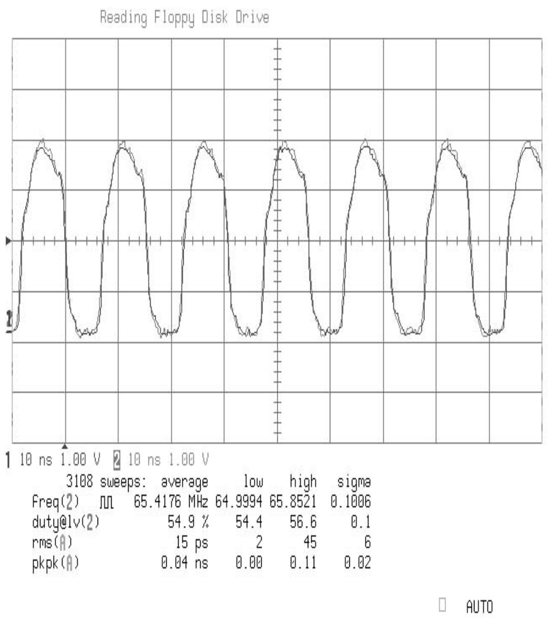
Fig. 4.12: Microphotograph of the DLL test chip.

Fig. 4.12 shows the microphotograph of the DLL test chip. The chip size is 2000μm x 2000μm, where its core size is 980μm x 980μm. Besides the DLL circuit, this chip also contains a digital-controlled oscillator (DCO) to generate the on-chip reference clock for the DLL. And this DCO can operate from 44MHz to 500MHz. An internal delay buffer is added to the test chip to insert different delays between output clock (OUT_CLK) and feedback (FB_CLK). The test chip can choose the reference clock from external reference clock or from the internal DCO. The clock buffer delay can be selected from external delay line or from internal delay buffer.



```
1 10 ns 1.00 V              2 10 ns 1.00 V
      410 sweeps:  average      low    high   sigma
  Freq(2)   ⎍⎍   65.5906 MHz 65.3728 65.7929 0.0576
  duty@lv(2)          54.3 %    53.9    54.7     0.1
  rms(A)                8 ps       1      21       4
  pkpk(A)            0.02 ns    0.00    0.06    0.01
```

(a)



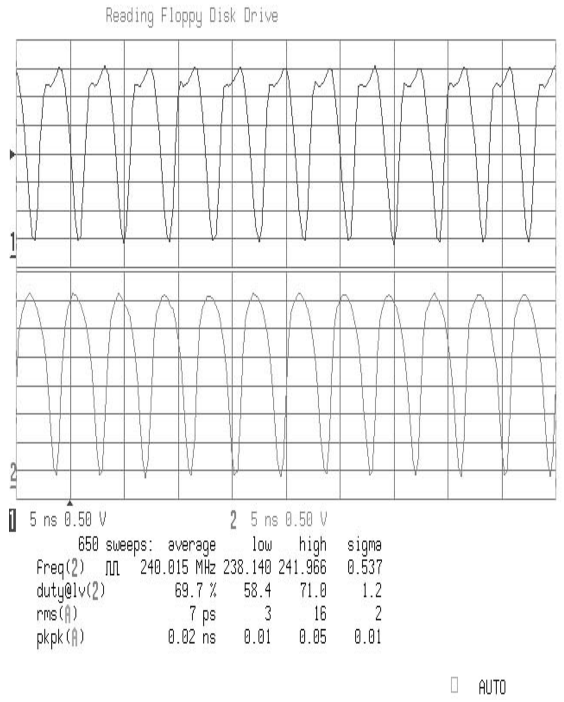
```
1 10 ns 1.00 V  2 10 ns 1.00 V
      3108 sweeps:  average      low    high   sigma
  Freq(2)   ⎍⎍   65.4176 MHz 64.9994 65.8521 0.1006
  duty@lv(2)          54.9 %    54.4    56.6     0.1
  rms(A)               15 ps       2      45       6
  pkpk(A)            0.04 ns    0.00    0.11    0.02
```

(b)

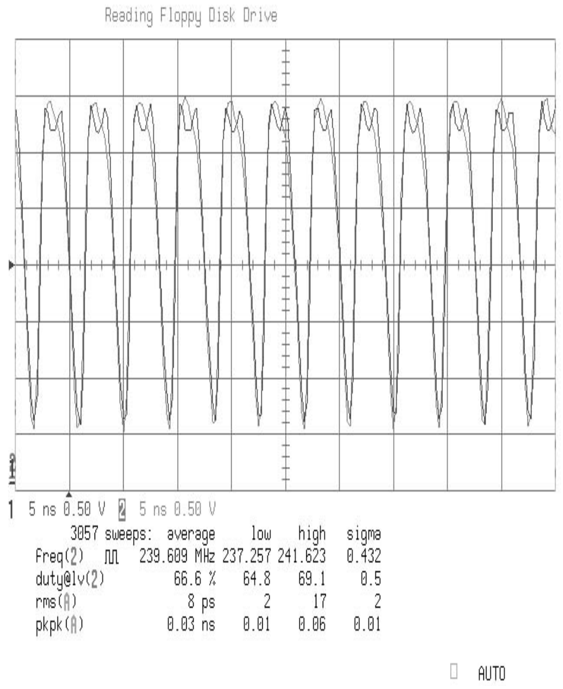
Fig. 4.13: Measured jitter at 65MHz. (a) DLL is reset (b) DLL is locked.

Due to the speed limitation of the I/O PADs, the output of DLL must be lowered for testing. Fig. 4.13 and Fig. 4.14 show the measured output waveform of the DLL at 65MHz and 487MHz respectively. In Fig. 4.13, signal at Channel 1 means the reference clock, and signal at Channel 2 means the feedback clock. In Fig. 4.14, signal at Channel 1 means the reference clock divided by 2, and signal at Channel 2 means the feedback clock divided by 2.

The rms jitter and peak-to-peak jitter of the DLL's output are measured by LeCroy LC584A. The rms and peak-to-peak jitter at 65MHz is 15ps and 40ps, at 487MHz is 8ps and 30ps respectively.



```
1  5 ns 0.50 V              2  5 ns 0.50 V
         650 sweeps:  average      low    high   sigma
  Freq(2)   ⎍⎍  240.015 MHz 238.140 241.966   0.537
  duty@lv(2)         69.7 %    58.4    71.0     1.2
  rms(A)              7 ps        3      16       2
  pkpk(A)          0.02 ns     0.01    0.05    0.01
```



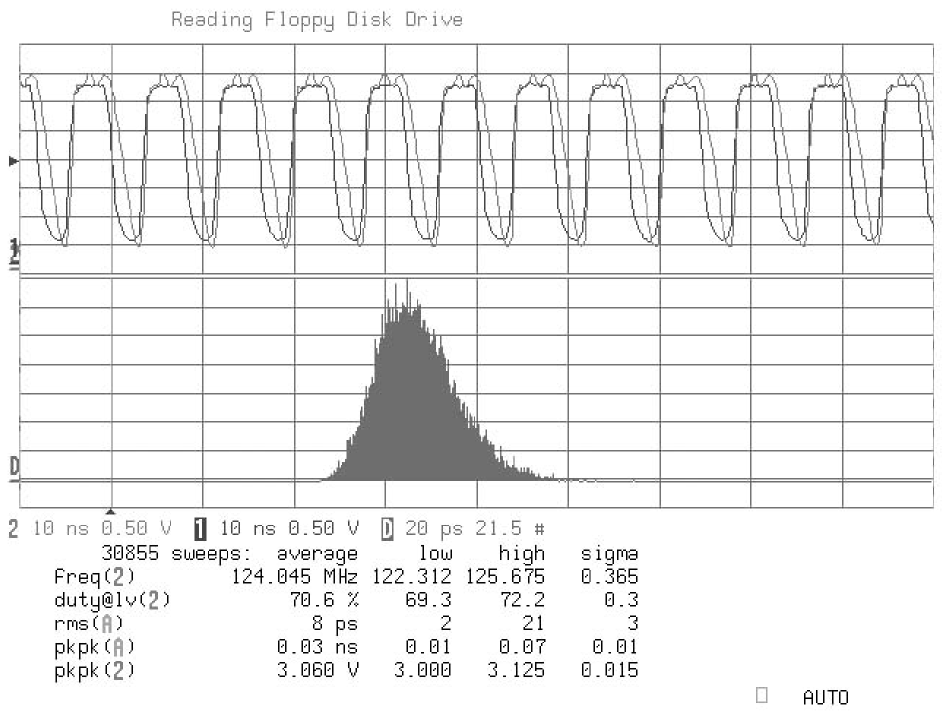
```
1  5 ns 0.50 V  2  5 ns 0.50 V
        3057 sweeps:  average      low    high   sigma
  Freq(2)   ⎍⎍  239.609 MHz 237.257 241.623   0.432
  duty@lv(2)         66.6 %    64.8    69.1     0.5
  rms(A)              8 ps        2      17       2
  pkpk(A)          0.03 ns     0.01    0.06    0.01
```

(a) (b)

Fig. 4.14: Measured jitter at 487MHz. (a) DLL is reset (b) DLL is locked.

Fig. 4.14 shows the measured DCDL's output divided by 2. This output signal should have almost 50% duty cycle. Due to the unbalanced rise time and fall time in I/O pads, the duty-cycle of measured signal does not approximate 50%. But the duty-cycle of the proposed DCDL's output is almost 50% as shown in Fig. 4.10.

Fig. 4.15 shows the measured long-term (over 370,260 clock cycles) jitter histogram of the proposed DLL. The measured long-term rms jitter and peak-to-peak jitter at 125MHz is 8ps and 30ps respectively.



```
2  10 ns 0.50 V   1  10 ns 0.50 V   D  20 ps 21.5 #
        30855 sweeps:  average      low     high    sigma
    Freq(2)         124.045 MHz 122.312 125.675   0.365
    duty@lv(2)          70.6 %     69.3    72.2     0.3
    rms(A)                 8 ps       2      21       3
    pkpk(A)             0.03 ns    0.01    0.07    0.01
    pkpk(2)            3.060 V    3.000   3.125   0.015
                                                              AUTO
```

Fig. 4.15: Measured long-term jitter histogram (at 125MHz).

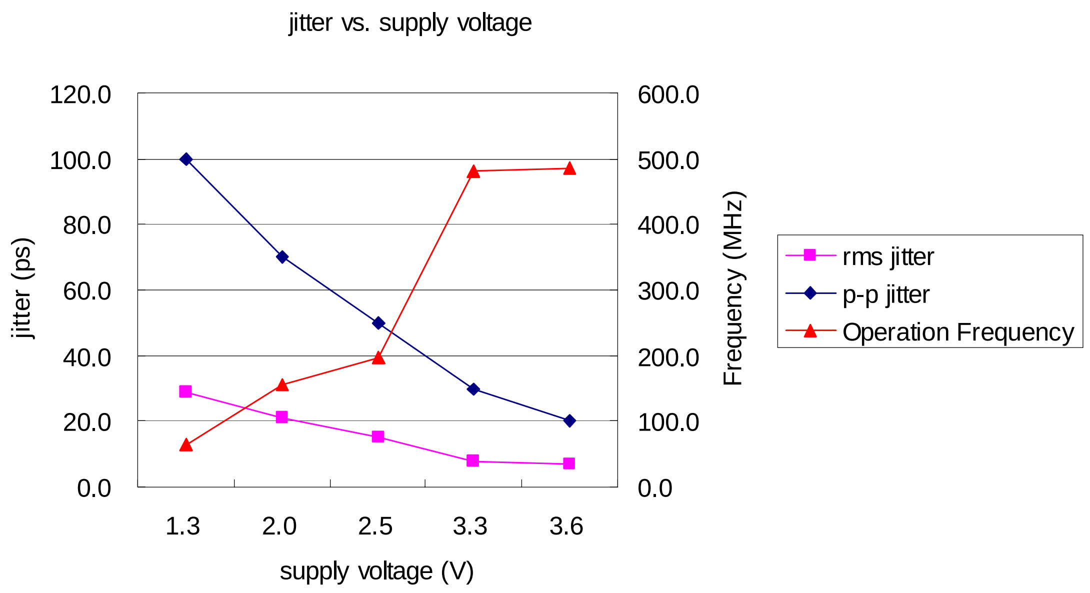


Fig. 4.16: Measured jitter and frequency vs. supply voltage.

Fig. 4.16 shows plots of measured jitter versus supply voltage. In low supply voltage, the resolution of the DCDL will decrease, and the sensitivity of the phase detector will also decrease. Thus both rms jitter and peak-to-peak jitter become worse than high supply voltage. The proposed DLL can still work at 65MHz with 1.3V supply. That means the proposed DLL can operate with low supply voltage.

Table 4.1: PERFORMANCE SUMMARY OF THE PROPOSED DLL

Technology	0.35μm SPQM CMOS

Power Consumption	50mW(@65MHz)
	210mW(@487MHz)

Max. rms jitter	16ps
Max. p-p jitter	40ps
Max. lock-in time	< 8 cycles
Chip Core Area	980x980μm^2

Table 4.1 summarizes the performance of the proposed DLL test chip. And table 4.2 lists the comparisons among different DLLs. The proposed DLL can achieve fast locking in a few clock cycles, and its jitter performance is also better than those described in [18-20]. And it has the best portability than the other designs.

Table 4.2: DLL PERFORMANCE COMPARISONS.

Performance Parameter	Proposed DLL	[17]	[18]	[19]	[20]
Process	0.35μm CMOS	0.35μm CMOS	0.25μm CMOS	0.4μm CMOS	0.4μm CMOS
DLL Power	210mW (@487MHz)	3.2mW (@100MHz)	3.3mW (@100MHz)	340mW (@400MHz)	18mW (@250MHz)
Phase resolution	20ps	200ps	160ps	40ps	~ 0ps
Max. Lock time	1+m+n cycles (m: clock buffer delay n: delay line delay)	< 5μsec	< 30 cycles	< 2.9μsec	< 10 cycles
Min. Frequency	65MHz	100MHz	90MHz	250MHz	150MHz
Max. Frequency	487MHz	100MHz	100MHz	500MHz	350MHz
Supply voltage	3.3V	2.0V	1.1V	3.3V	3.0V
Output jitter (p-p)	< 40ps	~ 0ps	95ps	< 250ps	<150ps

4.6 Summary

In this chapter, an all-digital fast-locking DLL for wide-range clock deskew applications is presented. The proposed DLL utilizes TDC circuit to complete coarse-tuning in one clock cycle. Moreover the proposed DCDL is extendable, and its phase resolution is determined by fine-tuning delay cells. Both fast-locking and minimum phase error can be achieved by the proposed DLL. The DLL can be implemented by standard cells, and hence it can be ported to different processes in minimum design cycle. The DLL test chip fabricated in the TSMC 0.35μm CMOS process can achieve a phase resolution better than 20ps, and the operation range of the test DLL chip ranges from 65MHz to 487MHz. The maximum rms jitter and maximum peak-to-peak jitter are less than 16ps and 40ps respectively with a 3.3V supply. For these reasons, the proposed DLL is very suitable for wide-range clock deskew applications demanded in system-level integration.

Chapter 5

All-Digital Multi-Phase Clock Generator Design

In this chapter, the design for all-digital multi-phase clock generator (ADMCG) is presented. Multi-phase clocks are useful in many applications. In high-speed serial link applications [29,30,34], multi-phase clocks are used to process data streams at the bit rate higher than internal clock frequencies. In clock multiplier applications [25,28,33], multi-phase clocks are combined to produce the desire output frequency for the synthesizer. And in microprocessors, multi-phase clocks can ease the clock constraints in pre-charged logic to achieve higher operating speed [31]. In wireless LAN baseband design, the multi-phase clocks can be used to find a better sampling point for the analog-to-digital converter (ADC) to improve overall system performance.

Both Phase-Locked Loops (PLL's) [34] and Delay-Locked Loop (DLL's) can be employed for multi-phase clocks generation. But DLL offers better jitter performance than PLL because the noise induced by power supply or substrate noise disappears at the end of the delay line. Oppositely, the ring oscillator of PLL accumulates jitter, and any uncertainty in an earlier transition affects all the following transitions, and its effect persists indefinitely [12,27,30,32]. Thus DLL is a good alternative for PLL's in multi-phase clocks generation applications.

But there are two major drawbacks of conventional DLLs. One is their limited phase capture range [12], and the other is restricted Voltage-Controlled Delay Line (VCDL) range to avoid false-lock to the harmonics [27,28]. By increasing the VCDL delay range and changing the phase alignment algorithm, it can be extended to infinite phase capture range. But the false-lock problem still cannot be overcome. Thus in [27,28], a self-correcting circuit is employed to prevent the DLL locks to an incorrect delay and it can bring the DLL back into a correct locked-state. However, this self-correcting circuit [27] is sensitive to the duty cycle of reference clock since it makes decisions based on the sampling values of multi-phase clock signals.

The register-controlled digital DLL is proposed in [36] to provide an all-digital solution for the DLL design. For multi-phase clock generation applications, this DLL can overcome the false-lock problem by setting the delay line in minimum delay time at the beginning of phase acquisition. However, the long lock-in time makes it not suitable for wide-range operations.

In this chapter, a new DLL-based approach for multi-phase clock generation is presented. The proposed All-Digital Multi-Phase Clock Generator (ADMCG) uses a Time-to-Digital Converter (TDC) to choose a reasonable delay range rather than to use self-correcting circuit. Thus its operation is very robust and can avoid possible false-lock as in conventional designs. The lock-in time of the proposed ADMCG can also be reduced by adding TDC module. After TDC operation, a fixed step search scheme is used in the ADMCG to fine-tune the output phase accuracy. The proposed architecture is all-digital and can be realized by standard cells. Thus it yields good testability, programmability, stability and portability over different processes. And the design time for multi-phase clock generator can also be reduced.

This chapter is arranged as follows: Section 5.1 describes the proposed all-digital multi-phase clock generator. Section 5.2 shows the implementation of the proposed ADMCG using standard cells and the test chip design for a 7:1 data channel compression transceiver.

Simulation and chip measurement results of the ADMCG test chip are shown in section 5.3. Finally, a brief summary is made in section 5.4.

5.1 The proposed ADMCG architecture

The proposed ADMCG architecture for multi-phase clock generation is shown in Fig. 5.1. The ADMCG consists of four major modules namely: Phase Detector (PD), Time-to-Digital Converter (TDC), Digital-Controlled Delay Line (DCDL), and ADMCG controller.

The DCDL is divided into K equal-delay stages, and all delay stages are controlled by the same control code. The TDC estimates the period of reference clock and passes it to the ADMCG controller for selecting the suitable delay range of the DCDL.

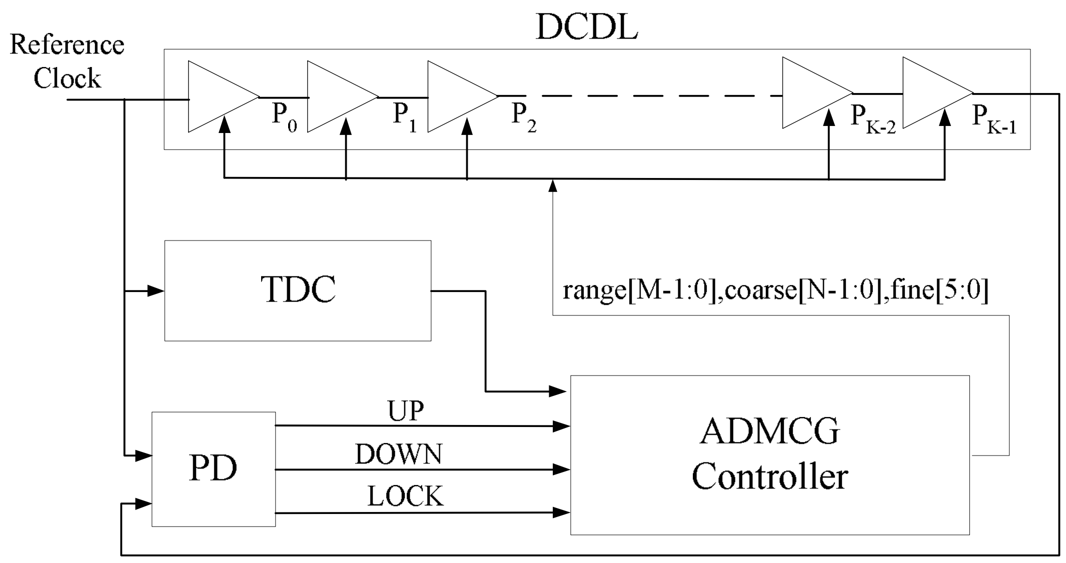


Fig. 5.1: The proposed ADMCG architecture.

The PD detects the phase error between reference clock and the delay line output (P_{K-1}). It generates UP and DOWN signal to indicate that the ADMCG controller should decrease or increase the delay time of the DCDL respectively. When phase error between reference clock and P_{K-1} is less than the dead zone of PD, the LOCK signal is asserted and then multi-phase clock signals: $P_0 – P_{K-1}$ are generated.

The delay range problem of conventional DLL is discussed in [12,27,28]. The reason that the DLL may lock to multiples of reference clock's period is because only the phase of delay line output and reference clock is compared. Thus when the delay line has a wide controllable range, the unpredictable initial delay time of delay line and unknown relationship among delay line output and reference clock may result in locking to multiples of reference clock's period, and hence multi-phase clock generation becomes fail.

Since wrong operating delay range for the delay line and lacking of information for reference clock's period is the reason that caused false-lock, how to dynamically adjust the delay line's operating range to a suitable range is the challenge for multi-phase clock generator design.

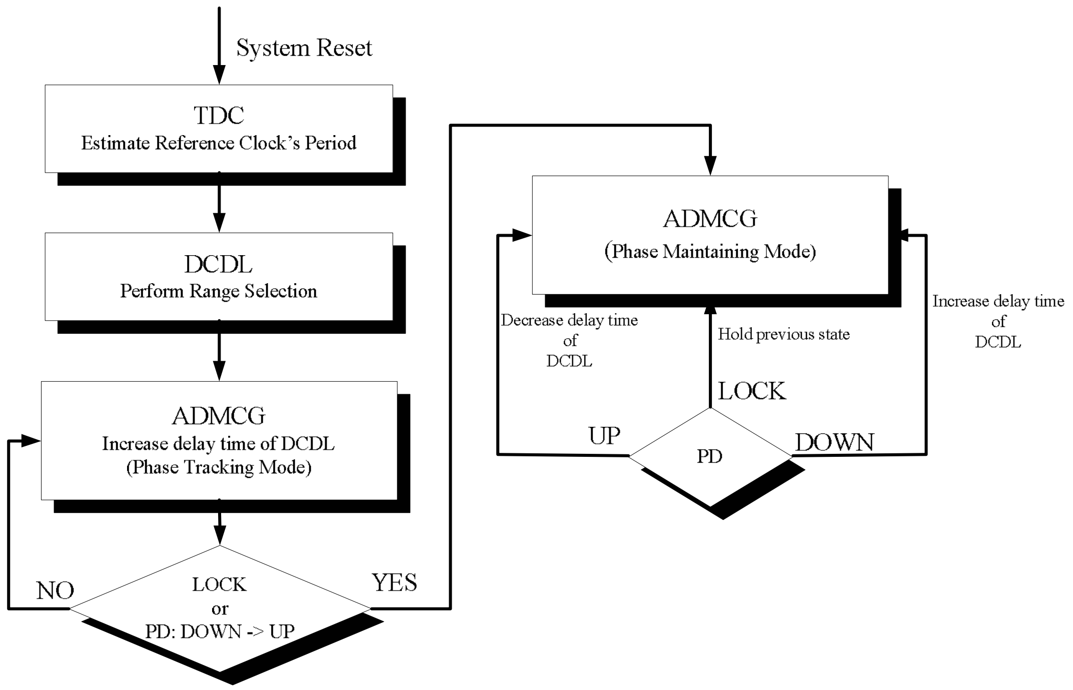


Fig. 5.2: The proposed ADMCG control algorithm.

Fig. 5.2 describes the proposed ADMCG control algorithm. As discussed in [12,27,28], to avoid false-lock, the DCDL should always operate under this delay range: $0.5*T_{REF} < T_{DCDL} < 1.5*T_{REF}$, where T_{REF} means the period of reference clock and T_{DCDL} means the delay time of the delay line.

In the proposed ADMCG architecture, the TDC shown in Fig. 5.3 converts the reference clock's period information (T_{REF}) into multiples of Range Delay Unit's (RDU's) delay time. After TDC encoder, the DCDL range selection control code (range[M-1:0]) is sent to the ADMCG controller. Then it makes the DCDL firstly operate in this delay range: $0.5*T_{REF} < T_{DCDL} < T_{REF}$.

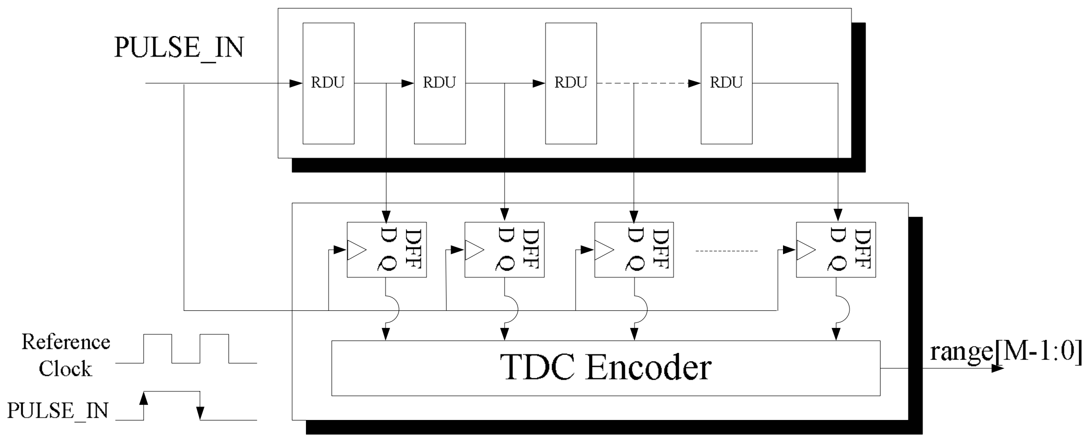


Fig. 5.3: The proposed Time-to-Digital Converter (TDC) for multi-clock generator.

After TDC operation, the ADMCG controller enters phase tracking mode, and it increases the delay time of the DCDL until the residual phase error between reference clock and P_{K-1} is disappeared and the PD's output changes from DOWN to UP (or LOCK is asserted). Then the ADMCG controller turns into phase maintaining mode, and it decreases or increases the delay time of the DCDL according to the PD's UP/DOWN signal respectively.

To speedup the lock-in time, in phase tracking mode, the phase search step is set to half of one coarse-tuning delay time. But after the ADMCG controller enters phase maintaining mode, the phase search step is reduced to one fine-tuning step.

Since the proposed ADMCG is not dependent on the relationship among multi-phase clock signals and it doesn't need to setup a start-up control to avoid the false-lock, the proposed design is very robust to Process variations, Voltage variations, and Temperature variations (PVT variations). Moreover, it is insensitive to the duty-cycle of the reference clock since only the rising edge of reference clock is used.

The output phase accuracy of the generated multi-phase clock signals is dependent on the phase resolution of the DCDL and the dead zone of the PD. And the operating frequency range of the proposed ADMCG is limited by the minimal delay time of the DCDL and the controllable range of each delay stage.

The proposed DCDL consists of K equal delay stages, and the architecture for one delay stage is shown in Fig. 5.4. The delay time of one delay stage is controlled by three cascading stages: range selection stage, coarse-tuning stage, and fine-tuning stage. And they are controlled by the range selection control code (range[M-1:0]), coarse-tuning control code (coarse[N-1:0]), and fine-tuning control code (fine[5:0]) respectively.

Range selection stage and coarse-tuning stage are implemented using the path selector. The difference between those two stages is that the RDU has larger delay than the coarse-tuning delay unit (CDU). The (M,N) parameters are used to adjust the operating range of the path selector by changing the number of selectable paths in the path selector. And to improve the phase resolution, the fine-tuning delay cell, which is discussed in section 2.1, is added after the coarse-tuning stage. In fine-tuning delay cell, it uses the six control bits: (EN1 A1 B1 EN2 A2 B2) to alter the delay time finely.

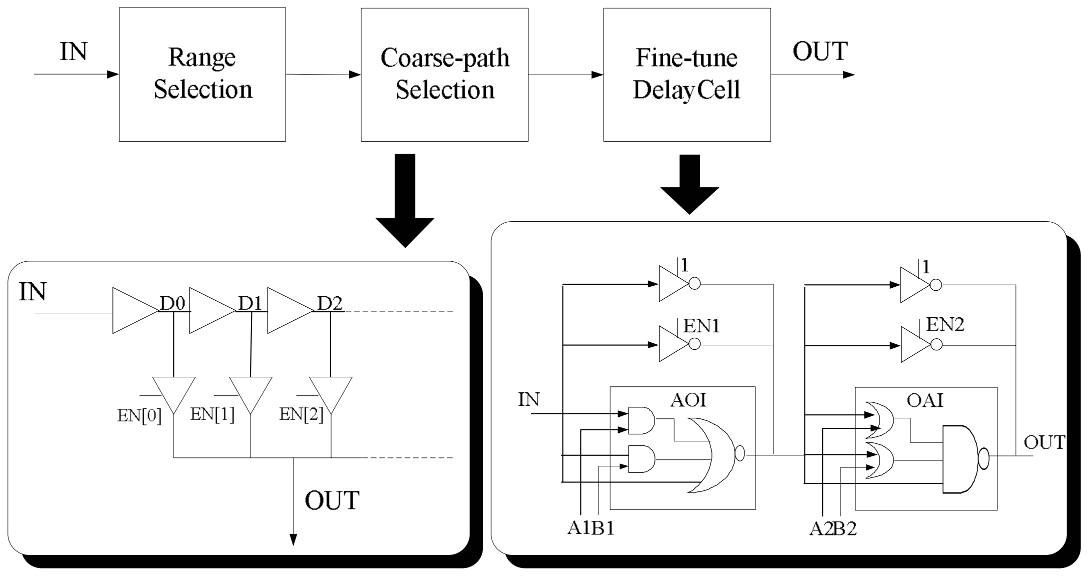


Fig. 5.4: The architecture of one delay stage.

The proposed TDC architecture is shown in Fig. 5.3. In Fig. 5.3, all RDUs are clear to low after system reset, and in the first reference clock cycle, the TDC's input (PULSE_IN) persists at high. This high signal will propagate through RDUs. And when the falling edge of PULSE_IN signal arrives, implying the end of the pulse, the D-Flip/Flops will sample the current state of each RDU's output. After the TDC encoder, the reference clock's period information (T_{REF}) can be converted into multiples of RDU's delay time. And the ADMCG controller uses this information to select a certain range for the DCDL.

The proposed high sensitivity PFD is used in the ADMCG design. The detail information about the high sensitivity PFD design is discussed in section 2.2. After using the digital amplifier in PD design, the dead zone of PD can be reduced to 50ps in the target process.

The ADMCG controller is described using Hardware Description Language (HDL) and then is synthesized by logic synthesizer. And all function blocks in the proposed ADMCG are cell-based design. Thus the proposed design can be easily ported to different processes with cell library support. And it can also reduce the design time and design complexity for multi-phase clock generator design.

5.2 ADMCG Test Chip Design

The ADMCG test chip is fabricated on a standard 0.35μm 1P4M CMOS process. To reduce area and power consumption of the DCDL, the RDU is implemented with delay cells provided in cell library. In those delay cells, the MOS channel length is longer than normal cells. Therefore they have an extremely large delay than normal cells. The delay time of one RDU is 1.6ns (T_{RDU}) in the target process. And the delay time of coarse-tuning delay cell is 0.16ns (T_{CDU}). After adding the fine-tuning delay cell, the phase resolution of each delay stage can be improved to 3ps on the average, and the total controllable range of the fine-tuning delay cell is 0.174ns (T_{FINE}).

To avoid large phase jump when path selection of coarse-tuning stage is changed, the value of T_{FINE} must keep larger than or equal to T_{CDU}. And the total controllable range of coarse-tuning stage also needs to be larger than T_{RDU}. Thus a 16-to-1 path selector is used in the coarse-tuning stage (i.e. $(16\text{-}1)*T_{CDU} > T_{RDU}$). After carefully selecting the delay cells of delay line, the jitter effect caused by the path selector can be minimized and the possibility to change the path selection can be reduced too.

In the test chip, the proposed ADMCG is applied to design a 7:1 data channel compression transceiver. The architecture of the transceiver is shown in Fig. 5.5. From design specifications, the reference clock period (T_{REF}) ranges from 50ns (20MHz) to 11.765ns (85MHz), and a 7-phase multi-phase clock generator is needed in the transceiver design. Thus a 4-to-1 path selector is used in range selection stage to provide a maximal DCDL delay time: 50.4ns (= $(7*(4\text{-}1)*T_{RDU} + 7*(16\text{-}1)*T_{CDU})$) larger than T_{REF}.

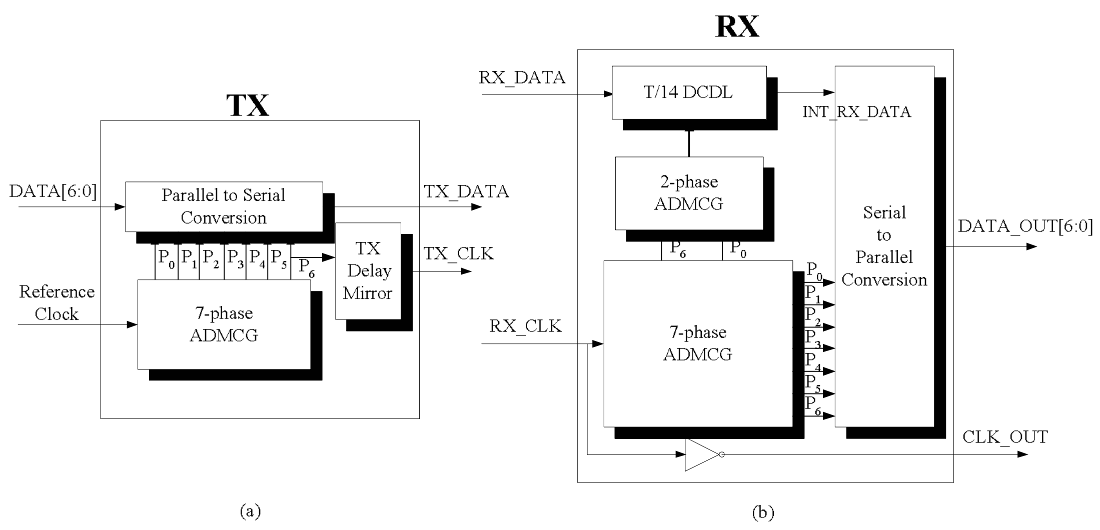


Fig. 5.5: The proposed 7:1 data channel compression transceiver (a) Transmitter (b) Receiver

The transmitter (TX) and the receiver (RX) are fabricated in the same test chip. And the transmitter’s outputs: TX_DATA and TX_CLK are sent to the receiver’s inputs: RX_DATA and RX_CLK respectively. In the transmitter, the generated 7-phase clock signals are used to transfer 7-bit data (DATA[6:0]) into one data channel (TX_DATA), and the transmitted data’s reference clock (TX_CLK) is also sent to the receiver. The “TX delay mirror” shown in Fig. 5.5(a) is used to compensate the delay time of parallel-to-serial converter.

The receiver shown in Fig. 5.5(b) recovers received data stream (RX_DATA) back to original 7-bit data (DATA_OUT[6:0]). The 2-phase ADMCG shown in Fig. 5.5(b) is used to estimate the accurate delay of $T_{REF}/14$. It aligns two adjacent phases of the 7-phase ADMCG’s outputs (i.e. P_6 and P_0) to measure the $T_{REF}/14$ delay, and the received data stream will firstly be delayed by $T_{REF}/14$ and then sampled by 7-phase multi-phase clock signals. Thus those multi-phase clock signals can sample the received data stream in the center of bit symbol boundary, and this maximizes the timing margin of the receiver circuit.

Since the RX_CLK may not have 50% duty cycle, the inverse of multi-phase clock signals cannot be directly applied to sample the received data stream. Thus to make a robust receiver, the 2-phase ADMCG is necessary for the proposed receiver circuit design.

5.3 Experimental Results for the ADMCG Test Chip

Fig. 5.6 shows the post-layout simulation waveform of the proposed ADMCG. To make sure that the proposed design would not cause a failure with a noisy reference clock, an 85MHz noisy reference clock (P_k-P_k jitter: ±500ps) is used in this simulation. After system reset (i.e. PDWN=1), the TDC measures the period of the reference clock, and makes the DCDL operate in a suitable delay range (i.e. $0.5*T_{REF} < T_{DCDL} < T_{REF}$). Then the ADMCG controller continues fine-tuning the output phase accuracy with PD's UP/DOWN signal. And when the phase error between delay line's output (PHASE[6]) and reference clock (CLK_IN) is minimized, the multi-phase clock generation is completed.

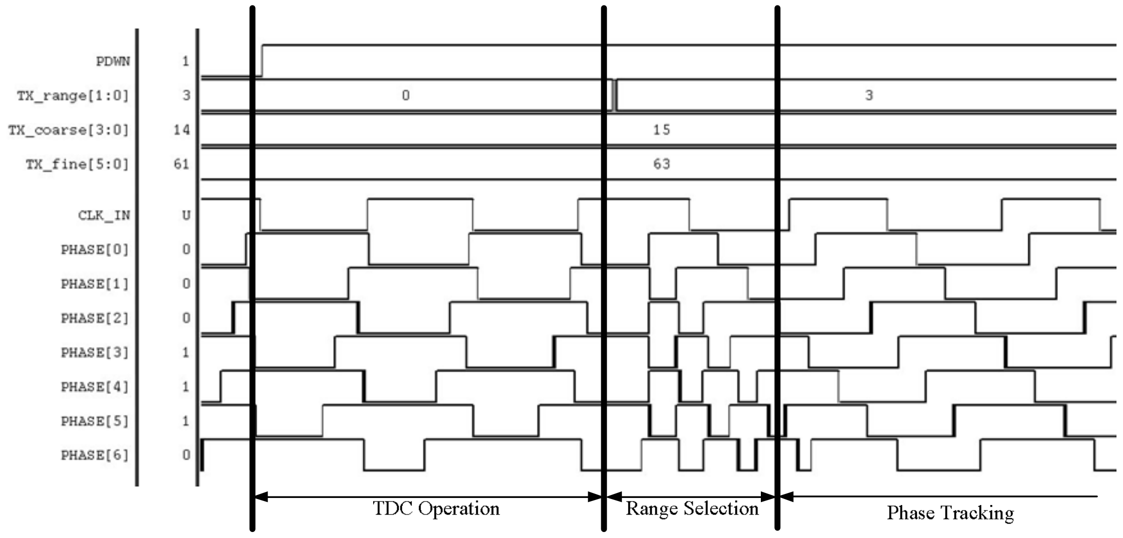


Fig. 5.6: The transient response of the ADMCG (at 85MHz)

The worst-case lock-in time of the proposed ADMCG, in terms of reference clock cycles, is equal to $T_{UPDATE}*(T_{TDC}+(P_{COARSE}-1)*2)$, where T_{UPDATE} means the ADMCG controller update interval, T_{TDC} means the TDC operation time, and P_{COARSE} means total paths in coarse-tuning stage.

To make sure that the previous update of DCDL control code takes effects on delay line's output, the ADMCG controller cannot update the DCDL control code at every cycle. Hence the T_{UPDATE} is chosen as 4. And TDC only needs one clock cycle to estimate the reference clock's period. Therefore the total lock-in time for the 7-phase ADMCG is < 124 (=4*(1+(16-1)*2) reference clock cycles.

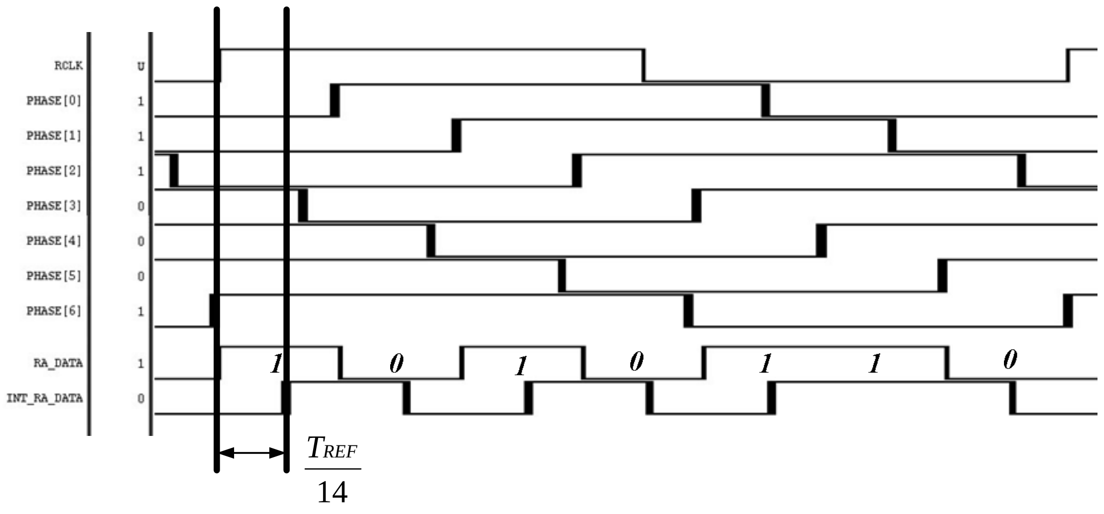
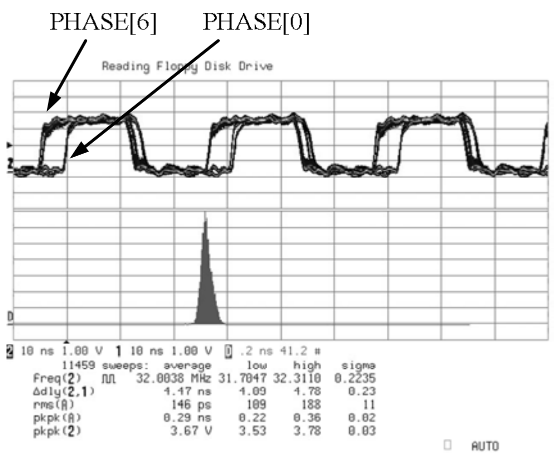


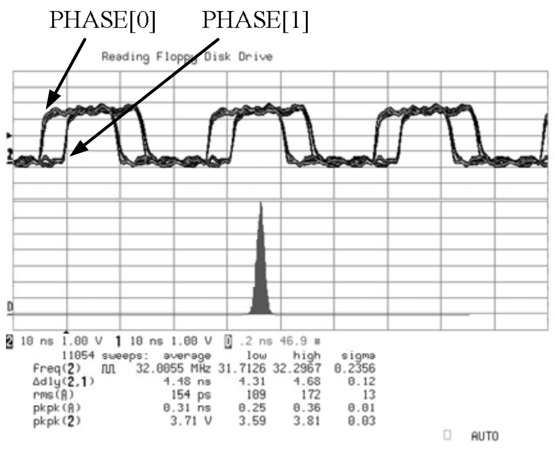
Fig 5.7: The post-layout simulation of the receiver (at 85MHz)

Fig. 5.7 shows the operation of the receiver. In the receiver, the 7-phase ADMCG generates 7-phase multi-phase clock signals (PHASE[6:0]) from the data's reference clock (RCLK). After ADMCG is locked, the 2-phased ADMCG estimates the $T_{REF}/14$ delay and then the received data stream (RA_DATA) is delayed by $T_{REF}/14$, which is also shown in Fig. 5.7 as INT_RA_DATA. As a result, the receiver can directly use the generated multi-phase clock signals to sample the delayed received data stream (INT_RA_DATA) in the center of bit symbol boundary and achieves a maximal timing margin in the receiver circuit.



```
2 10 ns 1.00 V  1 10 ns 1.00 V  0 .2 ns 41.2 #
      11459 sweeps:  average      low    high   sigma
  Freq(2)  ⎍⎍   32.0038 MHz 31.7047 32.3110  0.2235
  Δdly(2,1)         4.47 ns    4.09    4.78    0.23
  rms(A)            146 ps     109     188      11
  pkpk(A)          0.29 ns    0.22    0.36    0.02
  pkpk(2)           3.67 V    3.53    3.78    0.03
                                                      □  AUTO
```

(a)



```
2 10 ns 1.00 V  1 10 ns 1.00 V  0 .2 ns 46.9 #
      11054 sweeps:  average      low    high   sigma
  Freq(2)  ⎍⎍   32.0055 MHz 31.7126 32.2967  0.2356
  Δdly(2,1)         4.48 ns    4.31    4.68    0.12
  rms(A)            154 ps     109     172      13
  pkpk(A)          0.31 ns    0.25    0.36    0.01
  pkpk(2)           3.71 V    3.59    3.81    0.03
                                                      □  AUTO
```

(b)

Fig. 5.8: Measured multi-phase clock signals (at 32MHz).

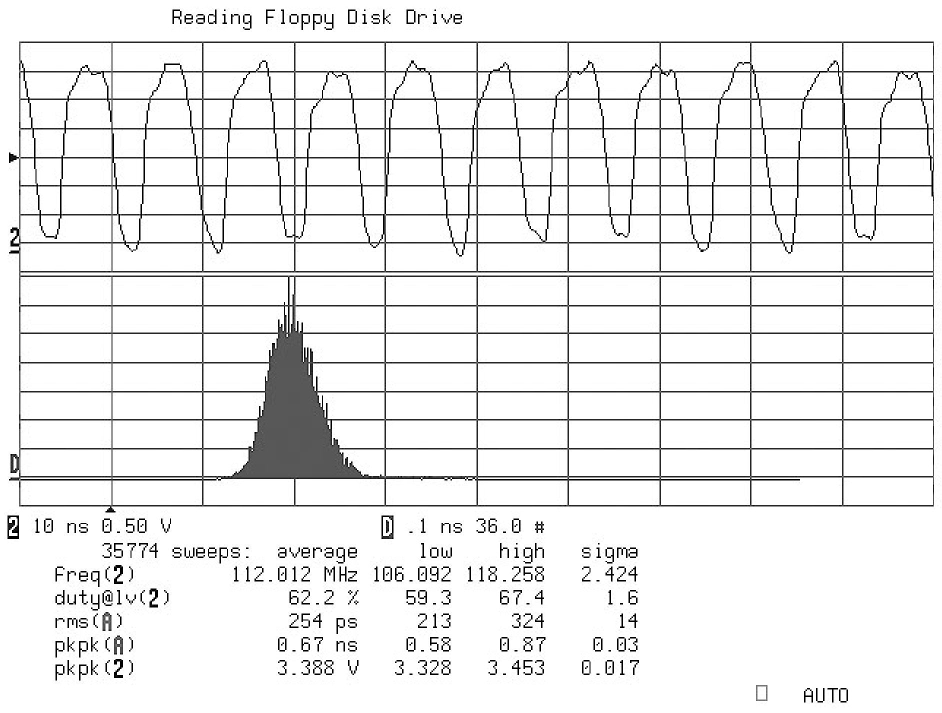
Fig. 5.8 shows the measured multi-phase clock signals with noisy digital circuitry (≈ 600mVpp supply noise). The reference clock is 32MHz oscillator with root-mean-square (rms) jitter: 79ps and P_K-P_K jitter: 180ps.

Due to the limitations of digital scope, only two data channels can be displayed simultaneously. Therefore PHASE[6] and PHASE[0] are shown in Fig. 5.8(a), and PHASE[0] and PHASE[1] are shown in Fig. 5.8(b).

The long-term P_K-P_K jitter histogram of output multi-phase clock signals and the measured delay time between two adjacent phases are also shown. Ideally, two adjacent phases should be 4.464ns (=(1/32MHz)/7) apart, and the measured results show that the maximum error is less than 0.36% (=(4.48ns$_{(\text{PHASE[0]} \sim \text{PHASE[1]})}$ – 4.464ns$_{(\text{Ideal})}$)/4.464ns). And the long-term rms jitter and P_K-P_K jitter of ADMCG's output are 154ps and 310ps respectively.

A repetition data stream "10101010…" is applied to the transmitter where the transmitted data (TX_DATA) have a transition at every rising edge of multi-phase clock signals. This test pattern is used to measure the output data jitter and check the stability of the

ADMCG's output. Thus the transmitted data looks like a clock signal and its frequency is 3.5 (=7/2) times higher than the reference clock. Fig. 5.9 shows the measured long-term P_K-P_K jitter histogram of the transmitted data. By chip measurement, the transmitted data's rms jitter and P_K-P_K jitter is 254ps and 670ps respectively.



```
                 Reading Floppy Disk Drive

2 10 ns 0.50 V                      D .1 ns 36.0 #
     35774 sweeps:  average      low    high   sigma
  Freq(2)        112.012 MHz 106.092 118.258   2.424
  duty@lv(2)          62.2 %    59.3    67.4     1.6
  rms(A)             254 ps     213     324      14
  pkpk(A)           0.67 ns    0.58    0.87    0.03
  pkpk(2)           3.388 V   3.328   3.453   0.017
                                                        AUTO
```

Fig. 5.9: Measured long-term jitter histogram of the transmitted data (at 32MHz).

Since the ADMCG needs to continue tracking the phase of the reference clock, jitter of the reference clock will influence the measurement for the output jitter of the ADMCG and the transmitted data jitter.

Total gate count of the transmitter and the receiver is 7343 and 9683 respectively, where the gate count of the 7-phase ADMCG is 7203. Power consumption of the transmitter is 17.3mW at 20MHz and 75.1mW at 85MHz. Power consumption of the receiver is 23.6mW at 20MHz and 85.5mW at 85MHz. Fig. 5.10 shows microphotograph of the test chip. And the core area of the test chip is 1380 μm x 1380 μm.

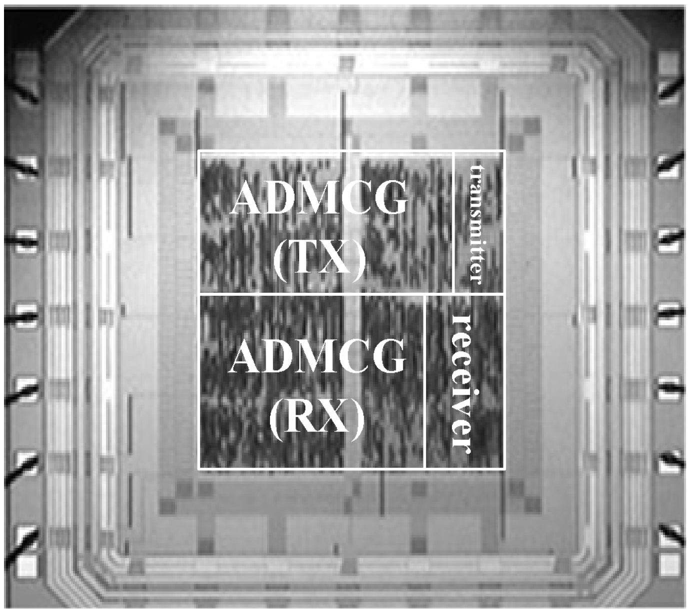


Fig. 5.10: Microphotograph of the ADMCG test chip.

5.4 Summary

In this chapter, an all-digital cell-based multi-phase clock generator is presented. The proposed ADMCG can overcome the false-lock problem in conventional designs. And in test chip, the ADMCG is applied to design a 7:1 data channel compression transceiver. The test chip shows that the proposed ADMCG has a wide frequency range (20-85MHz) and very robust to PVT variations and reference clock jitter. The proposed ADMCG can reduce both design time and circuit complexity. Therefore it is very suitable for many digital communication applications.

Chapter 6

Automated Synthesis of ADPLL, ADDLL, and ADMCG for SoC Applications

In this chapter, an automated synthesis design methodology for ADPLL/ADDLL /ADMCG is presented. In SoC design, the design time for each module is restricted. Thus each module should better be a reusable design so that the total design time for the SoC can be reduced. However, for different applications, the ADPLL/ADDLL/ADMCG may have different operating ranges or different lock-in time requirements, making it hard to design one ADPLL/ADDLL/ADMCG suitable for all applications. As a result, it often needs to redesign the ADPLL/ADDLL/ADMCG for target application and resulting longer design phase.

Due to well-developed cell libraries and logic synthesis tools, most logic or algorithmic operations, such as additions, multiplications, can easily be created using Hardware Description Language (HDL) with logic synthesizer. Thus if the ADPLL/ADDLL/ADMCG can also be automatic generated using standard cells from the cell-library, it can greatly reduce the design time and the design complexity for the ADPLL/ADDLL/ADMCG, and also has the best portability for different processes.

An EDA tool for clock synthesis PLL generation is proposed in [44] to simplify and accelerate all tasks associated with the design and deployment of high performance clock synthesis PLLs. But the proposed approach [44] is to provide a compiled analog PLL. Therefore it takes extra time to redesign analog circuits and do some custom layouts before the PLL compiler can be used. So if the ADPLL/ADDLL/ADMCG can be generated just using standard cells, users can quickly build up their ADPLL/ADDLL/ADMCG in a short time. And this is the goal for this research.

In this chapter, the proposed automated synthesis methodology provides a guideline to take user specifications, such as: output clock range, lock-in time, and so on to build up the suitable architecture for ADPLL/ADDLL/ADMCG design. The proposed flexible ADPLL/ADDLL/ADMCG architecture can be easily modified to fit different applications.

As a result, the proposed methodology uses both benefit of digital VLSI and cell-based design to build up the user-specified ADPLL/ADDLL/ADMCG in a short time, and reduces the design time and the design complexity of ADPLL/ADDLL/ADMCG, making it very suitable for System-On-Chip (SoC) applications.

This chapter is arranged as follows: section 6.1 describes the automated ADPLL synthesis flow. And the test chip measurement results are also shown in this section. Section 6.2 describes the automated ADDLL synthesis flow. Section 6.3 describes the automated ADMCG synthesis flow. Finally, a brief summary is made in section 6.4.

6.1 Automated ADPLL Synthesis

6.1.1 Cell Library Data Preparation

In chapter 3, most portions of the proposed ADPLL functional blocks can be described with behavior Hardware Description Language (HDL) code. And logic synthesizer is used to synthesize those behavior HDL codes to the gate-level HDL code. However, due to the limitations of the gate-delay model, the resolution of the DCO and the dead zone of the PFD can only be determined by SPICE circuit simulation. As a result, before the automated ADPLL synthesis flow can be used, the fine-tuning delay cell of the DCO and the PFD must be redesigned for the target cell library.

In section 2.1, the proposed fine-tuning delay cell of the DCO is created using standard cells. After SPICE circuit simulation, the lookup table for control the fine-tuning delay cell is created, and the resolution of the fine-tuning delay cell can then be determined. And the delay time of the coarse-tuning delay cell can also be determined from SPICE circuit simulation. The ring delay line of the TDC is a copied architecture from the DCO, so it can just mirror the architecture of the DCO with some reductions.

The design of the PFD is to select suitable standard cells to improve the sensitivity of the PFD (i.e. reduce the dead zone of the PFD). Since the sensitivity of the PFD is major determined by the digital pulse amplifier, thus the stage numbers of the digital pulse amplifier must be decided after SPICE circuit simulation.

After the fine-tuning delay cell of DCO and the PFD are created and the suitable coarse-tuning delay cell is selected from the cell library, the gate-level HDL code of the fine-tuning delay cell and the PFD is ready for the ADPLL architecture compiler. And the timing information of those modules can be supplied to the ADPLL architecture compiler to build up the suitable ADPLL architecture.

For different cell libraries, the design for the fine-tuning delay cell and the PFD must be performed once. And then the ADPLL architecture compiler can be used to build up the whole ADPLL circuit for different design specifications based on the current cell library.

6.1.2 ADPLL Architecture Compiler

The major design specifications for the ADPLL design are listed in table 6.1. The

ADPLL architecture compiler takes those specifications as the design parameters for choosing a suitable ADPLL architecture. Then it generates the HDL code for ADPLL design.

Table 6.1: The ADPLL SPECIFICATIONS

Parameter		Description
1	Target Process	Target Foundry Process
2	Reference Clock (MHz)	Min. and Max. required reference clock frequency
3	Output Clock (MHz)	Min. and Max. required output clock frequency
4	Programmable Input and Feedback Divider	Max. divide ratios for input and feedback divider
5	Lock-in time (# cycle)	Max. required lock-in time

The ADPLL output frequency range is determined by the DCO. The DCO's output clock period can be expressed as Eq. 6.1.

$$\text{PERIOD\_MAX} = \text{PERIOD\_MIN} + \text{COARSE\_UNIT} * N_{coarse} \qquad \text{(Eq. 6.1)}$$

where the PERIOD_MIN and PERIOD_MAX denote that the minimum and maximum clock period of the DCO. The N_{coarse} is the number of coarse-tuning delay cells used in coarse-tuning path selector, and the step size for DCO coarse-tuning is denoted as COARSE_UNIT. The delay time of the PERIOD_MIN comes from the gate delay of coarse-tuning path selector, fine-tuning delay cell and the reset stage.

The value of PERIOD_MIN and COARSE_UNIT can be gained from the cell library timing information. And when the output clock range is specified, the needed coarse-tuning delay cells (N_{coarse}) to cover the required output frequency range can be calculated from Eq. 6.1.

When the number of coarse-tuning delay cells (N_{coarse}) is determined, then the worst-case lock-in time in term of reference clock cycles for a binary search ADPLL controller can be calculated (see section 3.3.2). If the lock-in time of the binary search ADPLL controller doesn't meet the lock-in time specification, the TDC module is inserted to reduce the ADPLL's lock-in time.

The bit width for the TDC counter and the TDC digital processing unit are determined by the maximum reference clock period. After the reference clock range is specified, the bit width must be large enough to avoid overflow in the TDC counter.

The OUTPUT DCO is removed from the ADPLL circuit if only in-phase clock multiplier is needed. And the maximum divide ratios for input and feedback divider determine the counter bits of frequency divider.

The minimum reference clock period is used to setup the synthesis constraints for ADPLL controller. Since the ADPLL controller updates the DCO control code at reference clock rate. Thus the ADPLL controller needs to run at reference clock speed. Similarly, the frequency divider should work at maximum output clock speed.

The ADPLL architecture compiler builds up the behavior HDL code of the ADPLL controller and the frequency divider and also generates the synthesis constraints, so that the logic synthesizer can create the gate-level circuit to meet the timing requirements. The gate-level code of the DCO, the PFD, and the ring delay line of the TDC is generated by the ADPLL architecture compiler. And those modules link with other modules, which are synthesized by logic synthesizer, to generate the final gate-level HDL code.

6.1.3 Automated ADPLL Synthesis Flow

The proposed automated ADPLL synthesis flow is shown in Fig. 6.1. The design specifications and the cell library information are the inputs for the ADPLL architecture compiler.

The ADPLL architecture compiler builds up a suitable architecture to meet the design specifications and then generate the behavior HDL code and synthesis constraints for the logic synthesizer. For the timing critical modules such as: DCO and PFD, gate-level HDL code is directly generated.

In the proposed DCO architecture, path selector architecture is used to minimize the intrinsic delay time of the DCO, and it can increase the maximum operating frequency of the DCO. For deep-submicron processes, such as 0.18μm CMOS process, the interconnection RC has large influences on the performance of timing critical modules.

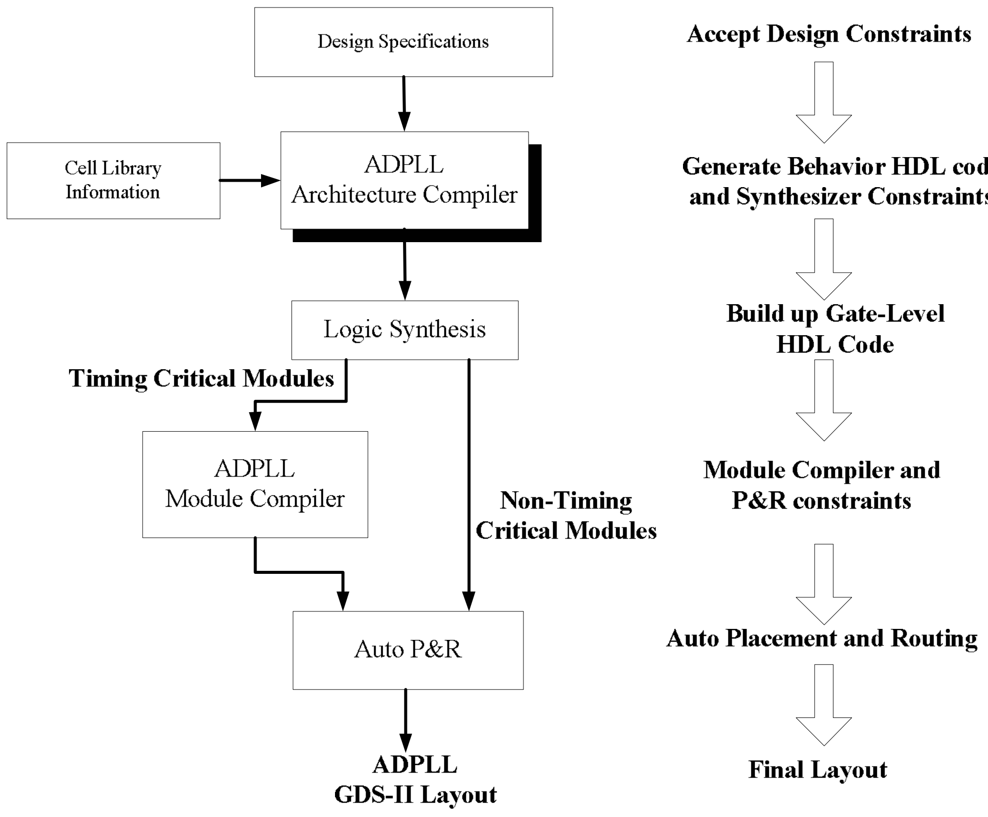

Design Specifications
Cell Library Information
ADPLL Architecture Compiler
Logic Synthesis
Timing Critical Modules
ADPLL Module Compiler
Non-Timing Critical Modules
Auto P&R
ADPLL GDS-II Layout
Accept Design Constraints
Generate Behavior HDL code and Synthesizer Constraints
Build up Gate-Level HDL Code
Module Compiler and P&R constraints
Auto Placement and Routing
Final Layout

Fig. 6.1: The proposed automated ADPLL synthesis flow.

The interconnection RC effects are illustrated in Fig. 6.2. In path selector architecture, the coarse-tuning stage of the DCO provides different delay times by selecting different numbers of delay cell in the delay path. Thus if the time constant R_1C_1 is much larger than R_2C_2, which means the delay time from node D1 to path selector output is much larger than the delay time from node D2 to path selector output, the monotonic response of the DCO may be disappear.

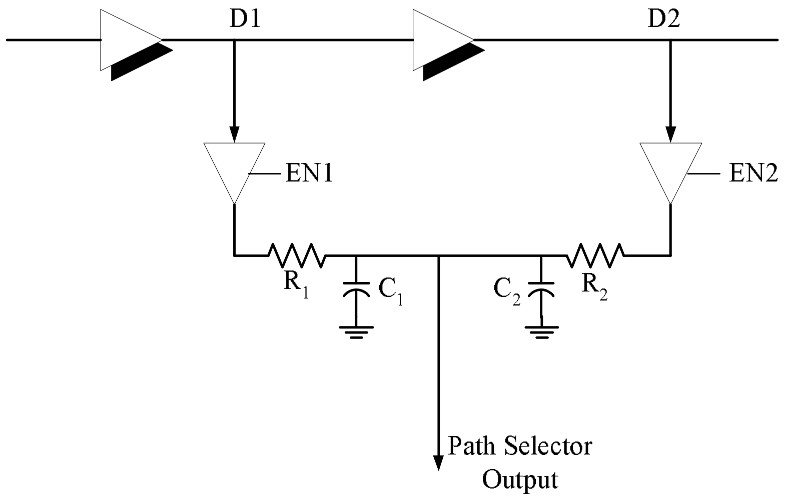


Fig. 6.2: The interconnection RC effects.

Thus the layout of the DCO and the TDC should better be manually placed and routed to minimize and balance the parasitic RC effects. But for 0.35μm CMOS process, since the performance of the DCO and TDC is not dominated by interconnection RC effects. They can be random placed and routed by APR tools with maximum occupied area constraints.

In Fig. 6.1, the ADPLL module compiler is used to generate the layout for the timing critical modules, and then the APR tools are used to complete the final layout of the ADPLL. For the other non-timing critical modules, the timing-driven P&R flow is used to meet the timing constraints. Timing constraints for those modules are translated from the logic synthesizer.

In the proposed automated ADPLL synthesis flow, the SPICE simulation for the fine-tuning delay cell and the PFD are needed. Then the ADPLL architecture compiler can use cell library information to generate the gate-level HDL code of the ADPLL. The layout of timing critical modules is generated by ADPLL module compiler, and final layout is finished by APR tools. By the proposed automated synthesis flow, the design for the ADPLL can be finished in a very short time.

6.1.4 Implementations Results

The proposed ADPLL is implemented with standard 0.35μm, 0.25μm, and 0.18μm CMOS process. Table 6.2 lists the DCO output frequency range for different processes. The coarse-bit means the bit width of coarse-tuning control code. For example, coarse_bit=6 means there are 64 (=2^6) paths in the DCO's coarse-tuning stage. The maximum operating frequency is dependent on the gate delay of the cell libraries. And the minimum operating frequency is dependent on how many coarse-tuning delay cells are used in DCO's coarse-tuning stage.

Table 6.2: DCO OUTPUT FREQUENCY RANGE FOR DIFFERENT PROCESSES

DCO architecture	DCO Output Frequency Range (unit: MHz)					
	0.35 μm		0.25 μm		0.18 μm	
coarse_bit	Min.	Max.	Min.	Max.	Min.	Max.
4	136.054	571.429	166.774	424.809	281.178	746.454
5	077.220	571.429	103.753	424.809	173.212	746.454
6	041.408	571.429	059.093	424.809	097.973	746.454
7	021.482	571.429	031.755	424.809	052.427	746.454

Table 6.3: ADPLL AREA INFORMATION FOR DIFFERENT PROCESSES

DCO architecture	ADPLL Chip Area (unit: μm^2)					
	0.35 μm		0.25 μm		0.18 μm	
coarse_bit	Gate Count	Area	Gate Count	Area	Gate Count	Area
4	4280	258185	3621	78676	4058	51642
5	4827	296205	4036	87967	4662	58393
6	5915	361286	5138	112860	6136	77258
7	8160	503071	6087	134542	7726	98345

Table 6.3 lists the ADPLL area information for different processes. The listed area information is not including the TDC module. This area information is the core area for the

ADPLL circuits after placement and routing. Increasing the coarse_bit may increase the frequency operating range of the ADPLL, but the area cost for the ADPLL is also increased too.

For the same DCO architecture in different processes, the gate-count of the ADPLL is almost the same. But the area of the ADPLL implemented in 0.18μm process is much smaller than the ADPLL implemented in 0.35μm or 0.25μm process.

If the TDC module is included in the ADPLL circuit, the extra gate count 2813 is needed for the TDC circuit. This large area cost of the TDC mainly comes from the TDC digital processing unit, which needs a divider to calculate Eq. 3.7.

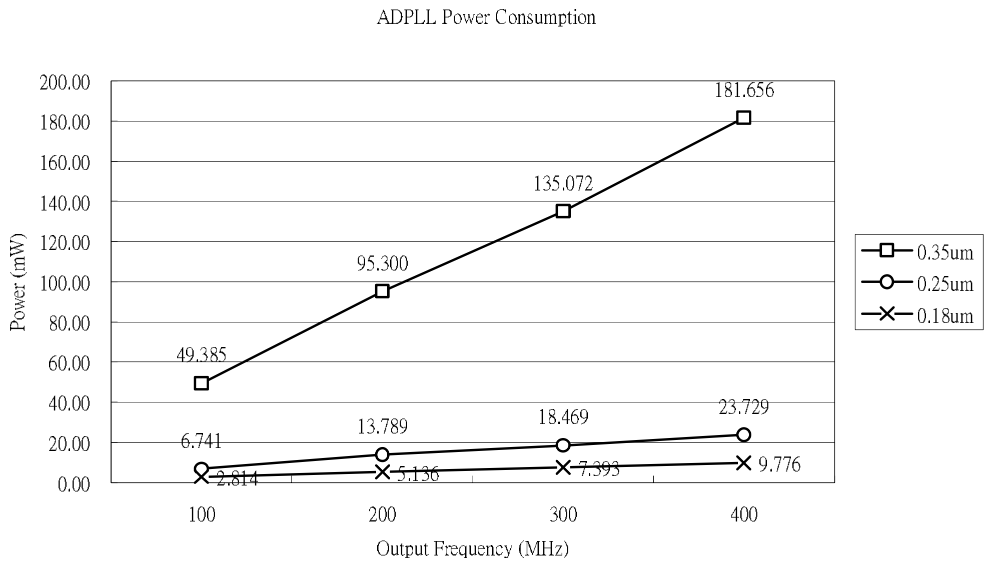


Fig. 6.3: Power Consumption of the ADPLL for different processes.

Fig. 6.3 shows the power consumption of the proposed ADPLL at different output frequencies. In this figure, the coarse_bit for the DCO architecture is 6. Since the parasitic capacitances and supply voltage are both reduced in 0.18μm process, the power consumption of the ADPLL implemented in 0.18μm process is much smaller than the ADPLL implemented in 0.35μm or 0.25μm process. It shows that in 0.18μm process, the maximum power consumption is 9.776 mW (at 400MHz)

6.1.5 Test Chip Measurement Results

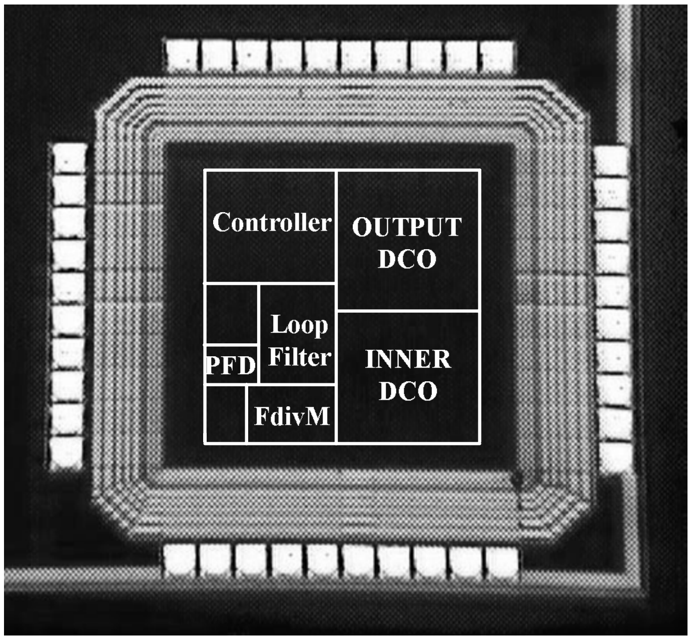


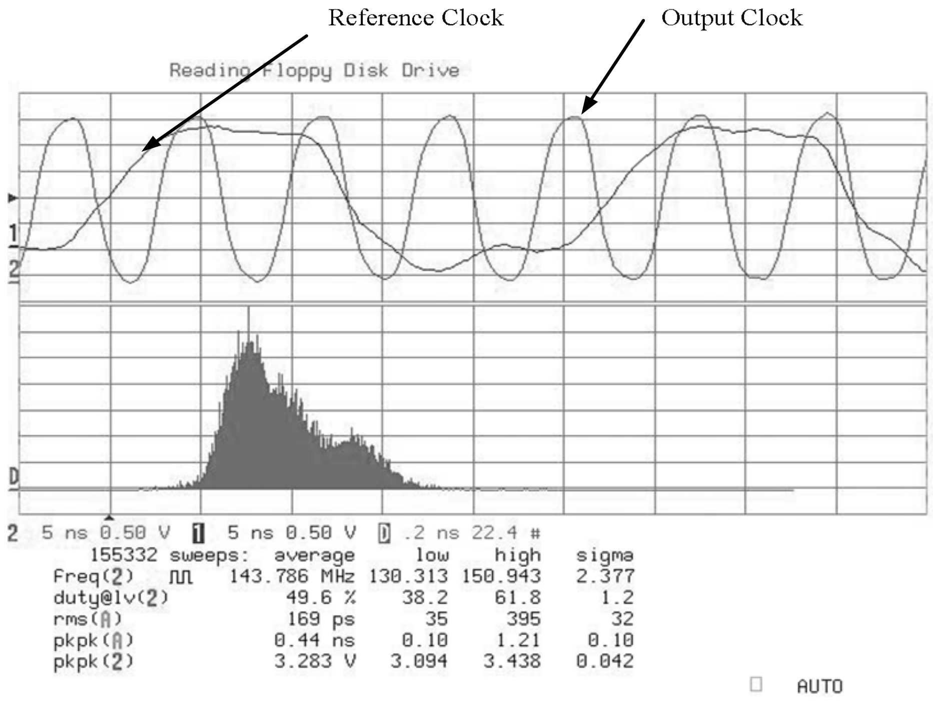
Fig. 6.4: Microphotograph of the ADPLL (UMC 0.18um).

Fig. 6.4 shows microphotograph of one ADPLL test chip for automated ADPLL synthesis flow. This test chip is fabricated on a standard 0.18μm CMOS 1P6M process. The core area of this test chip is 567μm x 567μm. The ADPLL design specifications and the ADPLL architecture compiler synthesis parameters are lists in table 6.4.

Fig. 6.5 shows the measured long-term P_K-P_K jitter histogram of the in-phase clock output (OUT_CLK). In this figure, the noisy reference clock is 36MHz (rms jitter: 60ps, P_K-P_K jitter: 160ps), and M=4, N=1. Thus the output frequency should be 144MHz (=36MHz*4). The long-term rms jitter and $P_\mathbf{k}$-$P_\mathbf{k}$ jitter for the in-phase clock multiplier applications are 169ps and 440ps respectively.

Table 6.4: The ADPLL TEST CHIP SPECIFICATIONS

Parameter		Description
1	Target Process	UMC 0.18um 1P6M Logic
2	Reference Clock (MHz)	1kHz-50MHz
3	Output Clock (MHz)	52MHz-746MHz
4	Programmable Input and Feedback Divider	M_{max}=16383, N_{max}=8
5	Lock-in time (# cycle)	< 50
6	coarse_bit	7
7	Automated ADPLL Synthesis time	229 sec

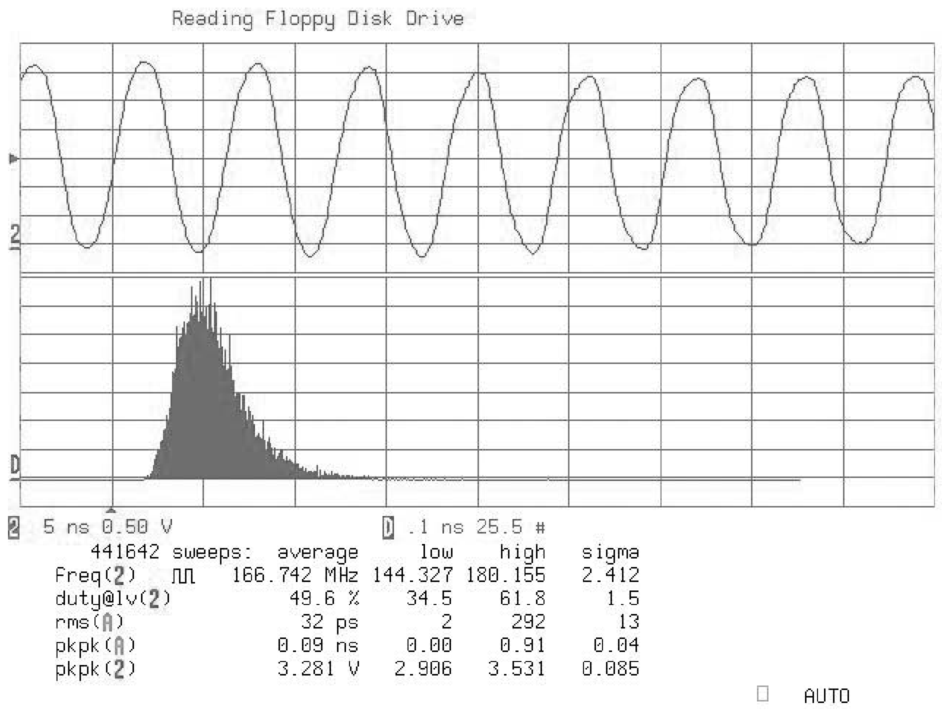


```
2  5 ns 0.50 V  1  5 ns 0.50 V  D .2 ns 22.4 #
     155332 sweeps:  average      low     high   sigma
  Freq(2)   ЛЛ   143.786 MHz 130.313 150.943   2.377
  duty@lv(2)          49.6 %     38.2    61.8     1.2
  rms(A)              169 ps       35     395      32
  pkpk(A)            0.44 ns     0.10    1.21    0.10
  pkpk(2)            3.283 V    3.094   3.438   0.042
```

Fig. 6.5: Measured long-term jitter histogram of in-phase clock output (at 144MHz).

Since the clock multiplier needs to continue tracking the phase of the reference clock, thus the jitter of the reference clock will influence the measurement for the output jitter of the ADPLL. And it can be shown that after ADPLL is locked, the output clock phase is aligned with the rising edge of the reference clock.

Fig. 6.6 shows the measured long-term P_K-P_K jitter histogram of the average clock output (AVG_CLK). In this figure, the reference clock is 10.1 kHz, and M=16383, N=1. Thus the output frequency should be 163.99MHz (=10.1 kHz * 16383). The long-term rms jitter and P_k-P_k jitter for frequency synthesis applications are 32ps and 90ps respectively. In frequency synthesis applications, the loop filter can filter out the resultant noise into the OUTPUT DCO, thus the jitter performance can be improved. In this chip measurement, a very high divide ratio: 16383 is used. The measurement results show that the ADPLL can still work at very high divide ratio.



```
2  5 ns 0.50 V                  D .1 ns 25.5 #
      441642 sweeps:  average      low    high   sigma
   Freq(2)   ЛЛ   166.742 MHz 144.327 180.155   2.412
   duty@lv(2)          49.6 %    34.5    61.8     1.5
   rms(A)               32 ps       2     292      13
   pkpk(A)            0.09 ns    0.00    0.91    0.04
   pkpk(2)           3.281 V    2.906   3.531   0.085
```

Fig. 6.6: Measured long-term jitter histogram of average clock output (at 166MHz).

6.1.6 Summary

In section 6.1, an automated ADPLL synthesis methodology is presented. The proposed synthesis methodology provides the guidelines for ADPLL design. And the proposed

ADPLL architecture is also very flexible and easy to be modified to meet different specifications. The proposed ADPLL is implemented with standard cells, and the implementations for standard 0.35μm/ 0.25μm/ 0.18μm CMOS processes are also shown in this section. From the chip measurement results, it shows that the proposed ADPLL can be used in various applications, and it has better portability than the other designs.

6.2 Automated ADDLL Synthesis

6.2.1 Cell Library Data Preparation

In the proposed ADDLL architecture (see chapter 4), the proposed ADDLL architecture is a cell-based design, and it can be automated synthesized. But due to the limitations of the gate-delay model, the resolution of the DCDL and the dead zone of the PFD can only be determined by SPICE circuit simulation. Thus the fine-tuning delay cell of the DCDL and the PFD must be redesigned for the target cell library.

In section 4.4, the proposed fine-tuning delay cell for DCDL design is created using standard cells. After SPICE circuit simulation, fast delay and slow delay of the fine-tuning delay cell can be determined, thus the phase resolution of the DCDL is determined. And delay time of the coarse-tuning delay cell can also be determined from SPICE circuit simulation.

Two phase detectors (PD) are used in the proposed ADDLL architecture. The coarse-tuning PD can be described using behavior HDL code and then synthesized to the gate-level HDL code. And the design for the fine-tuning PD is discussed in section 2.2.

After the fine-tuning delay cell of DCDL and the fine-tuning PD are created and the suitable coarse-tuning delay cell is selected from the cell library, the gate-level HDL code of the fine-tuning delay cell and the fine-tuning PD is ready for the ADDLL architecture compiler. And the timing information of those modules can be supplied to the ADDLL architecture compiler to build up the suitable ADDLL architecture.

For different cell libraries, the design for the fine-tuning delay cell and the fine-tuning PD must be performed once. And then the ADDLL architecture compiler can be used to build up the whole ADDLL circuit for different design specifications based on the current cell library.

6.2.2 ADDLL Architecture Compiler

The major design specifications for the ADDLL design are listed in table 6.5. The ADDLL architecture compiler takes those specifications as the design parameters for choosing a suitable ADDLL architecture. Then it generates the HDL code for ADDLL design.

Table 6.5: The ADDLL SPECIFICATIONS

Parameter		Description
1	Target Process	Target Foundry Process
2	Reference Clock (MHz)	Min. and Max. required reference clock frequency
3	External buffer delay (psec)	Max. delay time of external buffer tree
4	Phase error (psec)	Max. phase error after DLL is locked
5	Lock-in time (# cycle)	Max. required lock-in time

The ADDLL frequency operating range is determined by the DCDL. The DCDL's operating range can be expressed as Eq. 6.2.

$$\text{DELAY\_MAX} = \text{DELAY\_MIN} + \text{DLL\_COARSE\_UNIT} * N_{\text{dll\_coarse}} \qquad \text{(Eq. 6.2)}$$

where the DELAY_MIN and DELAY_MAX denote that the minimum and maximum delay time which DCDL can provide. The $N_{\text{dll\_coarse}}$ is the number of coarse-tuning delay cells used in coarse-tuning path selector, and the step size for DCDL coarse-tuning is denoted as DLL_COARSE_UNIT. The delay time of the DELAY_MIN comes from the gate delay of coarse-tuning path selector and the intrinsic delay time of fine-tuning delay cell.

The value of DELAY_MIN and DLL_COARSE_UNIT can be gained from the cell library timing information. And when the operating frequency range is specified, the needed coarse-tuning delay cells (N_{dll_coarse}) to cover the required delay range can be calculated from Eq. 6.2.

In the proposed ADDLL architecture, the TDC is used to reduce the lock-in time of the DLL. However, when the lock-in time is not a critical design issue, the TDC can be removed from the architecture, and the shift-register controlled ADDLL controller can be used to reduce the area cost of the ADDLL.

The minimum reference clock period and the maximum extra buffer delay are used to setup the synthesis constraints for ADDLL controller. The ADDLL controller can not update the DCDL control code at every cycle. It needs to wait for the response time of the DCDL and the delay time of the extra buffers. Thus the ADDLL architecture compiler takes those constraints to determine the update period of the ADDLL controller.

In the proposed ADDLL architecture, two PDs are used to minimize the phase error after the DLL is locked. If the requirement for the phase error can be achieved by just using coarse-tuning PD, the fine-tuning PD is removed from the architecture to reduce the area cost of the ADDLL.

The ADDLL architecture compiler builds up the behavior HDL code of the ADDLL controller and also generates the synthesis constraints, so that the logic synthesizer can create the gate-level circuit to meet the timing requirements. The gate-level code of the DCDL, and the fine-tuning PD are generated by the ADDLL architecture compiler. And those modules link with other modules, which are synthesized by logic synthesizer, to generate the final gate-level HDL code.

6.2.3 Automated ADDLL Synthesis Flow

The proposed automated ADDLL synthesis flow is very similar to the ADPLL synthesis flow as discussed in section 6.1.3. The difference is that the ADDLL architecture compiler and the ADDLL module compiler is used in the automated synthesis flow.

The design specifications and the cell library information are the inputs for the ADDLL architecture compiler. The ADDLL architecture compiler builds up a suitable architecture to meet the design specifications and then generate the behavior HDL code and synthesis constraints for the logic synthesizer. For the timing critical modules such as: DCDL and fine-tuning PD, gate-level HDL code is directly generated.

Due to the interconnection RC effects as discussed in section 6.1.3, the layout of the DCDL should better be manually placed and routed to minimize and balance the parasitic RC effects. But for 0.35um CMOS process, since the performance of the DCDL is not dominated by interconnection RC effects. They can be random placed and routed by APR tools with maximum occupied area constraints.

The layout for the timing critical modules is generated by ADDLL module compiler, and then the APR tools are used to complete the final layout of the ADDLL. For the other non-timing critical modules, the timing-driven P&R flow is used to meet the timing constraints. Timing constraints for those modules are translated from the logic synthesizer.

In the proposed automated ADDLL synthesis flow, the SPICE simulation for the fine-tuning delay cell and the fine-tuning PD are needed. Then the ADDLL architecture compiler can use cell library information to generate the gate-level HDL code of the ADDLL. The layout of timing critical modules is generated by ADDLL module compiler, and final layout is finished by APR tools. By the proposed automated synthesis flow, the design for the ADDLL can be finished in a very short time.

6.3 Automated ADMCG Synthesis

6.3.1 Cell Library Data Preparation

The proposed ADMCG architecture (see chapter 5) is a DLL-based architecture. Thus the needed cell library data information is almost the same with automated ADDLL synthesis flow discussed in section 6.2.1. The difference is that the range selection delay cell must be determined first then the suitable coarse-tuning delay cell is selected from the cell library.

After the fine-tuning delay cell of DCDL and the fine-tuning PD are created and the suitable range selection delay cell and coarse-tuning delay cell are selected from the cell library, the gate-level HDL code of the fine-tuning delay cell and the fine-tuning PD is ready for the ADMCG architecture compiler. And the timing information of those modules can be supplied to the ADMCG architecture compiler to build up the suitable ADMCG architecture.

For different cell libraries, the design for the fine-tuning delay cell and the fine-tuning PD must be performed once. And then the ADMCG architecture compiler can be used to build up the whole ADMCG circuit for different design specifications based on the current cell library.

6.3.2 ADMCG Architecture Compiler

The major design specifications for the ADMCG design are listed in table 6.6. The ADMCG architecture compiler takes those specifications as the design parameters for choosing a suitable ADMCG architecture. Then it generates the HDL code for ADMCG design.

The ADMCG frequency operating range is determined by the DCDL's range selection stage. The DCDL's operating range can be expressed as Eq. 6.3.

$$\text{DLINE\_MAX}= (\text{DLINE\_MIN} + \text{RANGE \_UNIT}*N_{\text{dll\_range}})*\text{Phase\_Number} \quad \text{(Eq. 6.3)}$$

where DLINE_MIN denotes that the minimum delay time in each delay stage, and DLINE_MAX denotes the max delay time which the total DCDL can provide. The N_{dll_range} is the number of range selection delay cells used in range selection path selector, and the step size for DCDL range selection is denoted as RANGE _UNIT. DELAY_MIN comes from the gate delay of range selection path selector, coarse-tuning path selector, and minimum delay time of fine-tuning delay cell. And since each delay stage is controlled by the same control code, thus the total delay time of the DCDL is the delay time of each stage multiplies by the phase number of multi-phase clock signals.

Table 6.6: The ADMCG SPECIFICATIONS

Parameter		Description
1	Target Process	Target Foundry Process
2	Reference Clock (MHz)	Min. and Max. required reference clock frequency
3	Multi-Phase number	Number of multi-phase clock signals
4	Phase Accuracy (psec)	Phase error in each multi-phase clock signals
5	Lock-in time (# cycle)	Max. required lock-in time

The value of DLINE_MIN and RANGE_UNIT can be gained from the cell library timing information. And when the operating frequency range is specified, the needed range selection delay cells (N_{dll_range}) to cover the required delay range can be calculated from Eq. 6.3.

In the proposed ADMCG architecture, the TDC is used to calculate the suitable delay range of the DCDL and avoid the false-lock problem as mentioned in section 5.1. However, when the lock-in time is not a critical design issue, the TDC can be removed from the architecture, and the shift-register controlled ADMCG controller can be used to reduce the area cost of the ADMCG.

The minimum reference clock period are used to setup the synthesis constraints for

ADMCG controller. Since the ADMCG controller updates the DCDL control code at reference clock rate. Thus the ADPLL controller needs to run at reference clock speed.

The ADMCG architecture compiler builds up the behavior HDL code of the ADMCG controller and also generates the synthesis constraints, so that the logic synthesizer can create the gate-level circuit to meet the timing requirements. The gate-level code of the DCDL, and the fine-tuning PD are generated by the ADMCG architecture compiler. And those modules link with other modules, which are synthesized by logic synthesizer, to generate the final gate-level HDL code.

In the ADMCG architecture, since the last phase is connected to the phase detector and internal control circuits, thus the loading in the last phase is different from the other phases. To improve the phase accuracy in multi-phase clock generation, the loading in each phase must be balanced. Thus after the gate-level HDL code of the ADMCG is generated by the logic synthesizer, the ADMCG architecture compiler will find out the loading cells of the last phase and add those loading cells to the other phases as dummy loads for balance the loading effects.

6.3.3 Automated ADMCG Synthesis Flow

The proposed automated ADMCG synthesis flow is very similar to the ADPLL synthesis flow as discussed in section 6.1.3. The difference is that the ADMCG architecture compiler and the ADMCG module compiler is used in the automated synthesis flow.

The design specifications and the cell library information are the inputs for the ADMCG architecture compiler. The ADMCG architecture compiler builds up a suitable architecture to meet the design specifications and then generate the behavior HDL code and synthesis constraints for the logic synthesizer. For the timing critical modules such as: DCDL and fine-tuning PD, gate-level HDL code is directly generated.

Due to the interconnection RC effects as discussed in section 6.1.3, the layout of the DCDL should better be manually placed and routed to minimize and balance the parasitic RC effects. But for 0.35μm CMOS process, since the performance of the DCDL is not dominated by interconnection RC effects. They can be random placed and routed by APR tools with maximum occupied area constraints.

The layout for the timing critical modules is generated by ADMCG module compiler, and then the APR tools are used to complete the final layout of the ADMCG. For the other non-timing critical modules, the timing-driven P&R flow is used to meet the timing constraints. Timing constraints for those modules are translated from the logic synthesizer.

In the proposed automated ADMCG synthesis flow, the SPICE simulation for the fine-tuning delay cell and the fine-tuning PD are needed. Then the ADMCG architecture compiler can use cell library information to generate the gate-level HDL code of the ADMCG. The layout of timing critical modules is generated by ADMCG module compiler, and final layout is finished by APR tools. By the proposed automated synthesis flow, the design for the ADDLL can be finished in a very short time.

6.4 Summary

In this chapter, an automated ADPLL/ADDLL/ADMCG synthesis flow is presented. The proposed design methodology provides the guidelines for ADPLL/ADDLL/ADMCG design. And the proposed automated synthesis flow provides a fast way to build up the ADPLL/ADDLL/ADMCG for different target applications. The proposed automated synthesis flow reduces design time for ADPLL/ADDLL/ADMCG design. Therefore it is very suitable for SoC applications.

Chapter 7

Conclusions and Future Works

In this dissertation, we have proposed the solutions to build up a high performance cell-based ADPLL/ADDLL/ADMCG for practical systems. The proposed binary search or TDC-based ADPLL can achieve fast locking time. And the proposed loop filter with two DCOs architecture can improve the jitter performance of ADPLL. As a result, the proposed ADPLL is very suitable for on-chip high-speed clock generation as demanded in current SoC designs.

The proposed ADDLL utilizes TDC and DTC to complete coarse-tuning phase acquisition in one clock cycle, resulting in less lock-in time. And the area cost is reduced by sharing TDC and DTC with delay line. The proposed DCDL can turn off unused delay cells at high frequency operation, thus it can save unwanted power consumptions in high frequency operation. Thus it is very suitable for wide-range clock deskew applications demanded in system-level integration.

The proposed DLL-based ADMCG uses a TDC to choose a reasonable delay range rather than to use self-correcting circuit. Thus its operation is very robust to PVT variations and can avoid possible false-lock to harmonics. The lock-in time of the proposed ADMCG can also be reduced by adding TDC module. The proposed ADMCG is applied to design a 7:1

data channel compression transceiver. In the proposed receiver architecture, which use a 2-phase ADMCG to estimate real value of $T_{REF}/14$, can maximizes the timing margin of the receiver circuit.

The proposed automated synthesis methodology provides a guideline to take user specifications to build up the suitable architecture for ADPLL/ADDLL/ADMCG design. The proposed methodology uses both benefits of digital VLSI and cell-based design to build up the user-specified ADPLL/ADDLL/ADMCG in a short time, and reduces design time and design complexity of ADPLL/ADDLL/ADMCG, making it very suitable for SoC designs.

Nevertheless, for some applications, which demand for very high accurate frequency synthesis, the proposed ADPLL architecture may be difficult to meet the required specifications. Since the proposed ADPLL architecture is integer-N architecture, thus output frequency resolution is dependent on reference frequency, and fine frequency resolution requires a small reference frequency. However, if reference clock rate is low, it may result in long lock-in time.

Thus the fractional-N PLL architecture [45] is a possible way to achieve a fine frequency resolution in frequency synthesizer applications. In this architecture, the integer divide ratio is periodically altered from N to N+1. The resulting average divide ratio will be increased from N by duty cycle of the N+1 division.

But this architecture may induce severe spurious tones due to the periodic modification of the divider modulus. It needs to be carefully designed to filter out this phase error and preserve the benefit of fractional-N architecture. Thus, the design and implementation of fractional-N ADPLL to achieve fine frequency resolution is the research topics remaining to be solved.

In the proposed automated ADPLL/ADDLL/ADMCG synthesis flow, the module compiler is needed to directly generate the layout of timing critical modules, such as: DCO and TDC. Currently these tasks are done by manual placement and routing. But in the further, the automatic layout generator for timing critical modules should be developed so that the proposed automated ADPLL/ADDLL/ADMCG design flow can be used in the deep sub-micron (< 0.18μm) CMOS processes.

For low-power or mobile applications, ADPLL/ADDLL/ADMCG needs to provide a power-saving mode so that the system can turn off the ADPLL/ADDLL/ADMCG if needed. In those applications, the previous lock states of ADPLL/ADDLL/ADMCG can be recorded, and when the system returns to its normal operation, the lock-in time can then be reduced. And in frequency synthesis applications, the proposed ADPLL may periodically turn on/off frequency and phase acquisition circuits (including INNER DCO) to save the active power. But this scheme may worsen the performance of ADPLL. Thus a better way to save active power consumption still needs to be investigated for further research.

References:

[1] Hee-Tae Ahn, and David J. Allstot, “A Low-Jitter 1.9-V CMOS PLL for UltraSPARC Microprocessor Applications,” *IEEE Journal of Solid-State Circuits*, Vol. 35, pp.450-454, May 1999.

[2] Inchul Hwang, Soonsub Lee, Sangwon Lee, and Soowon Kim, “A Digitally Controlled Phase-Locked Loop with Fast Locking Scheme for Clock Synthesis Application,” *IEEE International Solid-State Circuits Conference, 2000, Digest of Technical Papers*, pp. 168-169, Feb. 2000.

[3] Koichiro Minami, Muneo Fukaishi, Masayuki Mizuno, Hideaki Onishi, Kenji Noda, Kiyotaka Imai, Tadahiko Horiuchi, Hiroshi Yamaguchi, Takanori Sato, Kazuyuki Nakamura, and Masakazu Yamashina, “A 0.1μm CMOS, 1.2V, 2GHz Phase-Locked Loop with Gain Compensation VCO,” *Proceedings of IEEE Conference on Custom Integrated Circuits*, pp. 213-216, May 2001.

[4] Adrian Maxim, Baker Scott, Edmud M. Schneider, Melvin L. Hagge, Steven Chacko, and Dan Stiurca, “A Low-Jitter 125-1250-MHz Process-Independent and Ripple-Poleless 0.18-μm CMOS PLL Based on a Sample-Reset Loop Filter,” *IEEE International Solid-State Circuits Conference, 2001, Digest of Technical Papers*, pp. 394-395, Feb. 2001.

[5] Ian A. Young, Monte F. Mar, and Bharat Bhushan, “A 0.35μm CMOS 3-880MHz PLL N/2 Clock Multiplier and Distribution Network with Low Jitter for Microprocessors,” *IEEE International Solid-State Circuits Conference, 1997, Digest of Technical Papers*, pp. 330-331, Feb. 1997.

[6] Chang-Hyeon Lee, Jack Cornish, Kelly McClellan, and John Choma Jr., “Design of Low Jitter PLL for Cock Generator with Supply Noise Insensitive VCO,” *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, Vol. 1, pp. 233-236, June 1998.

[7] David W. Boerstler, “A Low-Jitter PLL Clock Generator for Microprocessors with Lock Range of 340-612MHz,” *IEEE Journal of Solid-State Circuits*, Vol. 34, pp. 513-519, Apr. 1999.

[8] Terng-Yin Hsu, Chung-Cheng Wang, and Chen-Yi Lee, “Design and Analysis of a Portable High-Speed Clock Generator,” *IEEE Transactions on Circuit and Systems II: Analog and Digital Signal Processing*, Vol.48, pp. 367-375, Apr. 2001.

[9] Jim Dunning, Gerald Garcia, Jim Lundberg, and Ed Nuckolls, “An All-Digital Phase-Locked Loop with 50-Cycle Lock Time Suitable for High Performance Microprocessors,” *IEEE Journal of Solid-State Circuits*, Vol.30, pp. 412-422, Apr. 1995.

[10] Jin-Jer Jong and Chen-Yi Lee, “A Novel Structure for Portable Digitally Controlled Oscillator,” *Proceedings of the 2001 IEEE International Symposium on Circuits and*

System, Vol.1, pp.272-275, May 2001.

[11] Saeki, T., Nakaoka, Y., Fujita, M., Tanaka, A., Nagata, K., Sakakibara, K., Matano, T., Hoshino, Y., Miyano, K., Isa, S., Nakazawa, S., Kakehashi, E., Drynan, J.M., Komuro, M., Fukase, T., Iwasaki, H., Takenaka, M., Sekine, J., Igeta, M., and Nakanishi, N., "A 2.5-ns Clock Access, 250-MHz, 256-Mb SDRAM with Synchronous Mirror Delay," *IEEE Journal of Solid-State Circuits*, Vol. 31, pp. 1656-1668, Nov. 1996.

[12] Yongsam Moon, Jongsang Choi, Kyeongho Lee, Deog-Kyoon Jeong, and Ming-Kyu Kim, "An All-Analog Multiphase Delay-Locked Loop Using a Replica Delay Line for Wide-Rang Operation and Low-Jitter Performance," *IEEE Journal of Solid-State Circuits*, Vol. 35, pp. 377-384, Mar. 2000.

[13] Ching-Yuan Yang, and Shen-Iuan Liu, "A One-Wire Approach for Skew-Compensating Clock Distribution Based on Bidirectional Techniques," *IEEE Journal of Solid-State Circuits*, Vol. 36, pp. 266-272, Feb. 2001.

[14] Sungjoon Kim, Kyeongho Lee, Yongsam Moon, Deog-Kyoon Jeong, Yunho Choi, and Hyung Kyu Lim, "A 960-Mb/s/pin Interface for Skew-Tolerant Bus Using Low Jitter PLL," *IEEE Journal of Solid-State Circuits*, Vol. 32, pp. 691-700, May 1997.

[15] Dinis M. Santos, "A CMOS Delay Locked Loop and Sub-Nanosecond Time-to-Digital Converter Chip," *IEEE Transactions on Nuclear Science*, Vol. 43, pp. 1717-1719, June 1996.

[16] Feng Lin, Jason Miller, Aaron Schoenfeld, Manny Ma, and R. Jacob Barker, "A Register-Controlled Symmetrical DLL for Double-Data-Rate DRAM," *IEEE Journal of Solid-State Circuits*, Vol. 34, pp. 565-568, Apr. 1999.

[17] Bum-Sik Kim and Lee-Sup Kim, "A Low Power 100MHz All digital Delay-Locked Loop," *Proceedings of IEEE International Symposium on Circuits and Systems*, June 9-12, 1997, Hong Kong.

[18] Guang-Kaii Dehng, June-Ming Hsu, Ching-Yuan Yang, and Shen-Iuan Liu, "Clock-Deskew Buffer Using a SAR-Controlled Delay-Locked Loop," *IEEE Journal of Solid-State Circuits*, Vol. 35, pp. 1128-1136, Aug. 2000.

[19] Bruno W. Garlepp, Kevin S. Donnelly, Jun Kim, Pak S. Chau, Jared L. Zerbe, Charles Huang, Chanh V. Tran, Clemenz L. Portmann, Donald Stark, Yiu-Fai Chan, Thomas H. Lee, and Mark A. Horowitz, "A Portable Digital DLL for High-Speed CMOS Interface Circuits, " *IEEE Journal of Solid-State Circuits*, Vol. 34, pp. 632-644, May 1999.

[20] Joo-Ho Lee, Seon-Ho Han, and Hoi-Jun Yoo, "A 330MHz Low-Jitter and Fast-Locking Direct Skew Compensation DLL," *IEEE International Solid-State Circuits Conference, 2000, Digest of Technical Papers*, pp. 352-353, Feb. 2000.

[21] Stefanos Sidiropoulos and Mark Horowitz, "A Semi-Digital DLL with Unlimited Phase Shift Capability and 0.08-400MHz Operating Range," *IEEE International Solid-State Circuits Conference, 1997, Digest of Technical Papers*, pp. 332-333, Feb. 1997.

[22] Joonbae Park, Yido Koo, and Wonchan Kim, “A Semi-Digital Delay Locked Loop for Clock Skew Minimization,” *Proceedings of 1999 Twelfth International Conference On VLSI Design*, pp. 584-588

[23] Koichiro Minami, Masayuki Mizuno, Hiroshi Yamaguchi, Toshihiko Nakano, Yusuke Matsushima, Yoshikazu Sumi, Takanori Sato, Hisashi Yamashida, Masakazu Yamashina, “A 1GHz Portable Digital Delay-Locked Loop with Infinite Phase Capture Ranges,” *IEEE International Solid-State Circuits Conference, 2000, Digest of Technical Papers*, pp. 350-351, Feb. 2000

[24] Jae Joon Kim, Sang-Bo Lee, Tae-Sung Jung, Chang-Hyun Kim, Soo-In Cho, and Beomsup Kim, “A Low-Jitter Mixed-Mode DLL for High-Speed DRAM Applications,” *IEEE Journal of Solid-State Circuits*, Vol. 35, pp. 1430-1436, Oct. 2000.

[25] Dagnachew Birru, “A NOVEL DELAY-LOCKED LOOP BASED CMOS CLOCK MULTIPLIER,” *IEEE Transactions on Consumer Electronics*, Vol.44, pp. 1319-1322, Nov. 1998.

[26] Yeo-San Song and Jin-Ku Kang, “A DELAY LOCKED LOOP CIRCUIT WITH MIXED-MODE TUNING,” *The First IEEE Asia Pacific Conference on ASICs*, pp. 347-350, Aug. 1999.

[27] David J. Foley and Michael P. Flynn, “CMOS DLL Based 2V, 3.2ps Jitter, 1GHz Clock Synthesizer and Temperature Compensated Tunable Oscillator,” *Proceedings of IEEE 2000 Custom Integrated Circuits Conference*, pp.371-374, May 2000.

[28] David J. Foley and Michael P. Flynn, “A 3.3V, 1.6GHz, Low-Jitter, Self-Correcting DLL Based Clock Synthesizer in 0.5μm CMOS,” *Proceedings of IEEE International Symposium on Circuit and Systems*, Vol.2, pp.249-252, May 2000.

[29] M.-J.Edward Lee, William J. Dally, John W. Poulton, Patrick Chiang, and Stephen F. Greenwood, “An 84-mW 4-Gb/s Clock and Data Recovery Circuit for Serial Link Applications,” *2001 Symposium on VLSI Circuits, Digest of Technical Papers*, pp.149-152, Jun. 2001.

[30] Yongsam Moon, Deog-Kyoon Jeong, and Gijung Ahn, “A 0.6-2.5-Gbaud CMOS Tracked 3X Oversampling Transceiver With Dead-Zone Phase Detection for Robust Clock/Data Recovery,” *IEEE Journal of Solid-State Circuits*, Vol.36, pp.1974-1983, Dec. 2001.

[31] Kouichi Yamaguchi, Muneo Fukaishi, Takehiko Sakamoto, Naoto Akiyama, and Kazuyuki Nakamura, “A 2.5-GHz Four-Phase Clock Generator With Scalable No-Feedback-Loop Architecutre,” *IEEE Journal of Solid-State Circuits*, Vol. 36, pp.1666-1672, Nov. 2001.

[32] Ali Hajimiri, Sotirios Limotyrakis, and Thomas H. Lee, “Jitter and Phase Noise in Ring Oscillators,” *IEEE Journal of Solid-State Circuits*, Vol. 34, pp.790-804, Jun. 1999.

[33] Li Jin Cheng and Qiu Yu Lin, “The Performances Comparison Between DLL and PLL

Based RF CMOS Oscillators,” *Proceedings of 4th International Conference on ASIC*, pp.827-830, Oct. 2001.

[34] Wei-Hung Chen, Guang-Kaai Dehng, Jong-Woei Chen, and Shen-Iuan Liu, “A CMOS 400-Mb/s Serial Link for AS-Memory Systems Using a PWM Scheme,” *IEEE Journal of Solid-State Circuits*, Vol. 36, pp.1498-1505, Oct. 2001.

[35] Ching-Che Chung and Chen-Yi Lee, “An All-Digital Phase-Locked Loop for High-Speed Clock Generation,” *IEEE Journal of Solid-State Circuits*, Vol. 38, pp. 347-351, Feb. 2003.

[36] A. Hatakeyama, H. Mochizuki, T. Aikawa, M. Takita, Y. Ishii, H. Tsuboi, S. Fujioka, S. Yamaguchi, M. Koga, Y. Serizawa, K. Nishimura, K. Kawabata, Y. Okajima, M. Kawano, H. Kojima, K. Mizutani, T. Anezaki, M. Hasegawa, and M. Taguchi, "A 256-Mb SDRAM using a register-controlled digital DLL," *IEEE Journal of Solid-State Circuits*, Vol. 32, pp. 1728 - 1734, Nov. 1997.

[37] Chao Xu, Winslow Sargeant, Kenneh Laker, and Jan Van der Spiegel, “FULLY INTEGRATED CMOS PHASE-LOCKED LOOP WITH 30MHZ TO 2GHZ LOCKING RANGE AND ±35PS JITTER,” *Proceedings of IEEE International Electronics, Circuits and Systems Conference*, pp.55-58, Sep. 2001.

[38] Heung-Gyoon Ryu and Eung-jin Ahn, “DLT REPLACEMENT AND SYNCHRONIZATION IN THE DIGITAL HYBIRID PLL FREQUENCY SYNTHESIZER,” *IEEE Transactions on Consumer Electronics*, Vol. 48, pp. 151-156, Feb. 2002.

[39] Takamoto Watanabe and Shigenori Yamauchi, “An All-Digital PLL for Frequency Multiplication by 4 to 1022 With Seven-Cycle Lock Time,” *IEEE Journal of Solid-State Circuits*, Vol. 38, pp. 198-204, Feb. 2003.

[40] Takanori Saeki, Masafumi Mitsuishi, Hiroaki Iwaki, and Mitsuaki Tagishi, “A 1.3-Cycle Lock Time, Non-PLL/DLL Clock Multiplier Based on Direct Clock Cycle Interpolation for Clock on Demand,” *IEEE Journal of Solid-State Circuits*, Vol. 35, pp. 1581-1590, Nov. 2000.

[41] George Chien and Paul R. Gray, “A 900-MHz Local Oscillator Using a DLL-Based Frequency Multiplier Technique for PCS Applications,“ *IEEE Journal of Solid-State Circuits*, Vol. 35, pp. 1996-1999, Dec. 2000.

[42] David J. Foley and Michael P. Flynn, “CMOS DLL-Based 2-V 3.2-ps Jitter 1-GHz Clock Synthesizer and Temperature-Compensated Tunable Oscillator,“ *IEEE Journal of Solid-State Circuits*, Vol. 36, pp. 417-423, Mar. 2001.

[43] Terng-Yin Hsu, Terng-Ren Hsu, Chung-Cheng Wang, Yi-Chuan Liu, and Chen-Yi Lee, “Design of a Wide-Band Frequency Synthesizer Based on TDC and DVC Techniques,” *IEEE Journal of Solid-State Circuits*, Vol. 37, pp. 1244-1255, Oct. 2002.

[44] John Horan, John Ryan, Kay Hearne, Mark Smith, Niall Donovan, Tholom Kiely,

Ciaran Cahill, and Stephen McDonagh, “PLLXPERT: An EDA tool for clock synthesis PLL generation,” *Proceedings of IEE Workshop on Systems on a Chip*, pp. 2/1-2/4, 2000.

[45] M. Stork, “NEW FRACTIONAL PHASE-LOCKED LOOP FREQUNECY SYNTHESIZER USING A SIGMA-DELTA MODULATOR,” *Proceedings of IEEE International Conference on Digital Signal Processing*, Vol.1, pp. 367-370, 2002.

[46] Ching-Che Chung and Chen-Yi Lee, “A new DLL-based approach for all-digital multi-phase clock generation, “ to be appeared in *IEEE Journal of Solid-State Circuits*.

[47] Broadcom Corporation, “Cable Modem Residential Gateway,“ web link, http://www.broadcom.com/cablemodem-gw.html.

著作目錄及發表論文

1. 期刊論文

Ching-Che Chung and Chen-Yi Lee, "A new DLL-based approach for all-digital multi-phase clock generation," to be appeared in *IEEE Journal of Solid-State Circuits*

Ching-Che Chung and Chen-Yi Lee, "An all-digital phase-locked loop for high-speed clock generation," in *IEEE Journal of Solid-State Circuits*, Vol.38, pp. 347-351, Feb. 2003.

2. 研討會論文

Ching-Che Chung and Chen-Yi Lee, "An all-digital phase-locked loop for high-speed clock generation," in *IEEE International Symposium on Circuits and Systems*, Vol. 3, pp.26-29, May 2002..

Hsie-Chia Chang, Ching-Che Chung, Chien-Ching Lin, and Chen-Yi Lee, "A 300MHz Reed-Solomon decoder chip using inversionless decomposed architecture for Euclidean algorithm," in *28th European Solid-State Circuits Conf. (ESSCIRC)*, Sep. 2002.

Jin-Jer Jong and Chen-Yi Lee, "A Novel Structure for Portable Digitally Controlled Oscillator," in *IEEE International Symposium on Circuits and Systems*, Vol. 1, pp.272-275, May 2001.

Yew-San Lee, Jin-Jer Jong, Tsyr-Shiou Perng, Li-Chyun Hsu, Ming-Yang Jaw, and Chen-Yi Lee, "A memory-based architecture for very high throughput variable length codec design," in *IEEE International Symposium on Circuits and Systems*, Vol. 3, pp.9-12, June 1997.

VITA

Ching-Che Chung received the B.S. degree from National Chiao Tung University, HsinChu, Taiwan, in 1997. Since September 1997, he has been a Ph.D. Student of the Si2 research group in the Department of Electronics Engineering, National Chiao Tung University. His research interests include system-on-chip design methodologies, cell-based and fully-custom VLSI design, high-speed interface circuit design, and wireless baseband processor design.