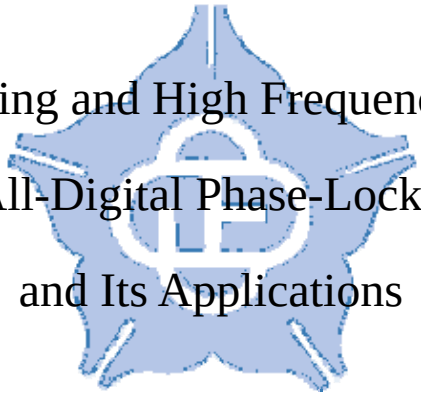
國 立 中 正 大 學

資訊工程研究所碩士論文

快速相位追蹤與高頻率倍數全數位鎖相迴路設計與應用



Fast Phase Tracking and High Frequency Multiplication Factor All-Digital Phase-Locked Loop and Its Applications

研究生： 柯鈞耀

指導教授： 鍾菁哲 博士

中華民國 九十九 年 七 月

國立中正大學碩士班研究生

學位考試同意書

本人所指導 資訊工程學系

研究生 柯鈞耀 所提之論文

應用於視訊系統之快速相位追蹤與高頻率倍數全數位式鎖相迴路 (Fast Phase Tracking and High Frequency Multiplication Factor All-Digital Phase-Locked Loop and Its Applications)

同意其提付 碩 士學位論文考試

指導教授 鍾菁哲 簽章

99 年 5 月 25 日

國立中正大學碩士學位論文考試審定書

資訊工程學系

研究生柯鈞耀 所提之論文

快速相位追蹤與高頻率倍數全數位鎖相迴路設計與應用
(Fast Phase Tracking and High Frequency Multiplication Factor All-Digital Phase-Locked Loop and Its Applications)
經本委員會審查，符合碩士學位論文標準。

學位考試委員會
召　集　人 李順裕 簽章

委　　員 鍾菁哲

李順裕　　

王士豪　　

指　導　教　授 鍾菁哲 簽章

中華民國 99 年 7 月 1 日

博碩士論文電子檔案上網授權書

（提供授權人裝訂於紙本論文書名頁之次頁用）

本授權書所授權之論文爲授權人在 **國立中正大學** (工學院)資訊工程學系所九十八學年度第二學期取得碩士學位之論文。

論文題目：Fast Phase Tracking and High Frequency Multiplication Factor All-Digital Phase-Locked Loop and Its Applications

指導教授：鍾菁哲 博士

茲同意將授權人擁有著作權之上列論文全文（含摘要），非專屬、無償授權國家圖書館及本人畢業學校圖書館，不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

- 讀者基非營利性質之線上檢索、閱覽、下載或列印上列論文，應依著作權法相關規定辦理。

授權人：柯鈞耀

簽　名：柯鈞耀　　中華民國99年7月28日

快速相位追蹤與高頻率倍數全數位鎖相迴路設計與應用

學生 ： 柯鈞耀　　指導教授 ： 鍾菁哲

國立中正大學資訊工程學系研究所

摘要

本論文探討開發應用於視訊系統的全數位時脈產生器，其主要功能是接收顯示卡發出的水平同步訊號 (HSYNC)，並依據使用者設定的螢幕解析度，產生高頻率像素時脈 (Pixel Clock) 來擷取視訊訊號資料。與傳統鎖相迴路相較，本論文提出電路在高頻率倍數鎖相迴路應用上，有較佳的追蹤相位能力。此電路在大量的雜訊干擾時，仍然可以擁有快速追蹤相位能力。並利用數位迴路的優點，可避免在先進製程上產生漏電問題，且可降低晶片面積並避免使用外接元件（如外接電容，額外的石英震盪器）。並開發抵抗水平同步訊號雜訊干擾的數位濾波器，以改善時脈產生器的輸出週期抖動。取樣時脈 (Pixel Clock) 的穩定度直接影響到顯示畫面的品質。因此，如何在高頻率倍數下，產生一個穩定的時脈訊號，是此電路設計的重點。本論文所提出的 ADPLL 將使標準元件庫 65nm 標準 CMOS 製程來製作晶片，驗證所提出的電路架構。

Fast Phase Tracking and High Frequency Multiplication Factor All-Digital Phase-Locked Loop and Its Applications

Student: Chiun-Yao Ko Advisor: Dr. Ching-Che Chung
Department of Computer Science and Information Engineering,
National Chung Cheng University

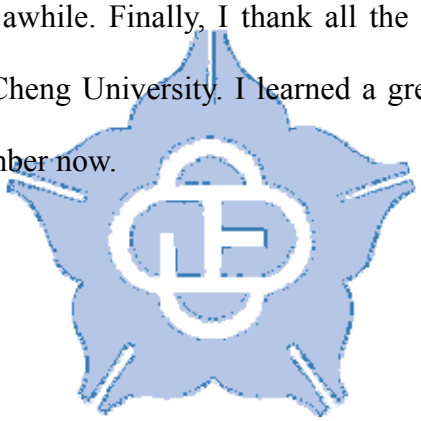
Abstract

An All-Digital Phase-Locked Loop (ADPLL) for video capture application is presented in this dissertation. The major function of this ADPLL is to generate the high speed pixel clock from the horizontal synchronization signal (HSYNC) of the Person Computer (PC) graphics card according to the user-defined video resolution. When it is compared with conventional Phase-Locked Loop (PLL), the proposed design have better phase tracking ability than conventional PLLs in high frequency multiplication factor applications. And it can accept large reference clock jitter and still have fast phase tracking ability. The advantages of digital design can overcome the leakage problem in advanced CMOS process, reduce chip area and avoid to use external components (such as external capacitor and oscillator). The digital loop filter was developed to resist the jitter effects of reference clock and to reduce the period jitter of pixel clock. The stability of sampling clock (pixel clock) will have large effects on video quality. Thus how to generate a very stable, and small phase drift pixel clock, is the major design challenge in this dissertation. And the proposed ADPLL was implemented with standard cells on a standard 65nm CMOS process to verify the performance of the proposed architecture.

Acknowledgements

I would like to express the deepest appreciation to my Professor Ching-Che Chung, who has the attitude and the substance of a genius: he continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this dissertation would not have been possible.

I would like to thank my parents and girl friend Jessy for their love and support while I am depressed for awhile. Finally, I thank all the friends I've met over two years at National Chung Cheng University. I learned a great deal from each of you, none of which I can remember now.



Contents

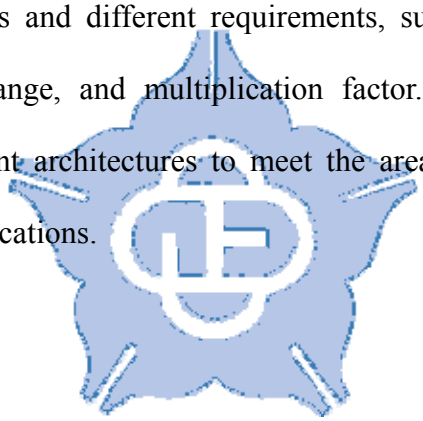
List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

The Phase-Locked Loop (PLL) is usually used for many applications, such as the frequency synthesizer, clock multiplier, clock and data recovery (CDR) and clock deskew. The PLL is also an indispensable module in System-on-a-Chip (SoC). But in reality, we can't have a PLL meet all applications or any system specifications. For different applications and system specifications, PLL has to be redesigned according to the application features and different requirements, such as locking time, jitter suppression, frequency range, and multiplication factor. Therefore, PLL will be implemented with different architectures to meet the area and power consumption requirements in the specifications.



1.2 Design Challenges

1.2.1 The Design Challenges in Conventional PLLs with Advanced Process

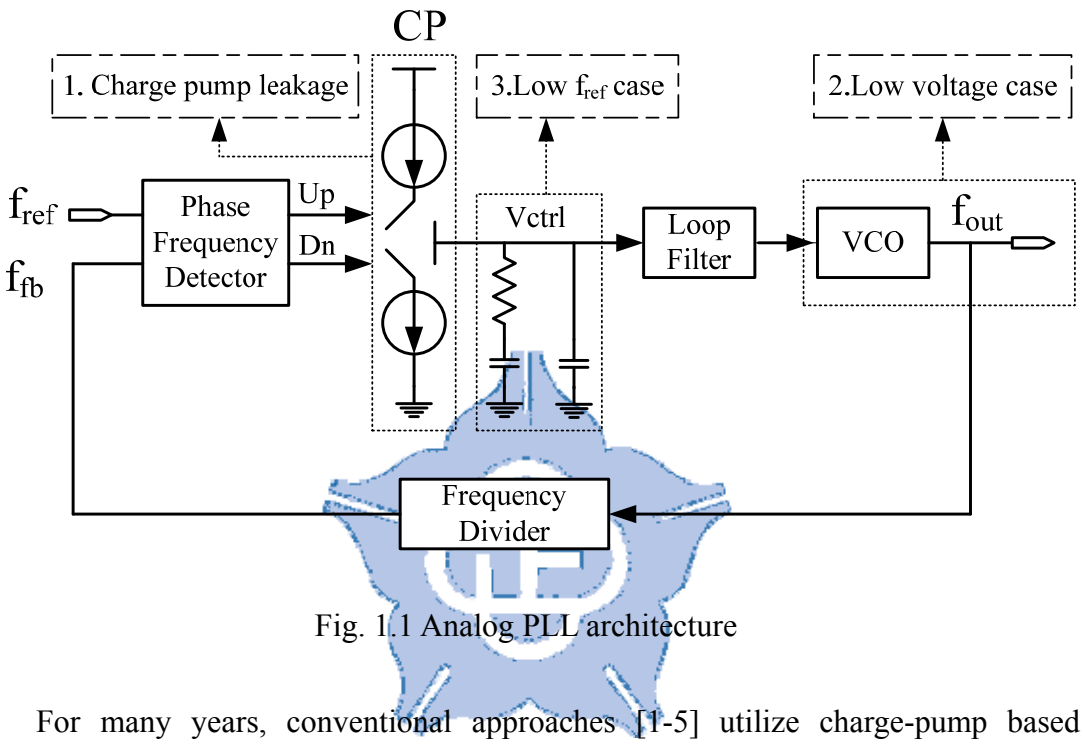


Fig. 1.1 Analog PLL architecture

For many years, conventional approaches [1-5] utilize charge-pump based structure to implement the PLL circuit. Fig. 1.1 shows the block diagram of this architecture. However, in advanced CMOS process, such as 65nm CMOS process, these conventional architectures will encounter great difficulties. In conventional PLL architectures, we will discuss three major design challenges in the following paragraphs.

1.

The most significant problem comes from the charge pump structure, because this structure has to store the control voltage (Vctrl) by capacitor to adjust the voltage controlled oscillator (VCO) and stabilize the output frequency. But to reduce chip area

and avoid using the special process (such as MIM capacitors), mostly the MOS capacitor is used. However, the transistor has serious leakage problem in deep submicron process. The leakage current in charge pump will cause ripple phenomenon and produces jitter on the output clock makes it difficult to design the PLL loop. Because this problem directly affects the jitter performance, the low leakage MOS capacitors are used in PLL to avoid the leakage problem. However, when using the low leakage MOS capacitors, it raises the overall operating voltage (from 1.0V to 1.2V in 65nm CMOS process), leading to increase the dynamic power consumption.

2.

Because the voltage in the 65nm CMOS process has been reduced to 1.0V, so when designs the gain value (K_{VCO}) of the VCO, it needs to trade-off between the output frequency range and the gain value (K_{VCO}). Therefore, it often needs to use the multi-frequency band technique to cut the VCO into several different working sections to resolve the problem in wide frequency range operation. But it also requires auxiliary circuit to do frequency band selection, resulting in the need for additional input control signals, and increasing the circuit costs.

3.

Basically when the PLL's reference clock (f_{REF}) is changed, the control voltage (Vctrl) will be charged or discharged with Up and Dn pulse, respectively, then the analog signal is filtered by the loop filter and transmitted to the VCO. Therefore, in some applications such as the high multiplication factor video clock generator, the reference clock frequency is very low (< 100kHz). Because the PLL loop refresh rate

is too low, due to the leakage problem of the MOS capacitors, the control voltage will have ripple phenomenon and resulting in unexpected period jitter.

1.2.2 The Difficulties of the Phase Tracking in PLL

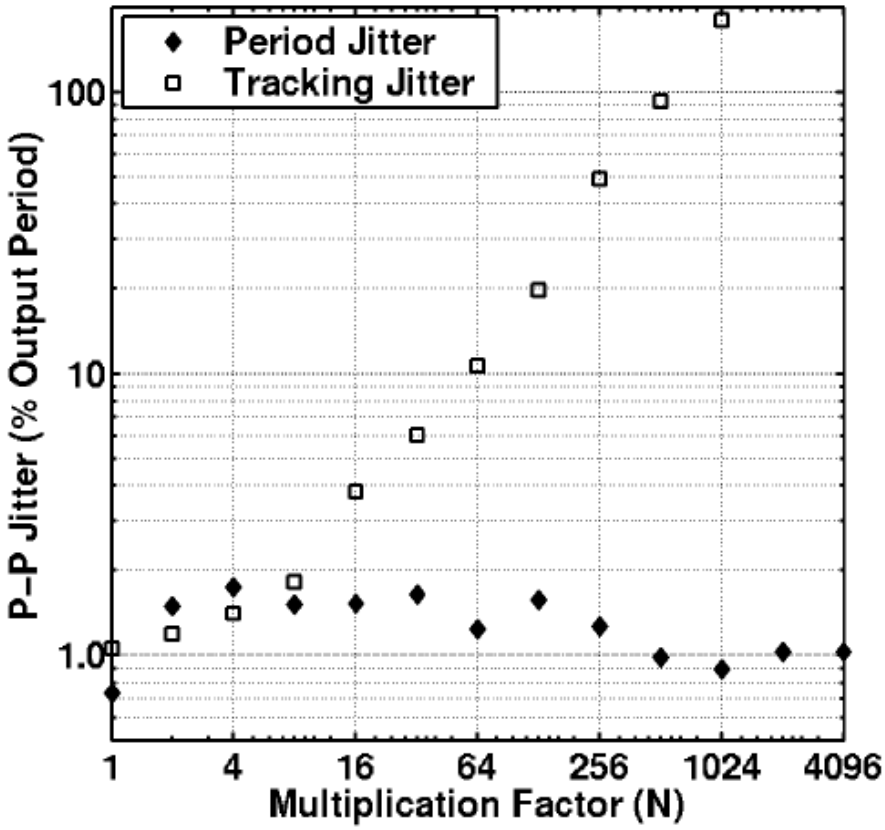


Fig. 1.2 Jitter versus multiplication factor at fixed 240MHz output [5]

Another problem in the conventional PLLs is the difficulty of the phase tracking. Fig. 1.2 shows the impact of the multiplication factor versus the period jitter and tracking jitter in the conventional PLL architecture [5]. The period jitter represents the variations of the output clock period, and the tracking jitter represents the phase error between the reference clock and output clock. In Fig. 1.2, the relation between the peak-to-peak period jitter and the frequency multiplication factor is not obvious. Regardless of how the frequency multiplication factor is (from 1 to 4096), the period

jitter is controlled less than 2% of the output clock period. On the contrary, in the conventional PLL architecture when the frequency multiplication factor is greater than 512, the peak-to-peak tracking jitter has been achieved 100% of the output clock period. In other words, the conventional PLL architecture is not suitable for the phase tracking in high frequency multiplication factor applications. One reason is that the conventional PLL architecture has ineffective phase tracking ability in high frequency multiplication factor applications. Another reason is, in such applications, most of the reference clocks have low frequency about kHz, and the loop refresh rate of the analog PLL is too low, so it will cause the leakage problem.

1.3 Video Display System

1.3.1 Video Display System Overview

The proposed high frequency multiplication factor All-Digital Phase-Locked Loop (ADPLL) is applied to the current video display system. The reason for selecting the video display system as the ADPLL application is because that this application has many requirements such as the high frequency multiplication factor and the ability of the phase tracking. In currently applications, the video display system is the most demanding, the following have a brief introduction.

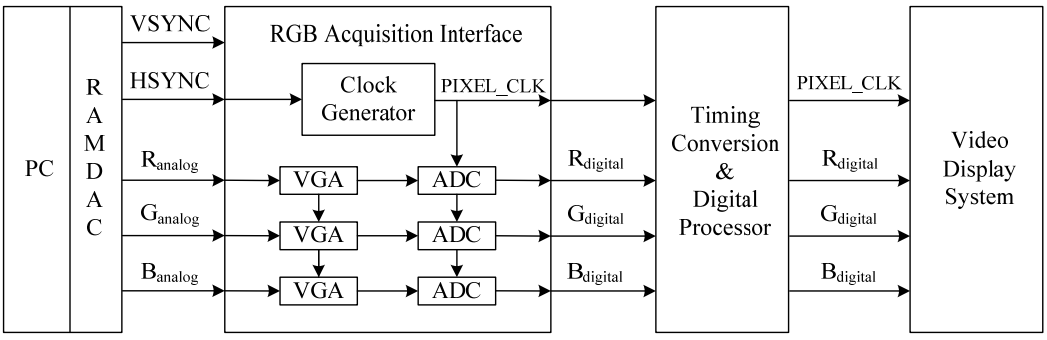


Fig. 1.3 Video display system

Fig. 1.3 shows the simplified block diagram of the video display system. The analog video signals RGB (Red/Green/Blue), vertical synchronous (VSYNC) and horizontal synchronous (HSYNC) signals from the Random Access Memory Digital-to-Analog Converter (RAMDAC) of the Personal Computer (PC) graphics card are delivered to the RGB acquisition interface. The RGB acquisition interface converts the analog video signals (RGB) into digital signals by variable gain amplifier (VGA) and analog-to-digital converter (ADC). Then the digital signals from digital processor are sent to video display system. The sampling clock (PIXEL_CLK) of ADC is generated by the clock generator. In general, the clock generator is implemented by the PLL. The clock generator according to the resolution of the display system uses HSYNC signal as reference clock to generate high speed pixel clock (PIXEL_CLK). The horizontal resolution is proportional to the frequency multiplication factor. Fig. 1.4 shows HSYNC and pixel clock timing diagram.

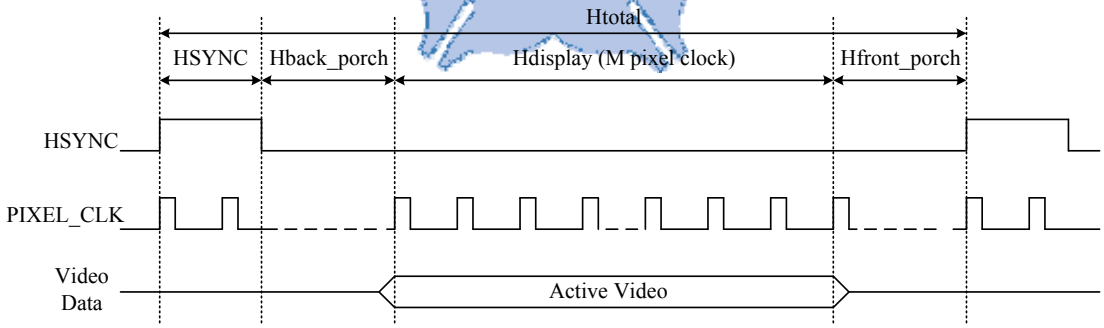


Fig. 1.4 HSYNC and pixel clock timing diagram

Table 1.1 Monitor Timing Specifications

Mode	Resolution		Refresh Rate	Horizontal Frequency	Pixel Frequency
	Active	Total			
XGA	1024×768	1344×806	60 Hz	48.4 kHz	65.000 MHz
		1328×806	70 Hz	56.5 kHz	75.000 MHz
		1312×800	75 Hz	60.0 kHz	78.750 MHz
		1376×808	85 Hz	68.7 kHz	94.500 MHz
SXGA	1280×1024	1688×1066	60 Hz	64.0 kHz	108.000 MHz
		1688×1066	75 Hz	80.0 kHz	135.000 MHz
		1728×1072	85 Hz	91.1 kHz	157.500 MHz
UXGA	1600×1200	2160×1250	60 Hz	75.0 kHz	162.000 MHz
		2160×1250	65 Hz	81.3 kHz	175.500 MHz
		2160×1250	70 Hz	87.5 kHz	189.000 MHz
		2160×1250	75 Hz	93.8 kHz	202.500 MHz
		2160×1250	85 Hz	106.3 kHz	229.500 MHz
WUXGA	1920×1200	2080×1235	60 Hz	74.0 kHz	154.000 MHz
		2592×1245	60 Hz	74.6 kHz	193.250 MHz
		2608×1255	75 Hz	94.0 kHz	245.250 MHz
		2624×1262	85 Hz	107.2 kHz	281.250 MHz

Video Electronics Standards Association (VESA) [6] defines the monitor timing specification and the detailed information is listed in Table 1.1. In video display system the higher monitor resolution, the higher monitor quality. For example, in WUXGA mode, the reference clock frequency is 74.556kHz, the pixel clock frequency is 193.250MHz, so the frequency of the pixel clock is up to 2592 times higher than the frequency of the reference clock. Therefore, the frequency multiplication factor of the clock generator is 2592. The high speed pixel clock generated by the clock generator has to align the phase of HSYNC signal, otherwise the video signals will be distorted after ADC sampling. Fig. 1.5 shows the relation between the RGB signals and the phase of the pixel clock. The valid sampling interval must be in the stable region of RGB analog signals, otherwise the captured video signals by the ADC will be wrong. Therefore, the clock generator has to accurately

tune the frequency of the pixel clock, and it must reduce the phase error between the pixel clock and HSYNC. In general video display application, the specifications of the phase error must be less than one third of the pixel clock period [6-10]. Otherwise signals distortion will be very serious and the monitor will have a flickering phenomenon. From previous discussions, the conventional PLL architectures can not be directly applied to this video display application, because it can not achieve these requirements in the high frequency multiplication factor condition.

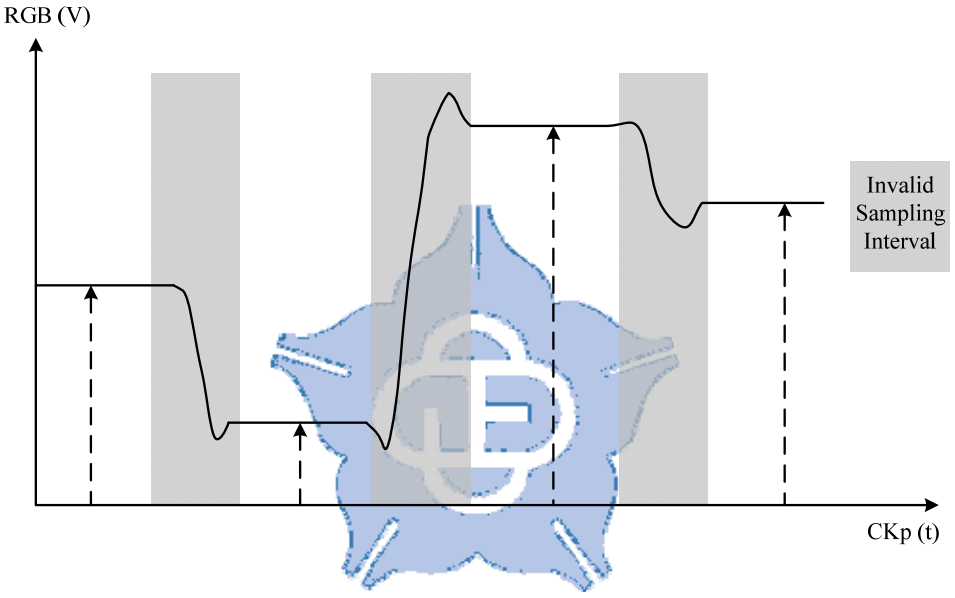


Fig. 1.5 RGB analog signal and pixel clock timing diagram

Table 1.2 HSYNC jitter measurement [9]

Pixel Clock Frequency		40MHz	160MHz	240MHz	320MHz
Radeon 8500	Effective Jitter (ns)	1.03nS	1.09nS	0.95nS	1.06nS
	Fraction of a pixel (%)	4.2%	17.3%	22.6%	33.5%
GeForce4	Effective Jitter (ns)	360pS	380pS	400pS	450pS
	Fraction of a pixel (%)	1.4%	6.1%	9.6%	14.1%
Parhelia-512	Effective Jitter (ns)	160pS	120pS	90pS	110pS
	Fraction of a pixel (%)	0.6%	2.0%	2.1%	3.5%

Table 1.2 shows the HSYNC jitter measurement results of several Personal Computer (PC) graphics cards measured by the UltraSharp Display Output Technology [9]. From Table 1.2, the HSYNC jitter of video display system may be as high as 1.06 ns and it is about 33.5% of the pixel clock period. Therefore, the conventional PLLs become more difficult to track the phase error.

1.3.2 The Difficulties of High Multiplication Factor ADPLL Design

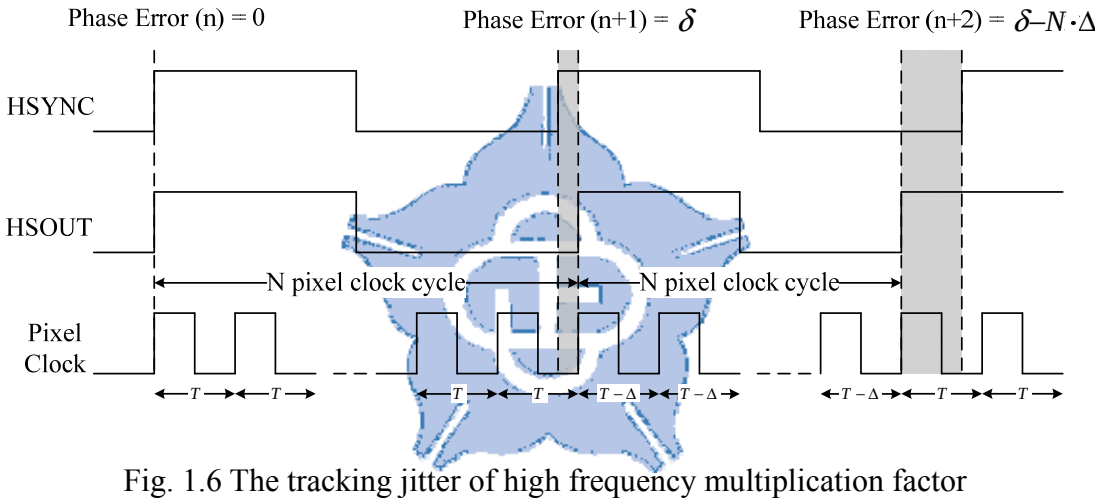


Fig. 1.6 The tracking jitter of high frequency multiplication factor

In the current video display system, the frequency multiplication factor of the clock generator is up to 2592. In this high frequency multiplication factor applications, any output frequency error and the reference clock jitter will cause enormous phase error accumulation. Fig. 1.6 explains the tracking jitter problem in high frequency multiplication factor PLL. If we assume that the frequency multiplication factor is N, the resolution of DCO is Δ, and the original output pixel clock period is T. In the beginning, the phase error between the HSYNC and HSOUT is zero. After one HSYNC clock, the phase of HSYNC leads the phase of HSOUT slightly, and the accumulative phase error is δ. Then the controller tunes the DCO period from T to T-

Δ to speed up the PLL frequency. Because the frequency multiplication factor N is a large number, so after one HSYNC clock the phase error will accumulate to δ-N · Δ. The phase error can not be reduced effectively and its amount is bigger than previous clock cycle. For example, if we assume that in WUXGA mode and the resolution of the DCO is 1ps, the frequency multiplication factor is 2592. After tuning the DCO step, the phase error will accumulate up to 2.592ns (=2592 · 1ps). Therefore, the conventional PLLs are not possible to have good performance in phase tracking in high frequency multiplication factor applications.

1.3.3 The Impact of HSYNC Jitter Injection

Because of the reference clock frequency is too low (31.5kHz to 106.3kHz), and the frequency multiplication factor is high, the PLL controller has to slightly adjust the DCO frequency, otherwise the phase error will be enlarged by the frequency multiplication factor N. In conventional PLLs, in order to stabilize the PLL loop, the step of the DCO has to be reduced. However, the reference clock (HSYNC) is not stable, and the period jitter of the reference clock period will up to 1.06ns [9]. When the reference clock has large jitter, the PLL has to track the phase error. However the DCO step is small after PLL is locked, so the phase tracking behavior will be slow. It causes the phase error can't be reduced less than one third of the pixel clock period [6-10]. Therefore, the PLL has to solve this problem when the reference clock has large jitter, otherwise the phase error can't meet the specification requirements.

1.4 Conventional PLLs Survey

1.4.1 A Fractural-DLL Based Clock Generator for Video Application

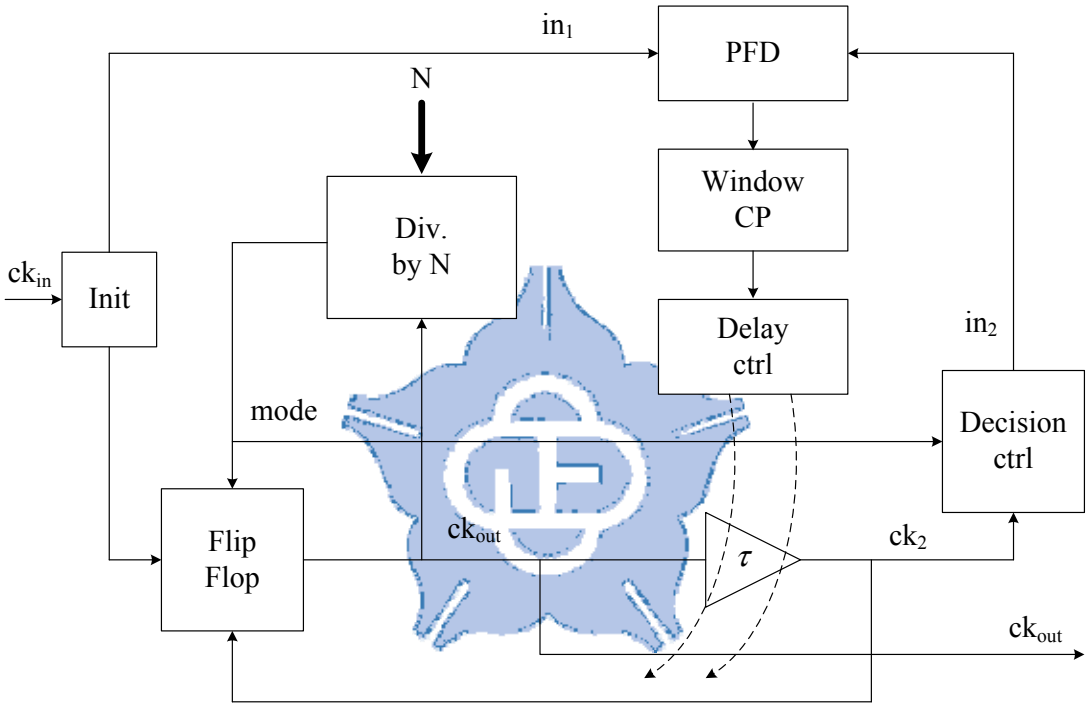


Fig. 1.7 Fractural-DLL based clock generator [7]

The fractural-DLL based clock generator is proposed by [7]. Fig. 1.7 shows the architecture of the fractural-DLL based clock generator. This clock generator uses analog phase and frequency detector (PFD), charge pump, and set/reset flip-flop to adjust the delay of delay cells. In the positive edge of the reference clock, the phase of high frequency reference clock is calibrated to avoid accumulating the phase error rapidly. The drawback of this architecture is that it needs a calibration circuit. Because when the reference clock has large jitter, the pixel clock perhaps may encounters with

glitch problem and the system will work incorrectly. Therefore, this architecture can't apply to video display system with large input jitter condition.

1.4.2 Video Capture PLL by Analog Bits Inc.

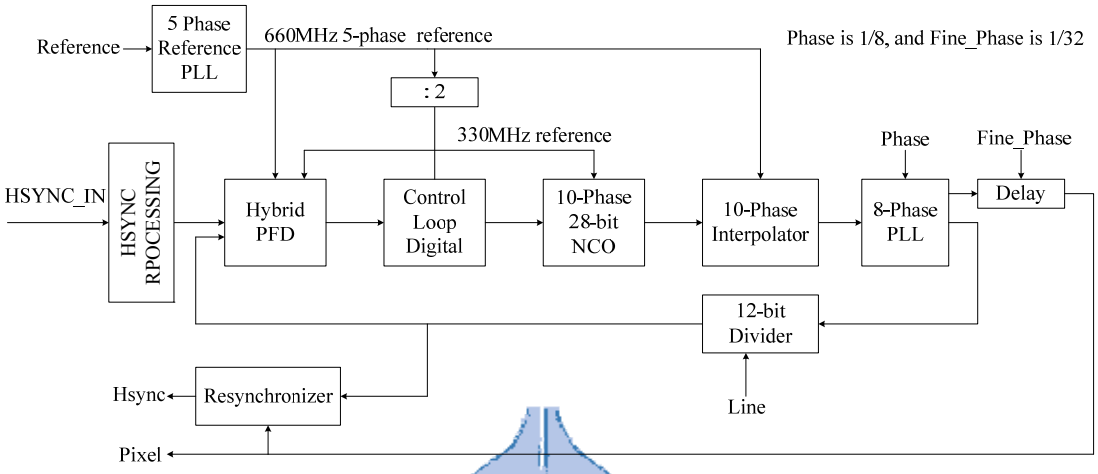


Fig. 1.8 Video capture PLL proposed by Analog Bits Inc. [8]

Fig. 1.8 shows the video capture PLL proposed by Analog Bits Inc. [8]. This video capture PLL uses three PLLs as clock generator. The 5-phase reference PLL uses a stable external reference clock (crystal at 14.3MHz) to generate a 660MHz 5-phase high speed clock. Then, this architecture uses 660MHz 5-phase high speed clock to control the 10-phase 28-bit NCO (Numerically Controlled Oscillator) to achieve phase tracking and frequency multiplication. In this architecture, it needs a GHz clock as the sampling clock. Because in this video display system, the requirement of the frequency is up to 230MHz in UXGA mode. In [8], it uses the multi-phase clock generator to avoid generating the GHz clock. However, this architecture requires a stable external oscillator or crystal, and three high speed PLLs will increase chip area and power consumption.

1.4.3 An ADPLL for Video Pixel Clock Generator

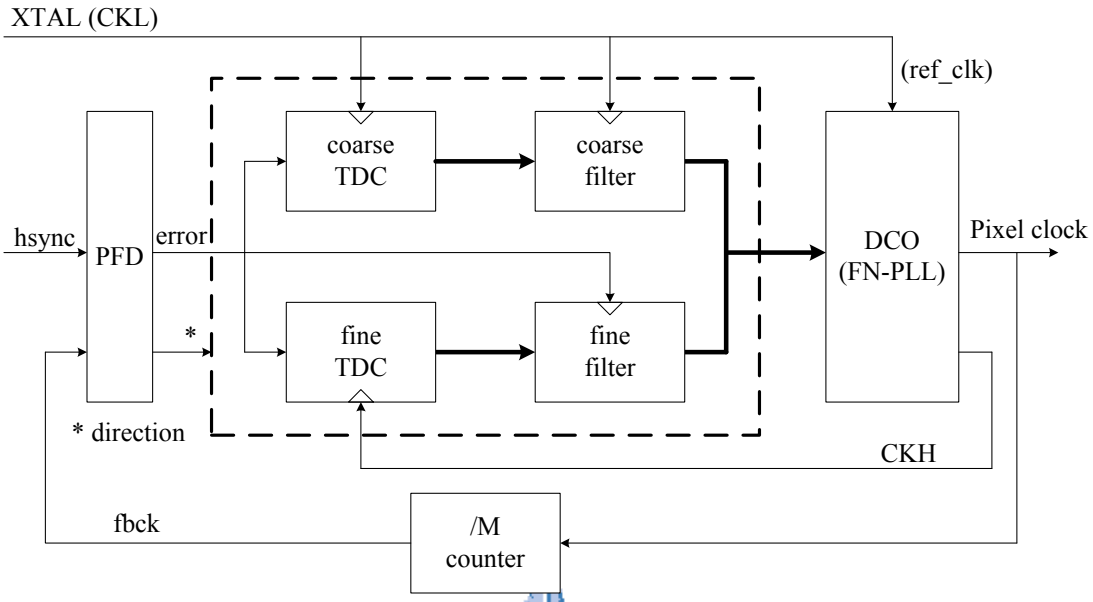


Fig. 1.9 Video pixel clock generator [10]

The video pixel clock generator is proposed by [10]. Fig. 1.9 shows the architecture of video pixel clock generator. From Fig. 1.9 the system has two loops, one is in the fractional-N DCO component and the other is in the feedback path of the pixel clock. Therefore, this architecture has to add additional circuit to make loop consistency. The fractional-N DCO component is composed of the PFD, charge pump, VCO, and the fractional divider circuit. The fractional-N DCO architecture is equivalent to analog charge pump based PLL. Due to the VCO in the fractional-N DCO, so it will also have the same leakage problem in advanced process. However, the fractional-N DCO also needs a stable external reference clock. From the above discussions, the video pixel clock generator is not suitable for video display system in advanced process.

1.5 Summary

Due to the design challenges in conventional PLLs, in recently, the All-Digital Phase-Locked Loops (ADPLLs) [11-16] have proposed to overcome the above problems. The feature of the ADPLL is that all digital control circuit. Therefore, when the system uses the digital controlled oscillator (DCO) to replace the voltage controlled oscillator (VCO), the leakage problem can be solved. For wide frequency range, the DCO uses the cascaded structure and all-digital controller, so the problem of the wide frequency range can also be solved. From the above discussions, the ADPLL will become more competitive and more essential in advanced process.

For the difficulties of high multiplication factor ADPLL design has discussed in section 1.3.2. In the proposed ADPLL design, we will use first-order sigma-delta modulator (SDM) to improve the equivalent DCO resolution for reducing the extensive phase error between HSYNC and HSOUT.

For the impact of HSYNC jitter injection has discussed in section 1.3.3. In the proposed ADPLL design, we will use the time-to-digital converter (TDC) in the proposed ADPLL to compensate the phase error caused by the HSYNC jitter. The TDC circuit will improve the overall performance, and also improve the phase tracking ability of the proposed ADPLL.

1.6 Thesis Organization

In this dissertation, we will design a fast phase tracking and high frequency multiplication ADPLL in 65nm CMOS process.

In chapter 2, all the details of the proposed ADPLL clock generator, including the circuit architecture, and circuit techniques are presented. In chapter 3, we show the

experimental results of the proposed ADPLL and the chip implementation. Finally, we make conclusions and point out future works in chapter 4.

Chapter 2 Architecture of Video Capture ADPLL

2.1 System Architecture Overview

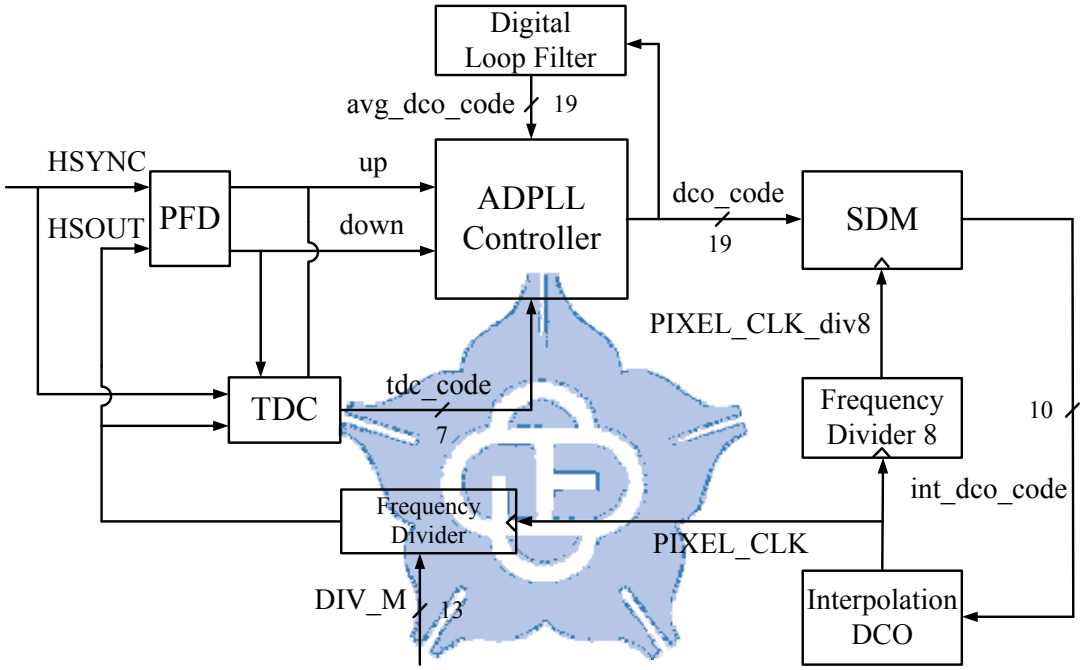


Fig. 2.1 The block diagram of proposed ADPLL architecture

Fig. 2.1 shows the block diagram of proposed ADPLL architecture. The proposed ADPLL is composed of seven blocks: Phase Frequency Detector (PFD), Time-to-Digital Converter (TDC), Sigma-Delta Modulator (SDM), Interpolation Digital Controlled Oscillator (DCO), ADPLL Controller, Digital Loop Filter, and Frequency Divider.

The working principle of proposed ADPLL is described as follows. The signals of HSYNC, RESET, and DIV_M are system inputs. The DIV_M signal is according

to the specification of video display system to decide the frequency multiplication factor. Both signals HSYNC and HSOUT are sent to the PFD, then the PFD compares which signal is leading or lagging, and then it generates Up and Down information. Simultaneously, the TDC uses up and down information to quantify the phase error and generate the TDC code (tdc_code). According to up and down information, the ADPLL controller adjusts the DCO output frequency to reduce the phase error between HSYNC and HSOUT signals to achieve target frequency. The Sigma-Delta Modulator (SDM) block is added to enhance the equivalent resolution of the DCO. Therefore, the control code (dco_code) from the ADPLL controller is sent into the SDM as its control signal. Then the SDM generates control signals (int_dco_code) to control the Interpolation DCO. HSOUT is the output signal of the Frequency Divider.

The rest of block structures are organized as follows. Section 2.2 describes the structure of the bang-bang PFD. Section 2.3 describes the DCO structure and the solution of the non-monotonic DCO. Section 2.4 and Section 2.5 describes the ADPLL controller and the system finite state machine. Then, section 2.6 discusses how to use dithering technology to improve the equivalent DCO resolution and its performance. Finally, Section 2.7 describes the structure of the time-to-digital converter (TDC) and how to use it to quantize the phase error between both HSYNC and HSOUT signals.

2.2 Phase and Frequency Detector

2.2.1 Structure

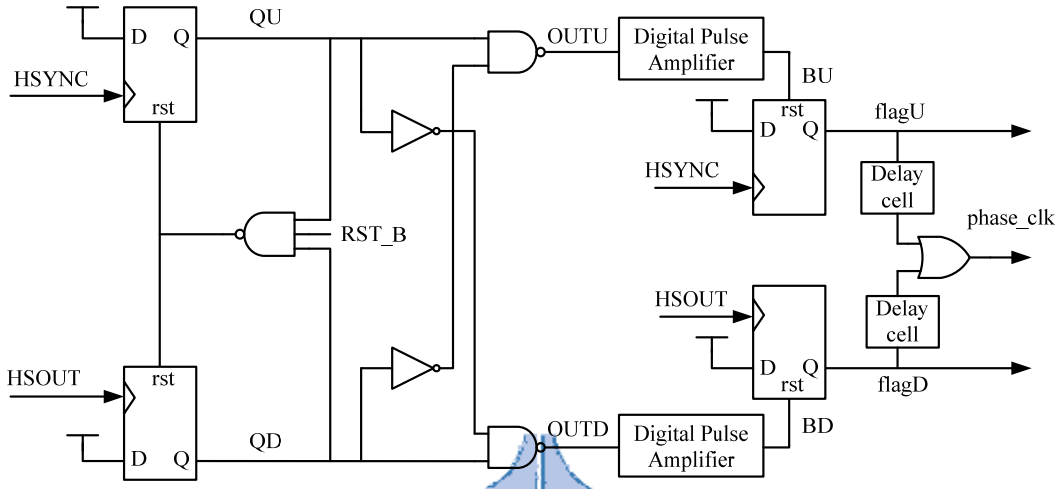


Fig. 2.2 The cell-based, three-state, bang-bang PFD architecture [12]

The phase and frequency detector (PFD) is used to detect the phase error and the frequency error. The three-state bang-bang PFD [12] is used in the proposed ADPLL which is the cell-based design. Fig. 2.2 shows the bang-bang PFD architecture. The bang-bang PFD has three operation conditions. When HSOUT leads HSYNC, a low pulse is generated at flagD. On the contrary, when HSOUT lags HSYNC, a low pulse is generated at flagU. And the last operation is, when HSOUT falls into the dead zone of the PFD, both flagU and flagD signals remain at high logic level. The dead zone means the dead region of the PFD. That is the phase error can't be distinguished between HSYNC and HSOUT.

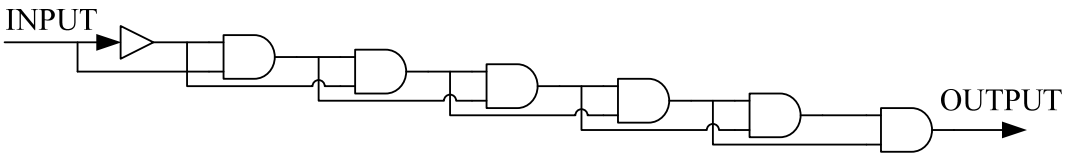


Fig. 2.3 Digital pulse amplifier architecture [12]

Fig. 2.3 shows the digital pulse amplifier [12] architecture. The digital pulse amplifier is used to reduce the dead zone of the PFD. When the input low pulse signal send into the digital pulse amplifier, the output signal whose pulse width will be increased. It's to meet the minimum pulse width requirement of the D-Flip/Flop's reset pin. In our work, we use both output signals flagU and flagD from the PFD to generate a new signal named phase_clk. The phase_clk signal is used as the reference clock for the ADPLL controller block.

2.2.2 Simulation Result

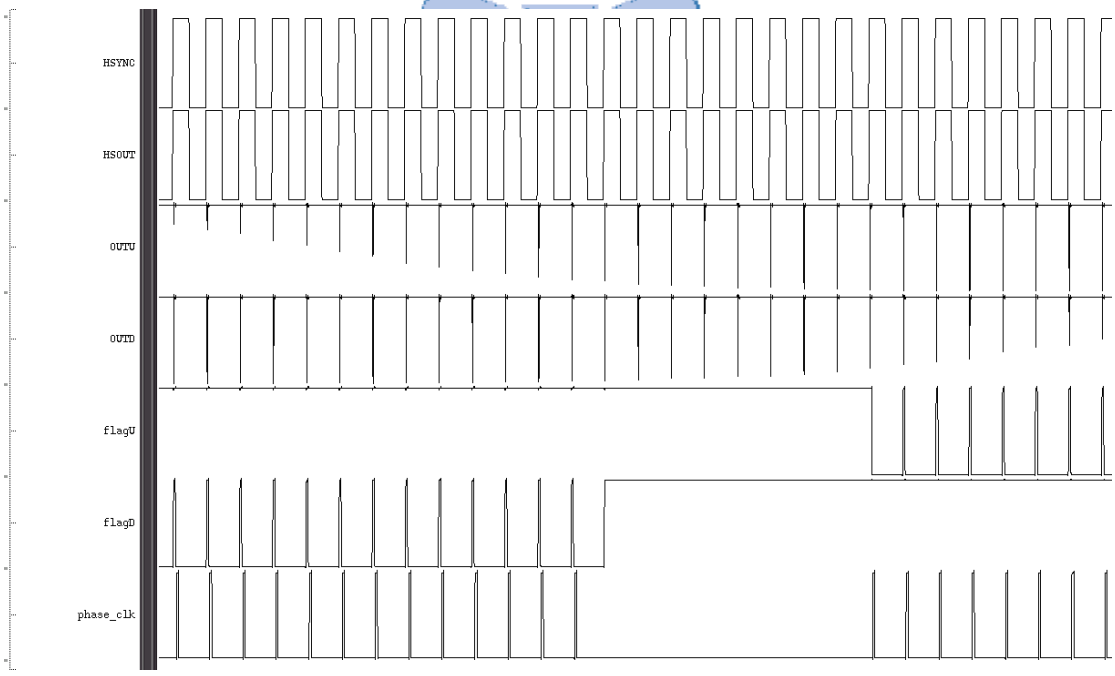


Fig. 2.4 Simulation result of the bang-bang PFD

Table 2.1 The dead zone of the bang-bang PFD

	Typical Case	Fast Case	Worst Case
Dead zone value (ps)	4	3	8

Fig. 2.4 shows the simulation result of the bang-bang PFD by UltraSIM simulator. It is simulated on UltraSIM SPICE mode at worst case. In order to measure the PFD dead zone under different PVT variations, the simulation switches the phase error from HSYNC leading HSOUT for 15ps to HSYNC lagging HSOUT for 15ps. Table 2.1 shows the PFD dead zone under different PVT variations.

2.3 Digital Controlled Oscillator

The digital controlled oscillator (DCO) is the most critical component in the all-digital phase-locked loop (ADPLL). Because the DCO usually occupies almost 50% area and power consumption of the ADPLL, and therefore how to design a DCO with lower power, smaller area and sufficient frequency resolution is very important while designing an ADPLL.

2.3.1 MUX-Type DCO structure

2.3.1.1 Structure

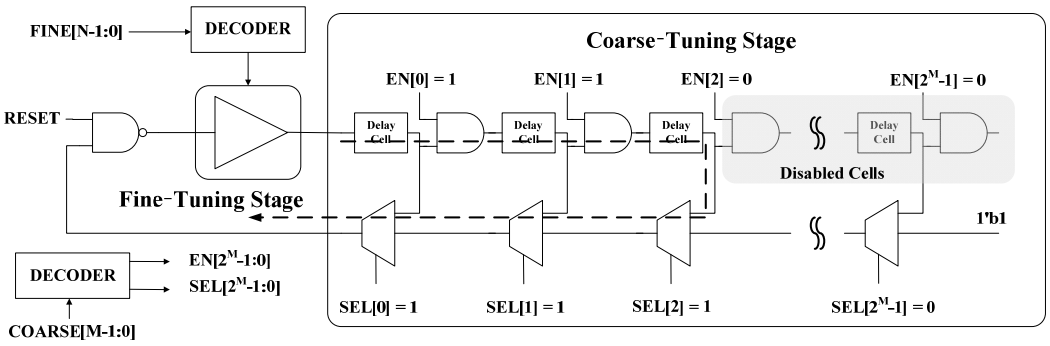


Fig. 2.5 The coarse-tuning stage of the MUX-type DCO.

Fig. 2.5 shows the architecture of the MUX-type DCO. The MUX-type DCO is composed of the coarse-tuning stage and the fine-tuning stage. The coarse-tuning stage which has (2^M-1) delay cells with (2^M-1) multiplexers can provide 2^M different delays. In order to generate a sufficient delay time in 65nm CMOS process, the delay cells which with larger MOS channel length in the cell-library are used to build up the coarse-tuning stage. And the two-input AND gates are added to each delay cell's output to disable the unused cells to save power consumption.

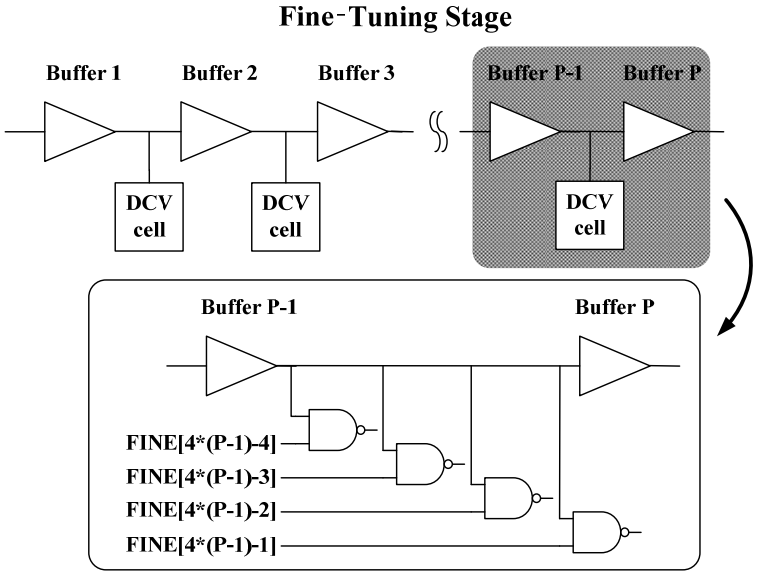


Fig. 2.6 The fine-tuning stage of the proposed DCO.

Fig. 2.6 shows the fine-tuning stage of the DCO. To achieve better DCO resolution, the digital-controlled varactors (DCVs) [11][20][24][26] are used in the fine-tuning stage. The fine-tuning stage has P buffers, in each buffer it connects to four NAND gates. When the fine-tuning control code (FINE[4*(P-1)-1:0]) is changed, the capacitance in the buffer's output node is also changed. Therefore a high resolution, linear fine-tuning delay stage can be created.

2.3.1.2 Problem of Non-monotonic DCO

In order to achieve both wide frequency range and high resolution with smaller chip area and lower power consumption, the cascaded structure is often used in designing the DCO [11][20][24][26]. In these DCOs, the coarse-tuning stage, which uses large delay cells to achieve wide-range delay control, is accompanied with a fine-tuning stage to improve the resolution of the DCO. In this cascaded architecture, it is often needed to overlap the sub-frequency band to make sure that there will not have any frequency dead zone in the DCO. But this makes the DCO's output

frequencies become non-monotonic with the DCO control codes. To alleviate the difficulty to design the ADPLL controller with these cascaded structure DCOs, the fine-tuning stage must have a delay controllable range larger than the delay step of previous coarse-tuning stage. However, it means that the coarse-tuning DCO control code must be determined in the frequency search mode, and it must be fixed after the frequency search is done. Then the ADPLL controller only adjusts the fine-tuning DCO control code to fine-tune the output frequency and to track the phase of the reference clock in a selected sub-frequency band.

In these ADPLLs [11][20][24][26], the proposed DCOs still have monotonic response if the coarse-tuning DCO control code is fixed while tuning the fine-tuning DCO control code. But for high frequency multiplication applications, such as line-locked PLLs or spread spectrum clock generator (SSCG) applications [23][26], it often needs to change the coarse-tuning DCO control code after frequency search is done. However, when we switch the coarse-tuning DCO control code to the adjacent sub-frequency band, because there are overlapped region between adjacent sub-frequency bands, the output frequency will become non-monotonic with input DCO control code. Thus in [26], the auto-adjust algorithm is proposed to solve the non-monotonic problem during sub-frequency band transition in SSCG application.

But the proposed auto-adjust algorithm [26] depends on the simulation results with PVT variations to decide a fixed compensation code. However, the overlapped region between adjacent sub-frequency bands will be changed with PVT variations. And this fixed compensation code must be designed for the worst-case. As a result, if the overlapped region is smaller than expectation, it will affect the DCO resolution and the jitter performance.

Therefore, it is difficult to design a monotonic MUX-type DCO. Non-monotonic or large resolution would take place and result in unstable loop tracking as shown in Fig. 2.7.

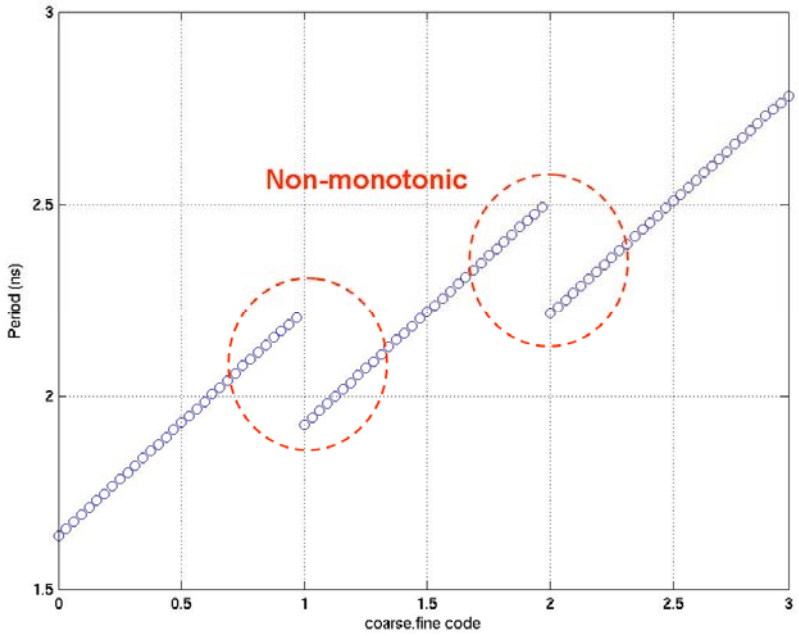


Fig. 2.7 Non-monotonic DCO

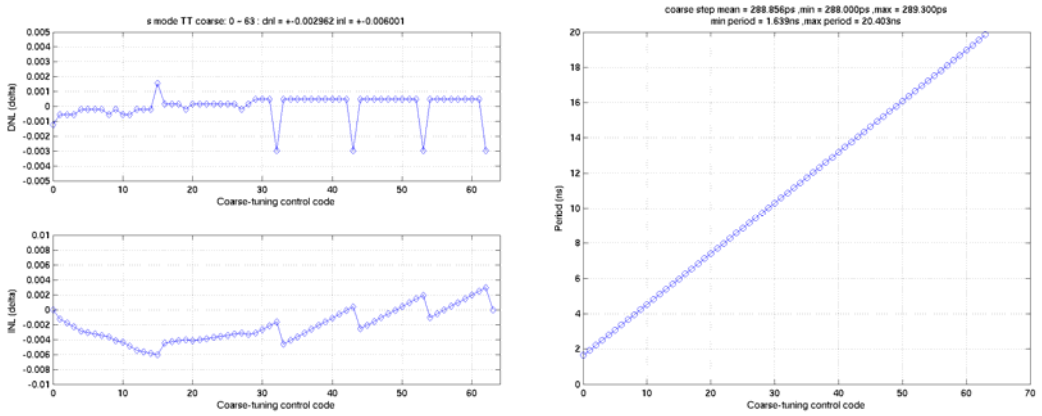
In order to solve the non-monotonic issue, the cell-based DCO with built-in self-calibration circuit (BISC) is describing in detail in section 2.3.2.

2.3.1.3 Simulation Result

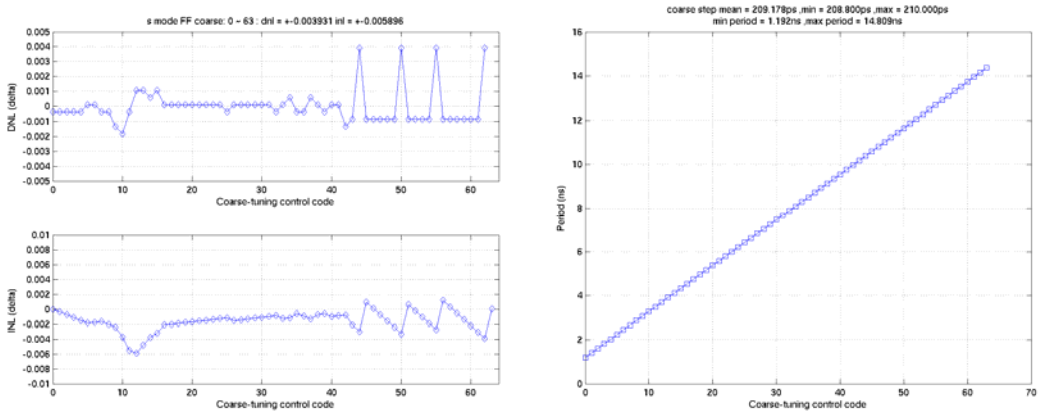
The MUX-type DCO is simulated with HSPICE. Fig. 2.8 shows the period of the DCO output clock versus coarse-tuning stage control code (0-63), when the fine-tuning stage control code is set to zero, and shows INL and DNL of the MUX-type DCO.

The simulation parameters for each corner are process, voltage, and temperature, respectively. The circle represents the TT corner, 1.0 V, 25°C, the square represents the FF corner 1.1V, 0°C, and the upward-pointing triangle represents the SS corner, 0.9V, 125°C, respectively.

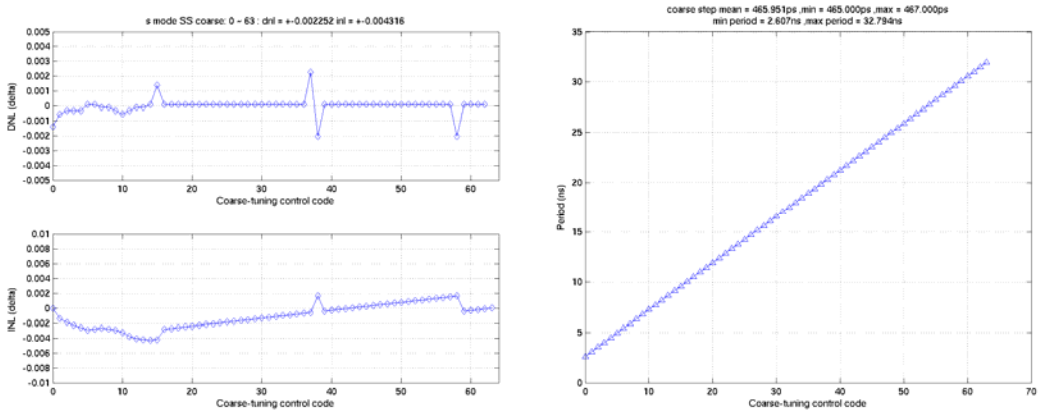
In TT corner, the DCO period range is from 1.639ns to 20.403ns, the DNL is ±0.002962Δ, and the INL is ±0.006001Δ. In FF corner, the DCO period range is from 1.192ns to 14.809ns, the DNL is ±0.003931Δ, and the INL is ±0.00589Δ. In SS corner, the DCO period range is from 2.607ns to 32.794ns , the DNL is ±0.002252Δ, and the INL is ±0.00431Δ. Three corners are shown in (a), (b), (c) , respectively.



(a) TT corner, DNL: ±0.002962Δ, INL: ±0.006001Δ, DCO range: 1.639ns ~ 20.403ns



(b) FF corner, DNL: ±0.003931Δ, INL: ±0.005896Δ, DCO range: 1.192ns ~ 14.809ns



(c) SS corner, DNL: ±0.002252Δ, INL: ±0.004316Δ, DCO range: 2.607ns ~ 32.794ns

Fig. 2.8 Simulation of MUX-type DCO period versus coarse-tuning code 0 ~ 63

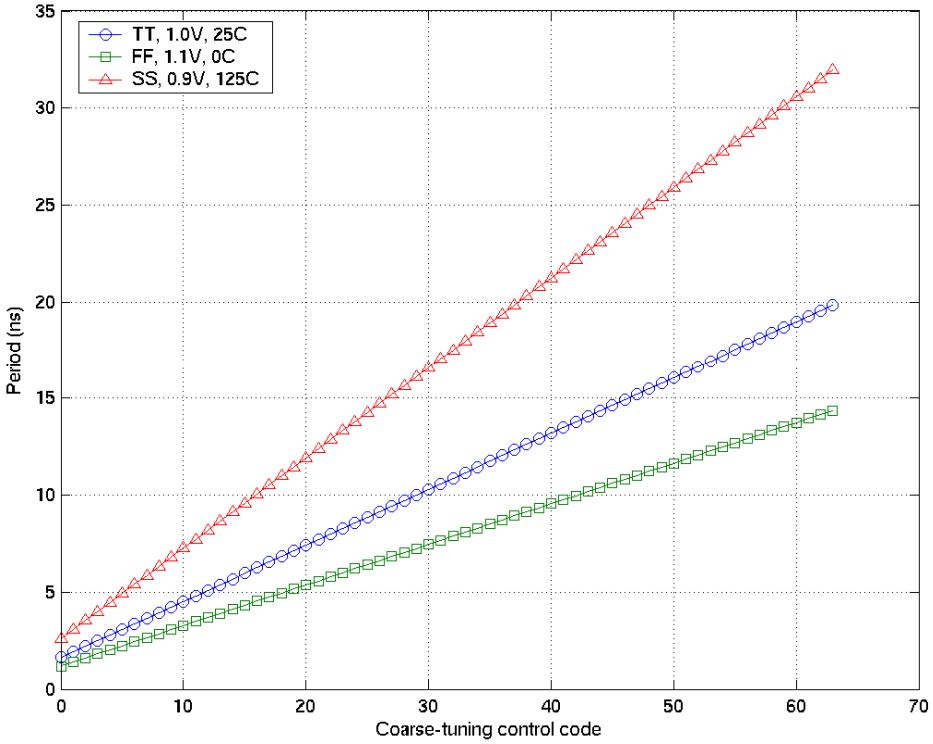


Fig. 2.9 Comparison of the MUX-type DCO period in PVT variations

Table 2.2 Properties of the MUX-type DCO coarse-tuning stage

ULTRASIM S mode Coarse-Tuning Stage Control Code : 0 ~ 63					
	Avg. Step (ps)	Max Step (ps)	Min Step (ps)	Max Period (ns)	Min Period (ns)
TT corner	288.856	289.300	288.000	20.403	1.6391
FF corner	209.178	210.000	208.800	14.809	1.192
SS corner	465.951	467.000	465.000	32.794	2.607

Fig. 2.9 shows the comparison of the MUX-type DCO period in PVT variations. Table 2.2 shows the properties of the MUX-type DCO coarse-tuning stage. The MUX-type DCO operation range is from 2.607ns to 14.809ns, covered in each corner.

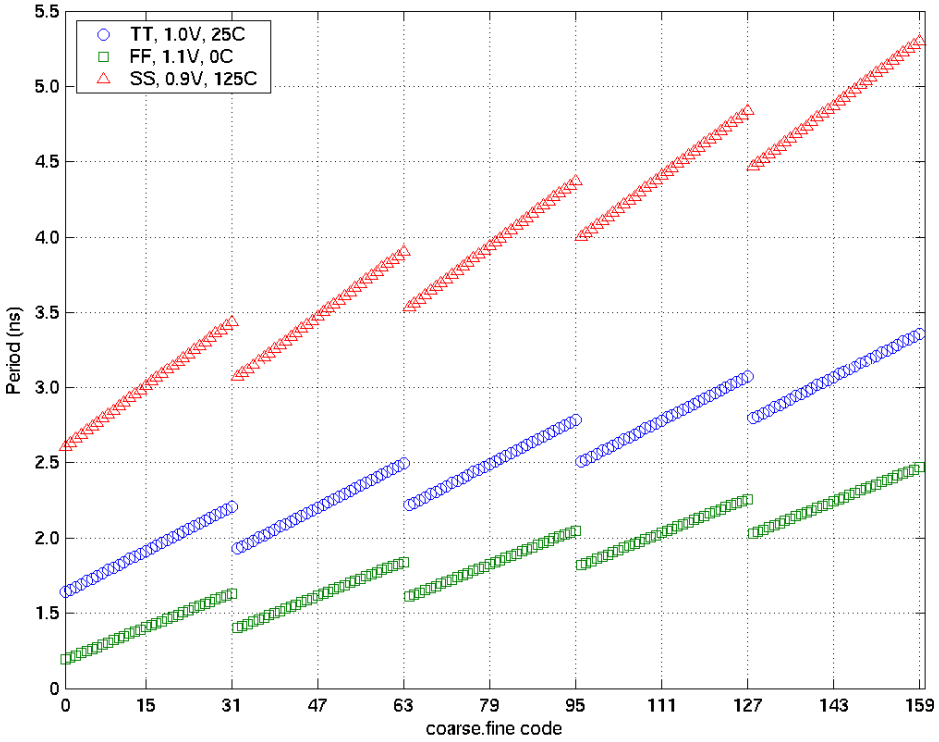


Fig. 2.10 Simulation of MUX-type DCO period

Table 2.3 Properties of the DCO fine-tuning stage

ULTRASIM S mode Fine-Tuning Stage Control Code : 0 ~ 31					
	Avg. Step (ps)	Max Step (ps)	Min Step (ps)	Range (ns)	Cover (ns)
TT corner	18.248	21.000	16.000	565.684	276.824
FF corner	14.175	16.000	12.000	439.439	230.268
SS corner	26.865	30.000	23.000	832.816	366.878

Fig. 2.10 shows the period of the MUX-type DCO in both coarse-tuning control code (0~63) and fine-tuning control code (0~31) under different PVT conditions. The average step of coarse-tuning stage delay is 288.856ps in TT corner, 209.178ps in FF corner, and 465.951ps in SS corner. The average step of fine-tuning stage delay is

18.248ps in TT corner, 14.175ps in FF corner, and 26.865ps in SS corner. The average range of fine-tuning stage delay is 565.684ps in TT corner, 439.439ps in FF corner, and 832.816ps in SS corner. The overlap delay is 276.824ps in TT corner, 230.268ps in FF corner, and 366.878ps in SS corner. Table 2.3 shows the properties of the DCO fine-tuning stage. We can see the range of fine-tuning stage is larger than one coarse-tuning stage delay step.

2.3.2 A Built-In Self-Calibration Circuit for Monotonic DCO

In this section, the cell-based DCO with built-in self-calibration (BISC) circuit to overcome the non-monotonic response problem in cascaded structure DCO is presented. The mechanism of self-calibration decides the compensation code for the DCO fine-tuning control codes when the coarse-tuning control codes are changed. The proposed self-calibration method can guarantee the monotonic response of the DCO, and therefore the advantages of using the cascaded structure DCOs can be retained.

2.3.2.1 Built-In Self-Calibration Circuit

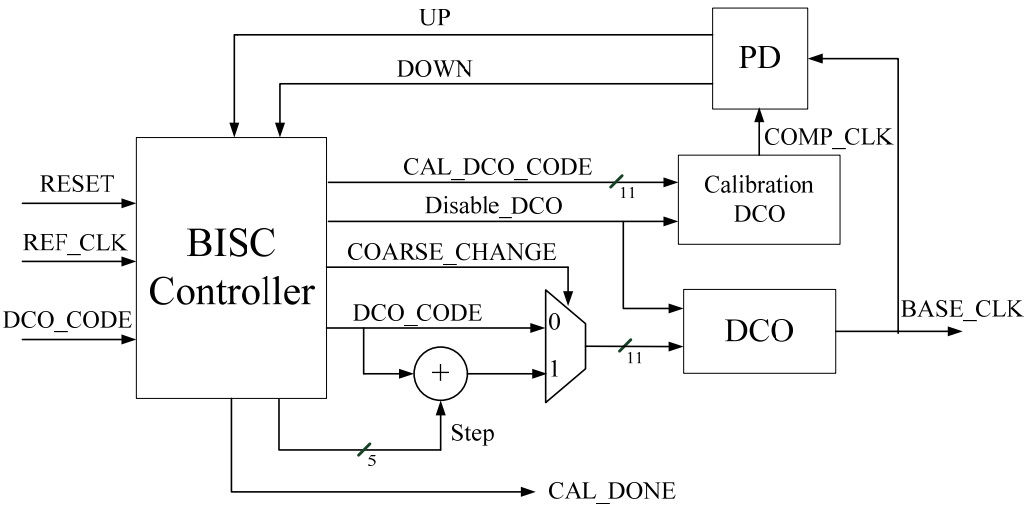


Fig. 2.11 The proposed DCO with built-in self-calibration circuit

Fig. 2.11 shows the architecture of the proposed DCO with BISC circuit. The DCO control code (DCO_CODE) which inputs to the DCO is sent to the BISC controller to detect if there has changes in the coarse-tuning control code. Then the compensation code (Step[4:0]) for DCO fine-tuning control is added to the current input DCO control code to make sure that the monotonic response of the DCO during coarse-tuning control code transitions.

In the cascaded structure DCOs [11][20][24][26], the DCO has the coarse-tuning stage and the fine-tuning stage. But in this architecture, it is often needed to overlap the sub-frequency band to make sure that there will not have any frequency dead zone in the DCO. Otherwise the output clock may have large cycle-to-cycle jitter while the DCO operates near the frequency dead zone.

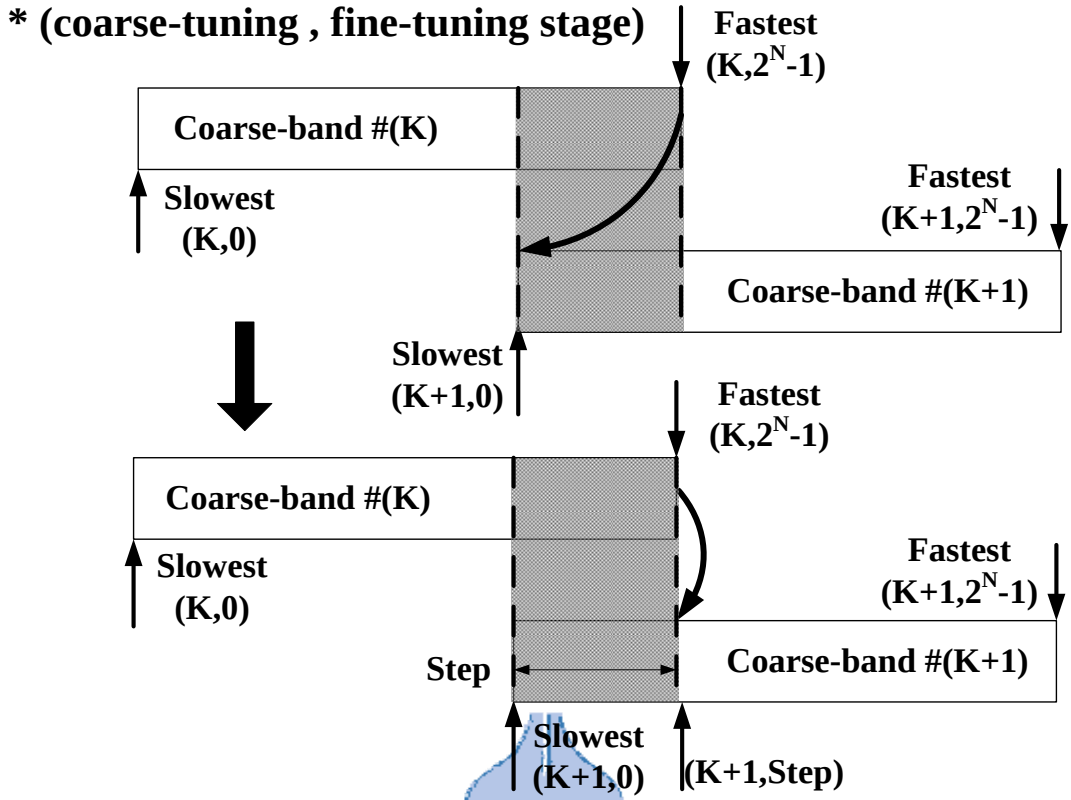


Fig. 2.12 The compensation code when sub-frequency band is changed

But if we overlap the sub-frequency band as shown in Fig. 2.12, it means that when the coarse-tuning DCO control code changes from the current code to the next coarse-tuning DCO control code, the output frequencies is not monotonically increasing. As a result, when the ADPLL controller adjusts the DCO control code from the coarse-band #(K) with fine-tuning control code (2^N-1) to the next coarse-band #(K+1), because the fine-tuning control code should reset to zero, and therefore the output frequency becomes slower than in previous DCO control code (i.e. coarse-band #(K) with fine-tuning control code (2^N-1)). And the ADPLL controller will encounter great difficulties in frequency tracking.

To avoid this phenomenon, compensation code should be added to the fine-tuning control code if there has changes in the coarse-tuning control code. In Fig. 2.12, a compensation code (Step[4:0]) is added to the fine-tuning control code so that the monotonic response can be still retained.

The compensation code (Step[4:0]) can be determined by circuit simulation with PVT variations. But if a fixed value compensation code is used in the ADPLL design, there will have too worse cycle-to-cycle jitter in worst-case conditions. In this work, we copied parts of the DCO circuit shown in Fig. 2.11 and named as "Calibration DCO" with the phase detector (PD), and the BISC controller to generate the compensation code (Step[4:0]) for current operating conditions. The calibration circuit starts to work when system is reset. After the calibration is done for the DCO, the compensation code is determined and then the ADPLL starts its normal operation.

The DCO control code is expressed in this format (coarse-tuning control code, fine-tuning control code). Two adjacent frequencies (K, 2^N-1) and (K+1, Step) are used to do frequency comparison, where the fine-tuning control code has N-bit. The DCO control code (K, 2^N-1) is applied to the DCO shown in Fig. 2.11 Then the DCO control codes (K+1, 0), (K+1, 1), … to (K+1, X) are sequentially applied to the "Calibration DCO". The phase detector detects if the frequency of the "Calibration DCO" is higher than the DCO. Thus after several calibration cycles, the compensation code (Step[4:0]) can be found to guarantee the monotonic response of the DCO in ADPLL normal operation mode.

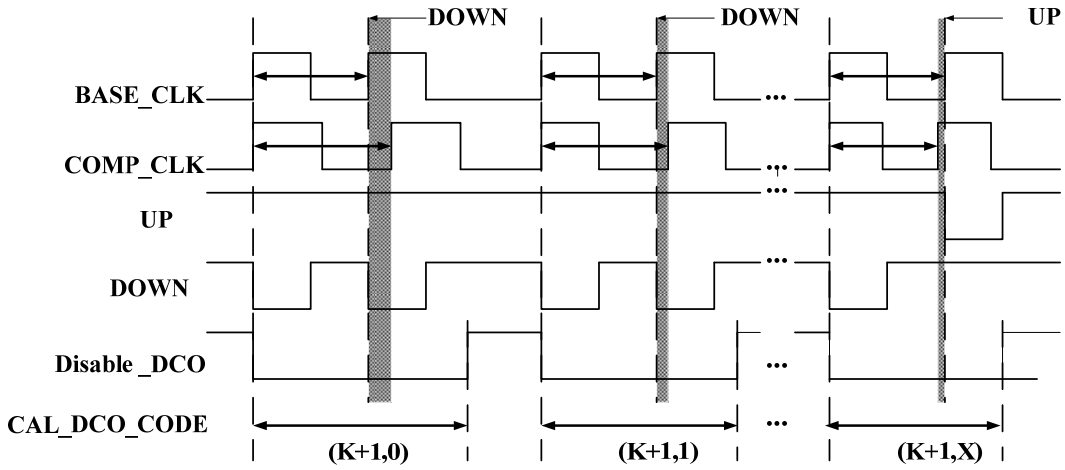


Fig. 2.13 The timing diagram of the proposed calibration circuit

The timing diagram of the proposed calibration circuit is shown in Fig. 2.13. In Fig. 2.13, the BASE_CLK is the output clock of the DCO circuit and the COMP_CLK is the output clock of the "Calibration DCO". The signal "Disable_DCO" is used to disable both the DCO and the "Calibration DCO" after each frequency comparison so that the phase detector can be used to perform frequency comparison. The BASE_CLK and the COMP_CLK are sent to the phase detector. The phase detector compares the phase of these two clocks. In the beginning of the calibration process, because there has overlapped sub-frequency bands in the DCO, the COMP_CLK is lagged to the BASE_CLK. Then the BISC Controller keeps increasing the fine-tuning DCO control code of "Calibration DCO" until the COMP_CLK leads the BASE_CLK. Then the value X shown in Fig. 2.13 is saved as fine-tuning compensation code (Step[4:0]).

After the calibration process is finished, the ADPLL returns to its normal mode. And the compensation code for DCO fine-tuning control code is added to current input DCO control code (DCO_CODE) to make sure that the monotonic response of

the DCO during coarse-tuning control code transitions. And if there has no change in the coarse-tuning control code, the input DCO control code is bypassed to the DCO.

2.3.2.2 Test Chip Architecture

The architecture of the proposed DCO in the test chip is mentioned in the previous section 2.2.1.1.

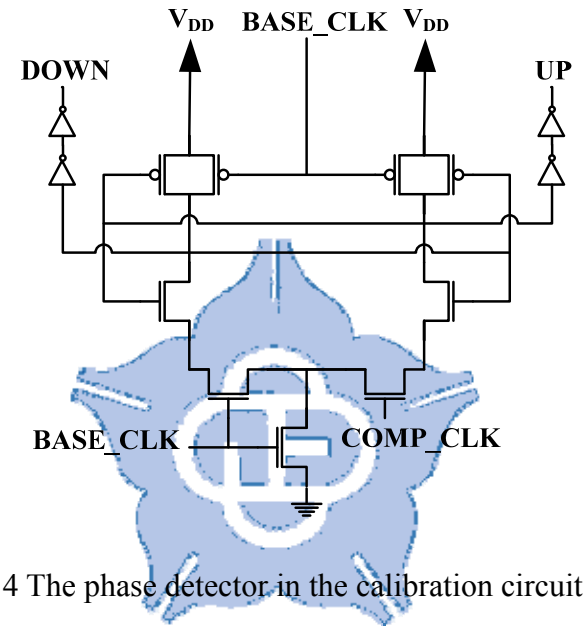


Fig. 2.14 The phase detector in the calibration circuit [24]

Fig. 2.14 shows the schematic of the phase detector [24] used in the calibration circuit. The principle of the phase detector is to determine which rising edge in the BASE_CLK or the COMP_CLK occurs later. This phase detector has a dead zone about 1ps in 65nm CMOS process which is sufficient to detect tiny frequency difference in frequency comparison. In this work, two additional inverters are added at the output port of the phase detector to increase the driving capacity.

The other circuits such as the BISC controller are written with hardware description language (HDL), and then the cell-based design flow is used to implement the full test chip.

2.3.2.3 Experimental Result

Table 2.4 shows the I/O PADs description of the proposed test chip, the 22 I/O PADs and 10 power PADs are used in this test chip.

Table 2.4 I/O PADs description

Input	Bits	Function	
REST	1	set chip to initial	
CLK	1	input reference clock	
DIV_M	1	divider multiplication factor	
COARSE_CODE	4	DCO Coarse-tuning stage control code	
FINE_CODE	5	DCO Fine-tuning stage control code	
STEP_SEL	1	Set the output step	
		Value	Step
		0	calibration up step
		1	calibration down step
MODE	2	Set the built-in self-calibration mode	
		Value	DCO Running Mode
		0	DCO auto upward running
		1	DCO auto downward running
		2	DCO fixed code running
Output	Bits	Function	
OUT_CLK	1	DCO output clock	
INIT_RUN_DCO_DOEN	1	BIST finish signal	
STEP	5	compensation code	
Power Pad	Pairs	Function	
VDDC+VSSC	1	CORE Power Pad	
VDDP+VSSP	4	Pad Power Pad	

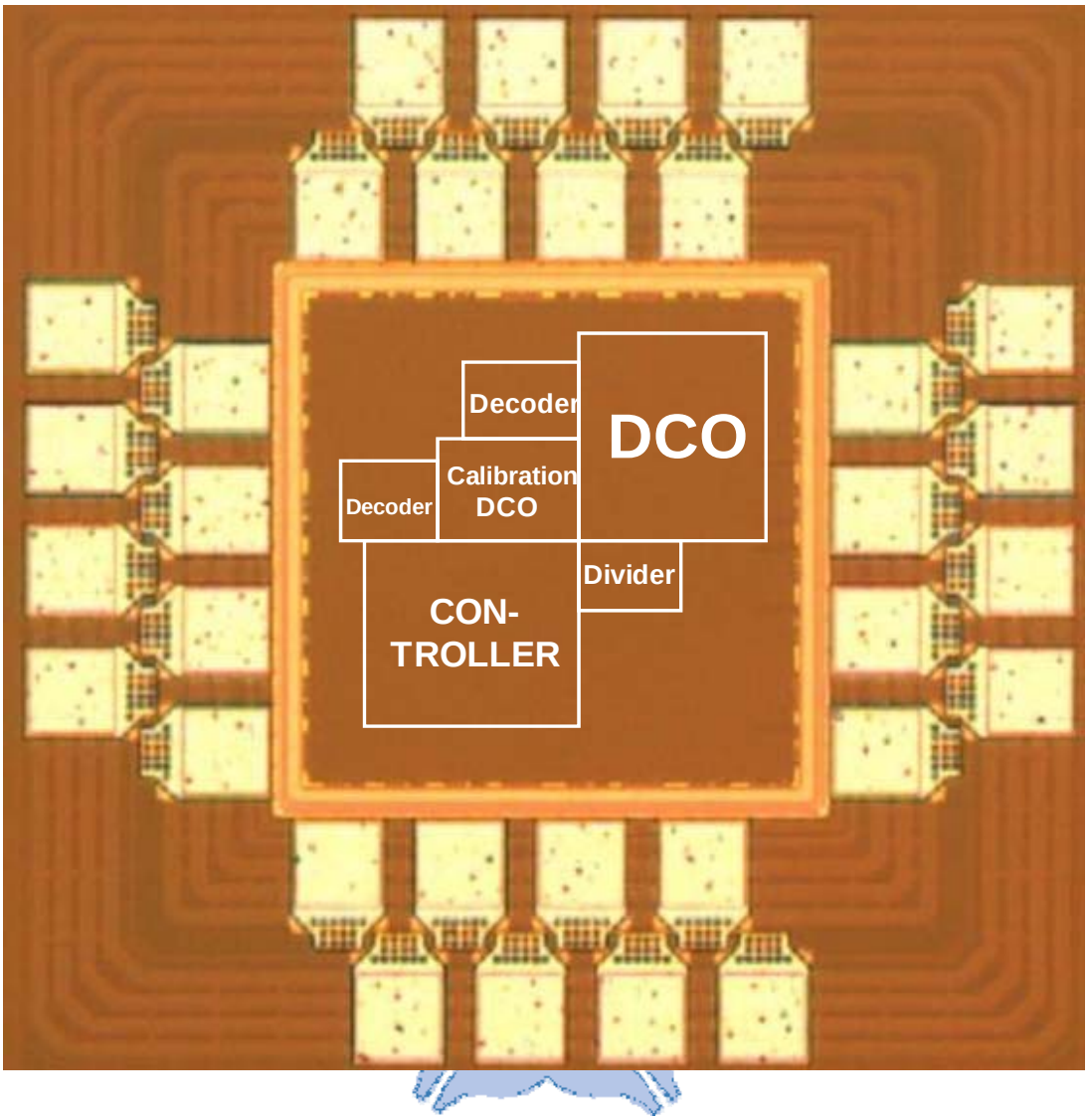


Fig. 2.15 The microphotograph of the test chip

Fig. 2.15 shows the microphotograph of the test chip. The test chip is implemented with a standard performance (SP) 65nm CMOS process. The design parameters of this test chip are determined as follows: M=6, N=5, P=9. It means that the proposed DCO has 64 coarse-tuning steps in the coarse-tuning stage and 32 fine-tuning steps in the fine-tuning stage.

Table 2.5 Properties of the DCO

	Chip Meas.	PostSim TT	PostSim FF	PostSim SS
Coarse-Tuning Step (ps)	291.980	288.856	209.178	465.951
Fine-Tuning Range (ps)	653.693	565.68	439.44	832.82
Average Resolution (ps)	18.268	18.25	14.18	26.87
Max. Frequency (MHz)	538.704	610.090	839.067	383.568
Min. Frequency (MHz)	75.135	49.012	67.527	30.493
Compensation code	21	17	18	15

Table 2.5 shows the properties of the proposed DCO in chip measurement and in post-layout simulation with PVT variations. The compensation code varies with different PVT conditions. The fine-tuning range is always larger than coarse-tuning step with different PVT conditions. The measurement results show that the DCO can output frequency ranges from 75.135MHz to 538.704 MHz. And the resolution in the proposed DCO is about 18.268ps from chip measurement results.

Fig. 2.16 shows the simulation results of DCO's output period vs. DCO control code in the non-calibrated DCO with PVT variations. Because the sub-frequency band is overlapped, therefore the output period is not monotonically decreasing while the DCO control code is increasing.

After the calibration process is done, the output period becomes monotonically decreasing while the DCO control code is increasing. Hence the proposed self-calibration circuit can make sure that the monotonic response of the DCO during DCO coarse-tuning control codes transitions as shown in Fig. 2.17.

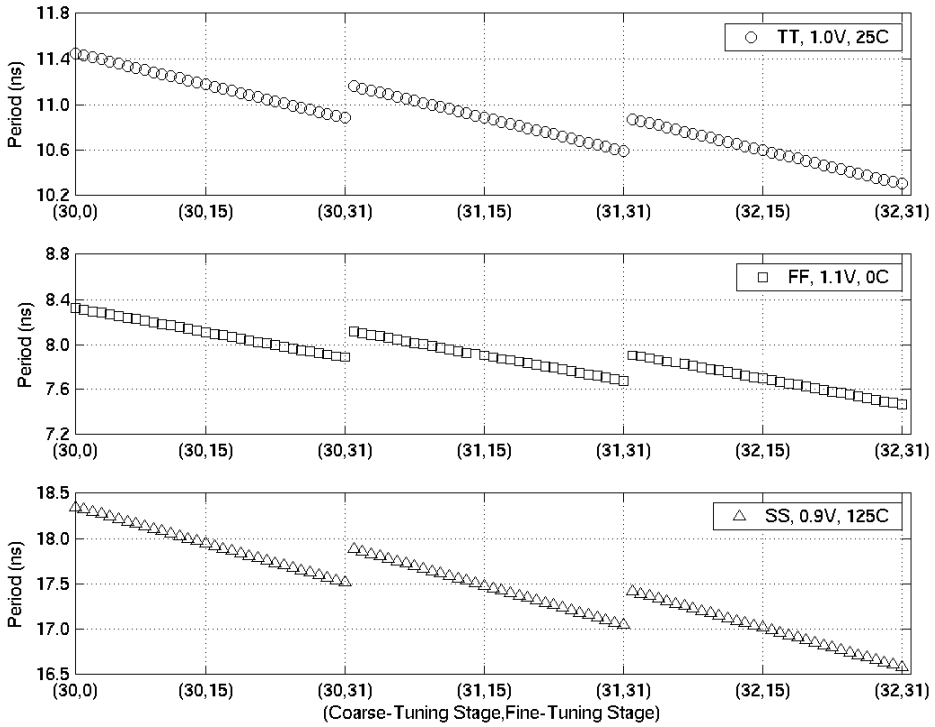


Fig. 2.16 Output period of the non-calibrated DCO

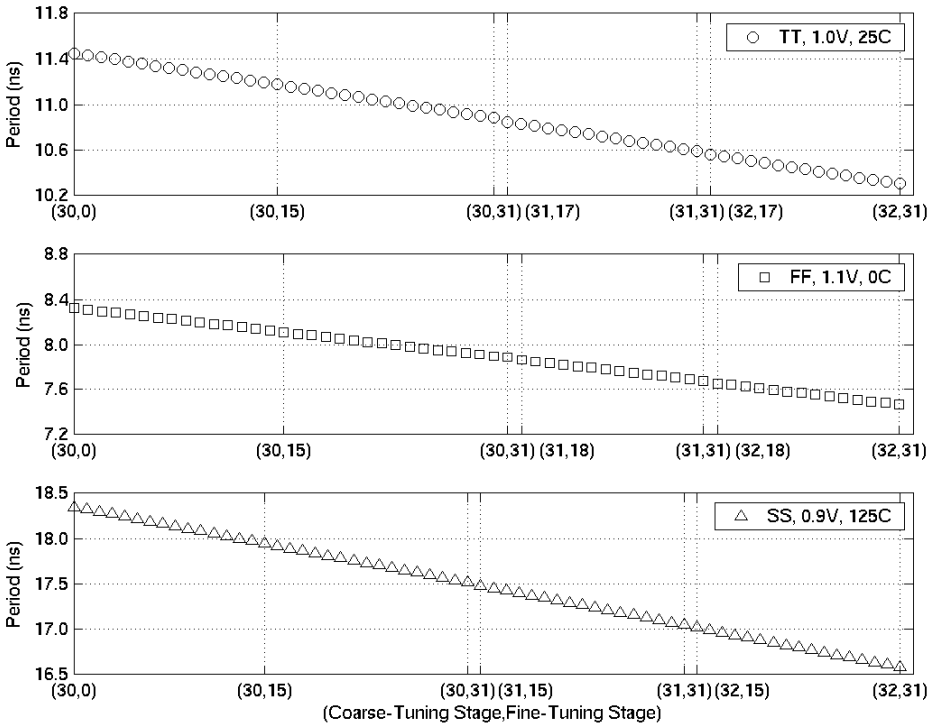


Fig. 2.17 Output period of the calibrated DCO

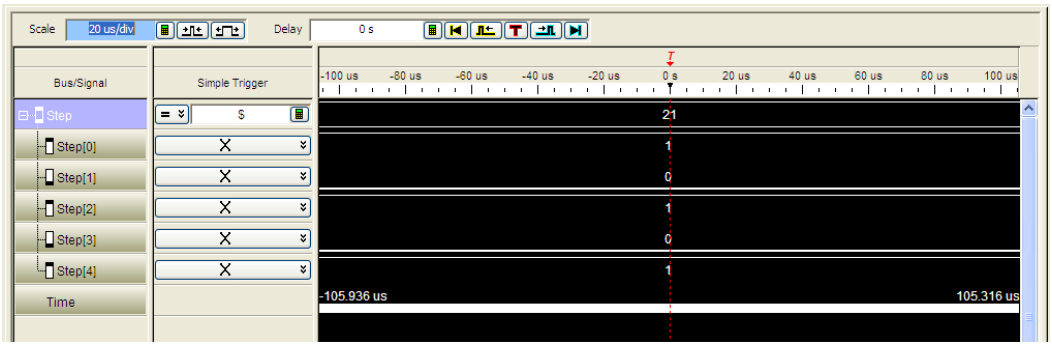


Fig. 2.18 Measured calibration circuit output result

Table 2.6 Period of the DCO output Clock

(Coarse-Tuning,Fine-Tuning)	Period(ns)
(14,00)	16.5017631
(14,31)	15.8480704
(15,00)	16.2097834
…	…
(15,18)	15.8552763
(15,19)	15.8413321
(15,20)	15.8243584
(15,21)	15.8079831
(15,22)	15.7875566
(15,23)	15.7757003

Fig. 2.18 shows the calibration result measured by the Logic Analyzer. The compensation code (Step[4:0]) output by the proposed BISC circuit is 21_{10} in this case. And Table 2.6 shows the measurement results of the DCO output period. It shows that the output period at (15,21) is smaller than (14,31). Thus after the calibration with the proposed BISC circuit, the output period becomes monotonically decreasing while the DCO control code is increasing. Although the output period at (15,19) is already smaller than the output period at (14,31), we choose (15,21) as output to tolerate jitter effects of the DCO.

After calibration process is done, the BISC controller adds the compensation code (Step[4:0]) 21_{10} (10101_2) to the DCO control code if there has changes in the coarse-tuning control code. And it can make sure that the monotonic decreasing response of the DCO during DCO coarse-tuning control code transitions.

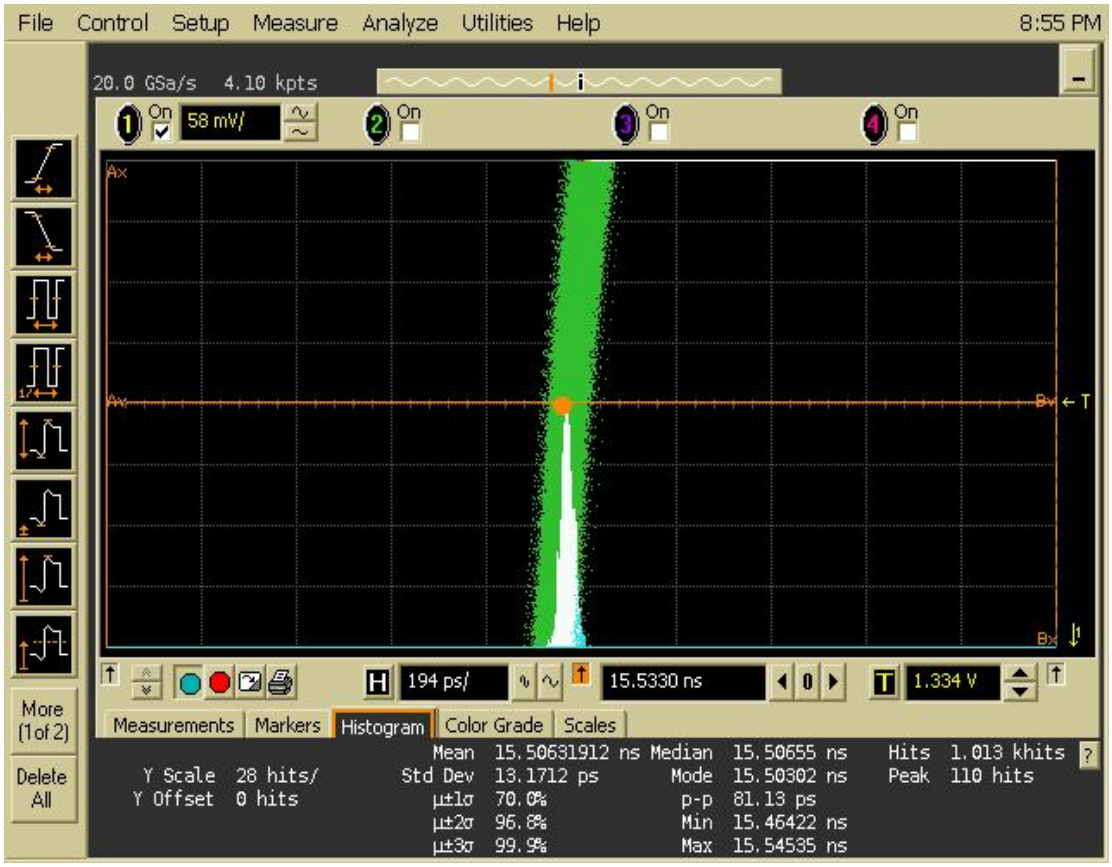


Fig. 2.19 Measured jitter histogram operates at 64.489 MHz

Fig. 2.19 shows the jitter measurement results of the DCO output clock. The root-mean-square jitter and peak-to-peak jitter at 64.489 MHz is 13.171ps and 81.130ps, respectively. Table 2.7 summarizes the test chip performance. In Table 2.7, the interpolated DCO consumes large power consumption thus is not suitable for low-power applications. The proposed DCO with BISC circuit has smaller area and lower power consumption and is very suitable for ADPLL design.

Table 2.7 Performance summary

Performance Indices	**This work**	**[27]**	**[25]**
Process	65nm CMOS	0.18µm CMOS	0.13µm CMOS
Design Approach	Cell-Based	All-Digital	All-Digital
DCO Type	Cascaded	Interpolated	Interpolated
Supply (V)	1.0	1.8	1.28
Frequency Range (MHz)	75.14 - 538.70	33 - 1040	300 - 1300
rms Jitter (ps)	13.171 (@64.49MHz)	13.8 (@950MHz)	10.4 (@950MHz)
p-p Jitter (ps)	81.130 (@64.49MHz)	86.7 (@950MHz)	59 (@950MHz)
LSB Resolution (ps)	18.268	N/A	5.9
Chip Area (mm^2)	0.01	0.32 (Chip) 0.06 (DCO)	0.0075(DCO)
Power (mW)	0.142 (@ 58.7MHz) 0.205 (@481.6MHz)	15.7 (@1.04GHz)	4.48 (@ 950MHz)
Performance Indices	**[26]**	**[22]**	**[11]**
Process	0.18µm CMOS	90nm CMOS	0.18µm CMOS
Design Approach	Cell-Based	Cell-Based	Cell-Based
DCO Type	Cascaded	Cascaded	Cascaded
Supply (V)	1.8	1	1.8
Frequency Range (MHz)	27 - 54	191 - 952	378 - 2400
rms Jitter (ps)	94 (@54MHz)	8.24 (@952MHz)	76 (@134MHz)
p-p Jitter (ps)	N/A	49.95 (@952MHz)	2000 (@134Hz)
LSB Resolution (ps)	1.1	1.47	65 (DCO1)
Chip Area (mm^2)	0.156	N/A	0.16
Power (mW)	1.2 (@54MHz)	0.14 (@200MHz)	15 (@348MHz)

2.3.2.4 Conclusion

In section 2.3.2, a monotonic DCO with built-in self-calibration circuit in 65nm CMOS technology is presented. The proposed DCO can output frequency ranges from 75.135MHz to 538.704 MHz with low-power consumptions. The proposed calibration circuit can solve the non-monotonic problem in cascaded architecture DCOs when the coarse-tuning control code is changed thus is very suitable for ADPLL design in SoC applications.

2.3.3 Interpolation-type DCO

The previous section 2.3.1 proposed the MUX-type DCO, but this structure may encounters glitch problem and DCO non-monotonic issue. Section 2.3.2 then proposed the built-in self-calibration circuit to correct the non-monotonic response in the cascading DCO. Although this method can solve the non-monotonic problem, however, this method may require additional area, power and complex circuit. It increases the design cost and has a heavy burden on the designer. In this section, we will present another, more intuitive DCO architecture called the interpolation-type DCO to solve the problems encountered previously.

Compared with the previous MUX-type DCO, the interpolation-type DCO using the interpolator circuit as fine-tuning stage in the two adjacent sub-frequency to generate the fine-tuning delay. So it doesn't need to overlap the sub-frequency band to make sure that there will not have any frequency dead zone in the DCO. Because using the interpolator to generate fine-tuning delay, it can seamlessly switch the

control code in two adjacent sub-frequency delays. It can make sure that the output frequency has monotonic response.

2.3.3.1 Structure

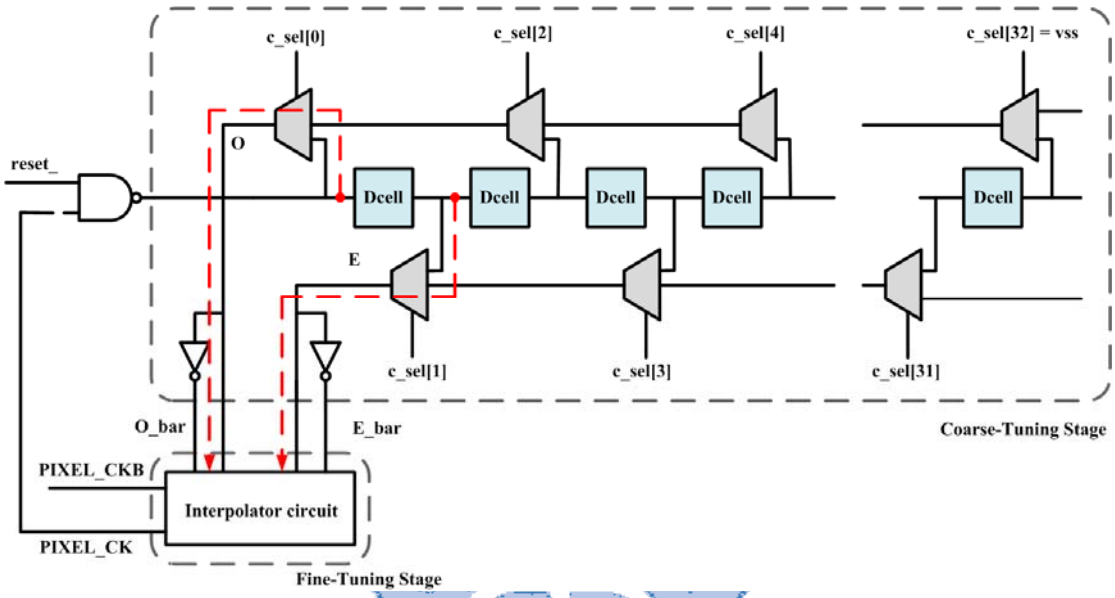


Fig. 2.20 The architecture of the interpolation-type DCO

Fig. 2.20 shows the architecture of the interpolation-type DCO. The interpolation-type DCO is composed of the coarse-tuning stage and the fine-tuning stage. The coarse-tuning stage which has $32(=2^5)$ delay cells with $33(=2^5+1)$ multiplexers can provide $32(=2^5)$ different delays. In order to generate a sufficient delay time in 65nm CMOS process, the delay cells in the cell-library are used to build up the coarse-tuning stage.

In the coarse-tuning stage, the controller selects two adjacent branch delays as one coarse-tuning step and sends two branch delays to the fine-tuning stage by both signals O and E.

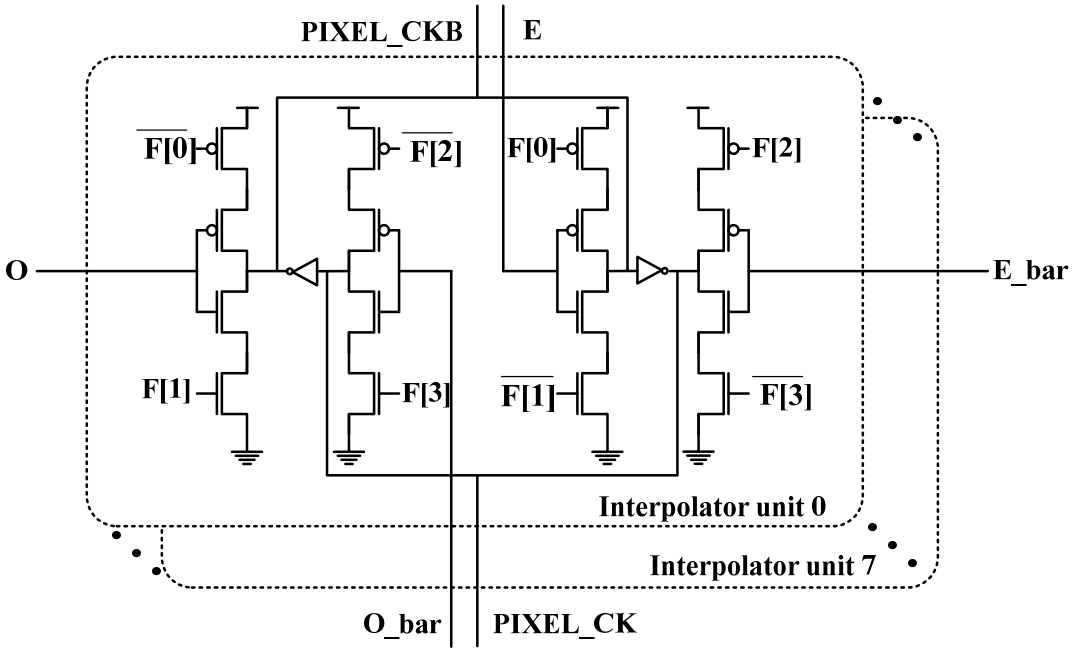


Fig. 2.21 The fine-tuning stage of the interpolation-type DCO [25]

The interpolation-type DCO uses the interpolator circuit as its fine-tuning stage. The interpolator circuit [25] is shown in Fig. 2.21. The interpolator circuit receives the signals O, E, O_bar, E_bar from coarse-tuning stage as its input and generates output signal (PIXEL_CK) as DCO output clock. The interpolator circuit has 8 interpolator units in parallel to generate a sufficient equivalent resolution. The interpolator unit is composed of tri-state inverters and inverters. The fine-tuning signal controls the relative weight of two selected branches. Each interpolator unit has 8 ($=2 \cdot 2^2$)(i.e. F[0],…, F[3], $\overline{F[0]}$,…, $\overline{F[3]}$) fine-tuning control signals and it can provide 4 different delays. So the interpolator circuit has 64 ($=2 \cdot 2^2 \cdot 2^3$)(i.e. F[0],…, F[31], $\overline{F[0]}$,…, $\overline{F[31]}$) fine-tuning control signals and it can provide 32($=2^2 \cdot 2^3$) different delays in one fine-tuning stage. Table 2.8 shows the switching sequence of the interpolator circuit. Therefore, the resolution of the coarse-tuning stage is about 550.129ps, and the resolution of the fine-tuning stage is about 17.188ps ($550.129 / 8 \cdot 2^2$) for the proposed delay cell in 65nm CMOS process.

Table 2.8 Interpolation Switching Sequence

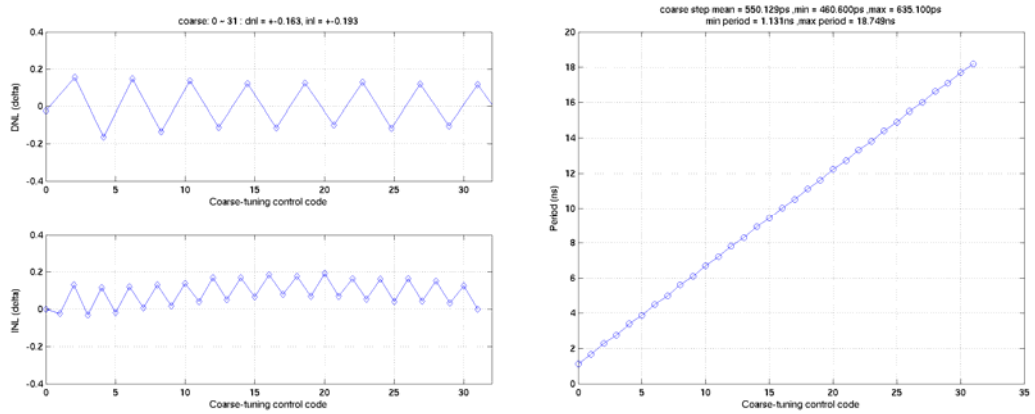
Seq	F[0]	F[1]	F[2]	F[3]	F[4]	…	F[30]	F[31]	Fine-tuning step
0	0	1	1	1	1	…	1	1	$1/32 \cdot T_{coarse\_step}$
1	0	0	1	1	1	…	1	1	$2/32 \cdot T_{coarse\_step}$
2	0	0	0	1	1	…	1	1	$3/32 \cdot T_{coarse\_step}$
3	0	0	0	0	1	…	1	1	$4/32 \cdot T_{coarse\_step}$
…	…	…	…	…	…	…	…	…	…
30	0	0	0	0	0	…	0	1	$31/32 \cdot T_{coarse\_step}$
31	0	0	0	0	0	…	0	0	$32/32 \cdot T_{coarse\_step}$

2.3.3.2 Simulation Result

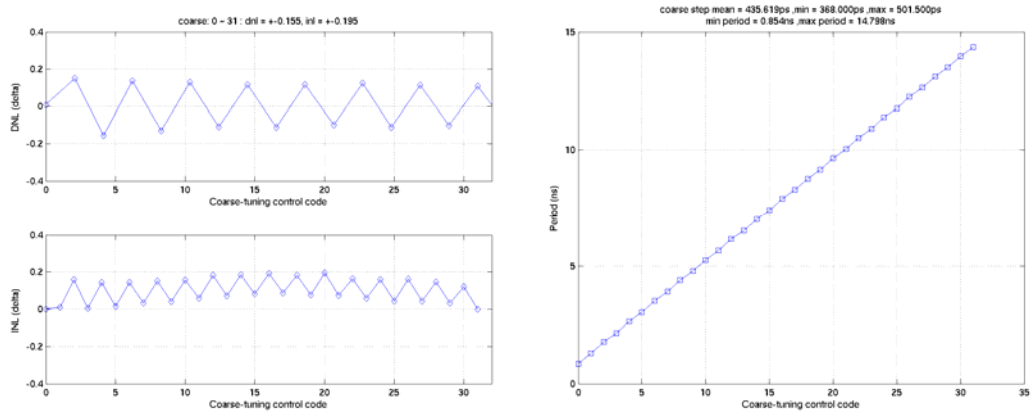
The interpolation-type DCO is simulated in post-layout simulation. Fig. 2.22 shows the period of the DCO output clock versus coarse-tuning stage control code (0-31), when fine-tuning stage control code set to zero, and shows INL and DNL of the interpolation-type DCO.

The simulations variables of each corner are process, voltage, and temperature, respectively. The circle represents the TT corner, 1.0 V, 25°C, the square represents the FF corner 1.1V, 0°C, and the upward-pointing triangle represents the SS corner, 0.9V, 125°C, respectively.

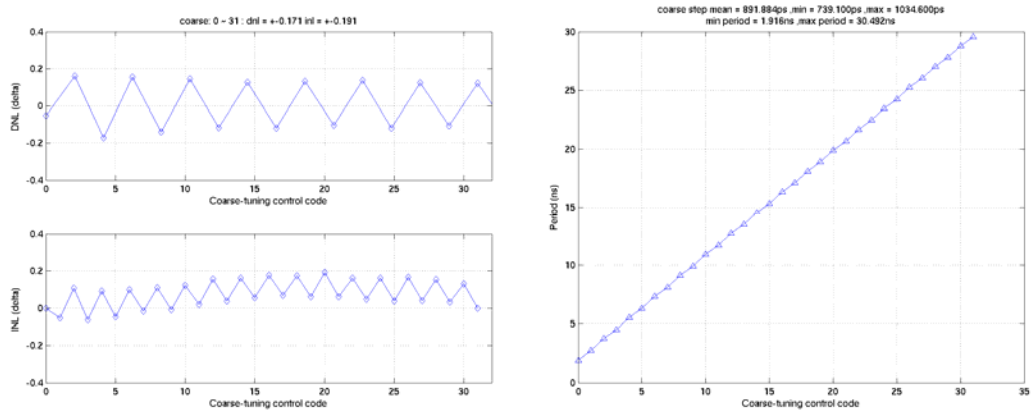
In TT corner, the DCO period range is from 1.131ns to 18.749ns, the DNL is ±0.163Δ, and the INL is ±0.193Δ. In FF corner, the DCO period range is from 0.854ns to 14.798ns, the DNL is ±0.155Δ, and the INL is ±0.195Δ. In SS corner, the DCO period range is from 1.916ns to 30.492ns, the DNL is ±0.171Δ, and the INL is ±0.191Δ. Three corners are shown in (a), (b), and (c), respectively.



(a) TT corner, DNL: ±0.163Δ, INL: ±0.193Δ, DCO range: 1.131ns ~ 18.749ns



(b) FF corner, DNL: ±0.155Δ, INL: ±0.195Δ, DCO range: 0.854ns ~ 14.798ns



(c) SS corner, DNL: ±0.171Δ, INL: ±0.191Δ, DCO range: 1.916ns ~ 30.492ns

Fig. 2.22 Simulation of DCO period versus coarse-tuning code 0 ~ 31

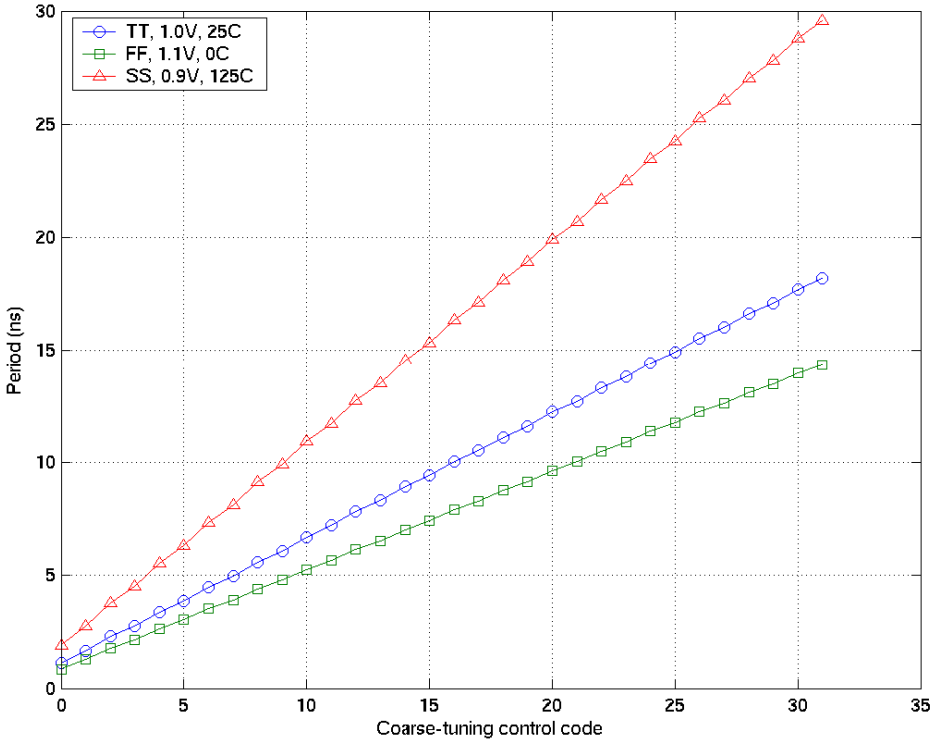


Fig. 2.23 Comparison of the interpolation-type DCO period in PVT variations

Fig. 2.23 shows the comparison of the DCO period in PVT variation. The DCO operation range is from 1.914ns to 14.798ns, covered in each corner.

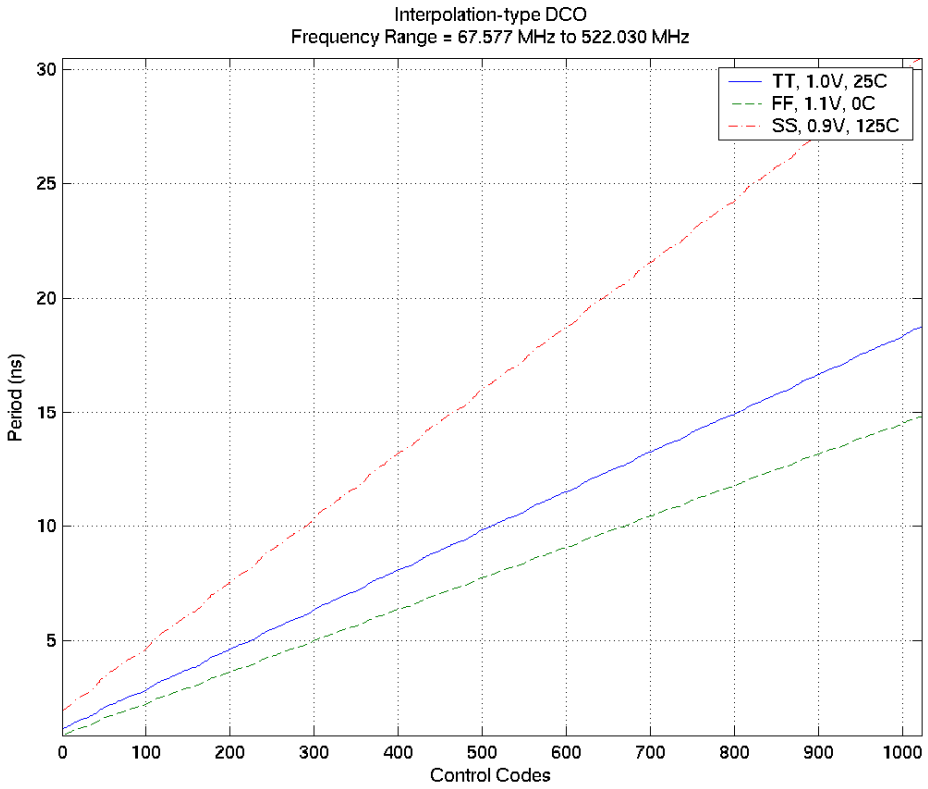


Fig. 2.24 Simulation of interpolation-type DCO

Fig. 2.24 shows the period of the interpolation-type DCO in both coarse-tuning control code (0~31) and fine-tuning control code (0~31) under different PVT variations. Table 2.9 shows the properties of the interpolation-type DCO. The average step of coarse-tuning stage delay is 550.129ps in TT corner, 435.619ps in FF corner, and 891.844ps in SS corner. The average range of fine-tuning stage delay is 517.522ps in TT corner, 406.500ps in FF corner, and 846.494ps in SS corner. The average step of fine-tuning stage delay is 17.205ps in TT corner, 13.617ps in FF corner, and 27.907ps in SS corner. We can see the range of fine-tuning stage delay and the step of one coarse-tuning stage delay are almost equal.

Table 2.9 Properties of the interpolation-type DCO

Post-Layout Simulation DCO Control Code : 0 ~ 1023					
	Avg. Coarse Step (ps)	Max Coarse Step (ps)	Min Coarse Step (ps)	Max Period (ns)	Min Period (ns)
TT corner	550.129	635.100	460.600	18.749	1.131
FF corner	435.619	501.500	368.000	14.798	0.854
SS corner	891.884	1034.600	739.1000	30.492	1.916
	Avg. Fine Range (ps)	Max Fine Range (ps)	Min Fine Range (ps)	Max Frequency (ns)	Min Frequency (ns)
TT corner	517.522	593.500	433.700	884.173	53.336
FF corner	406.500	464.300	344.700	1171.235	67.577
SS corner	846.494	976.200	700.600	522.0297	32.796
	Avg. Fine Step (ps)	Max Fine Step (ps)	Min Fine Step (ps)		
TT corner	17.205	45.400	5.000		
FF corner	13.617	33.500	4.000		
SS corner	27.907	77.900	6.400		

2.4 ADPLL Controller

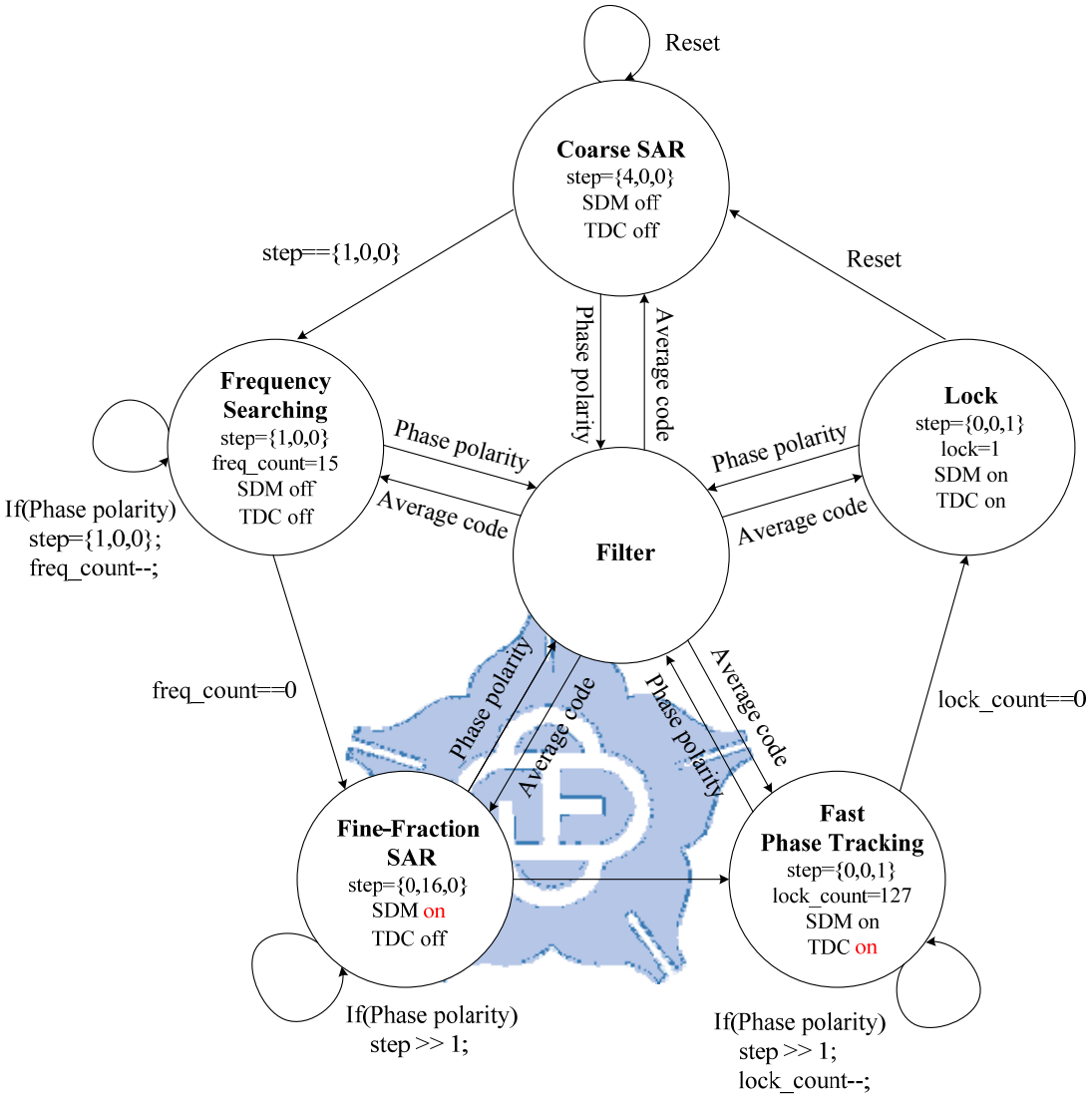


Fig. 2.25 The finite state machine of the ADPLL controller

Fig. 2.25 shows the system states of proposed ADPLL controller. The algorithm of the controller will influence the overall tracking performance and the locking time. The proposed ADPLL has divided into six states, and these are Coarse SAR, Frequency Searching, Fine-Fraction SAR, Fast Phase Tracking, Lock, and Filter respectively. In the proposed ADPLL, the length of step code is 19 bits. The step code is composed of 5bits coarse-tuning code, 5bits fine-tuning code and 9bits

fraction-tuning code. In Fig. 2.25, the step code is expressed by {coarse-tuning code, fine-tuning code, fraction-tuning code}.

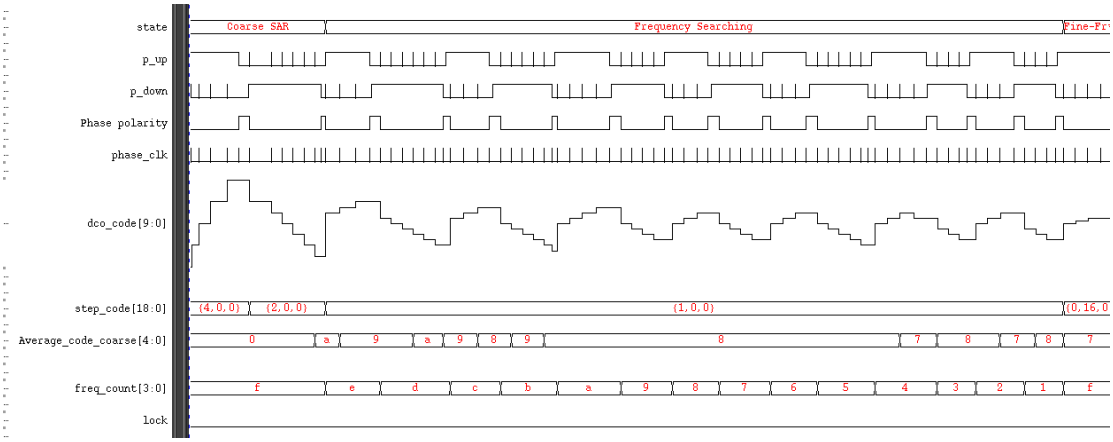


Fig. 2.26 Timing diagram in Coarse SAR state and Frequency Searching state

The first state is coarse successive approximation register (SAR) tuning stage. In this state, the controller accords to the reference clock frequency to find the approximate frequency. Whenever a change in phase polarity occurs, the step code is divided by 2 to reduce the tuning step. The average code from digital loop filter is loaded into the control code, restoring the baseline frequency. When the step code is reduced from {4,0,0} to {1,0,0}, the ADPLL controller enters into Frequency Searching state. The purpose of Frequency Searching state is to find the suitable coarse-tuning code. In this state, the step code is fixed to {1,0,0} for 16 clock cycles, and the digital loop filter will update the average code to find the best baseline frequency. Fig. 2.26 shows the timing diagram in both Coarse SAR state and Frequency Searching state.

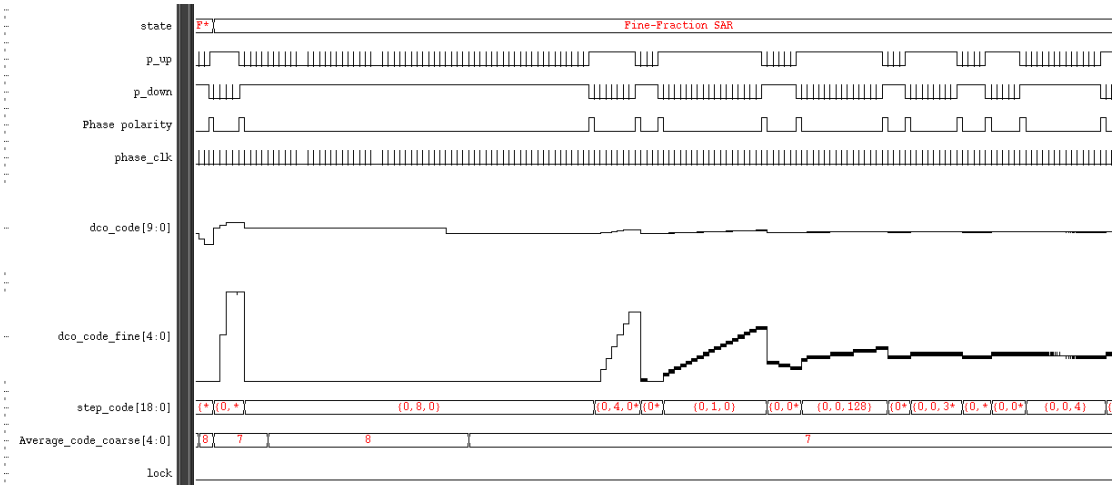


Fig. 2.27 Timing diagram in Fine-Fraction SAR state

When the freq_count equal to zero, the controller enters into Fine-Fraction SAR state. In this state, the step code is be initialed to {0,16,0}, and the working principle is similar to Coarse SAR state. Both the fine-tuning code and the fraction-tuning code is changed until the step code is reduced to {0,0,1}, but the coarse-tuning code is changed when the frequency is located between two frequency bands. Then, the Sigma-Delta Modulator is turned on to dither the DCO fine-tuning code to improve the DCO equivalent resolution. Fig. 2.27 shows the timing diagram in Fine-Fraction SAR state.

When the step code is reduced down to {0,0,1}, the controller enters to Fast Phase Tracking state to track the phase error between HSYNC and HSOUT. After this state, the TDC circuit is turned on to compensate the phase error when the reference clock has the instant input jitter. When the lock_count is equal to zero, the controller enters into the last state Lock, and then the ADPLL is lock. In these 6 states expect Filter state, when a change in phase polarity the controller will send current control code to filter to calculate baseline frequency control code. Finally, Fig. 2.28 shows the timing diagram of the ADPLL locking procedure.

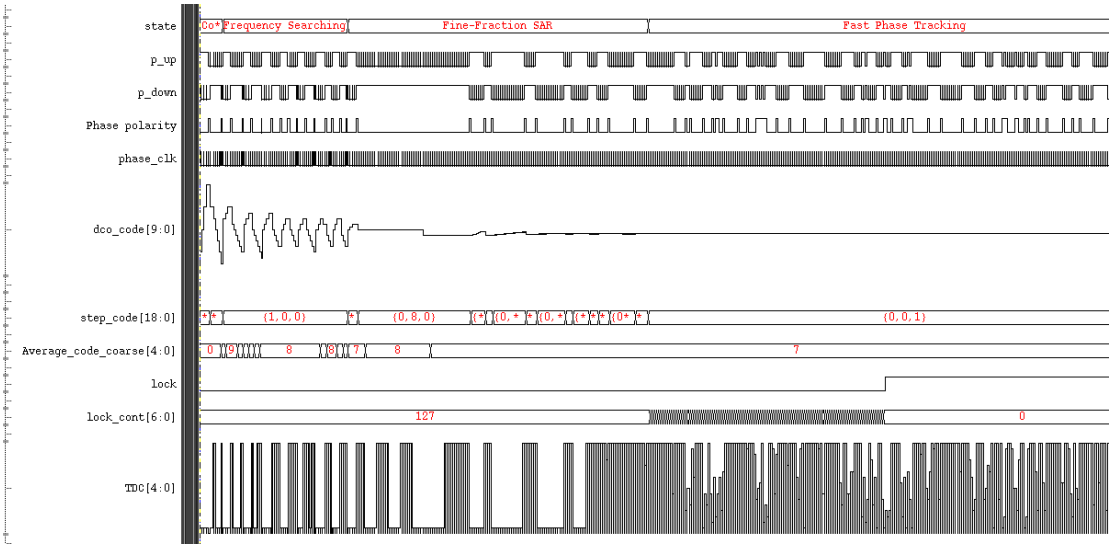


Fig. 2.28 Timing diagram of the ADPLL locking procedure

2.5 Digital Loop Filter

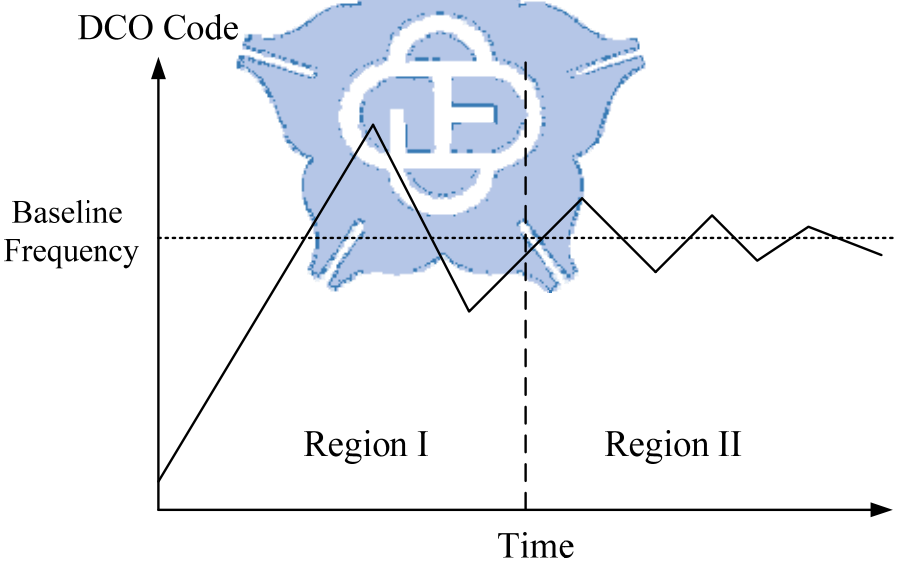


Fig. 2.29 ADPLL Frequency tracking procedure

The ADPLL continues tracking the frequency and the phase of the reference clock by changing the DCO control code. Fig 2.29 explains the tracking procedure and locking procedure of the ADPLL. In Region I, the ADPLL has large frequency error and phase error. After entering Region II, the ADPLL has small frequency error

and phase error, and the frequency tiny swings nearby the baseline frequency, as shown in Fig 2.29. However, when there has HSYNC jitter or supply noise, in Region II, the ADPLL loop will be unstable and has large noise in pixel clock. Therefore, the ADPLL has to keep tracking frequency for stabilizing the loop. The digital loop filter [28] is introduced to reduce these effects and makes the ADPLL output period jitter can be minimized and has a stable loop.

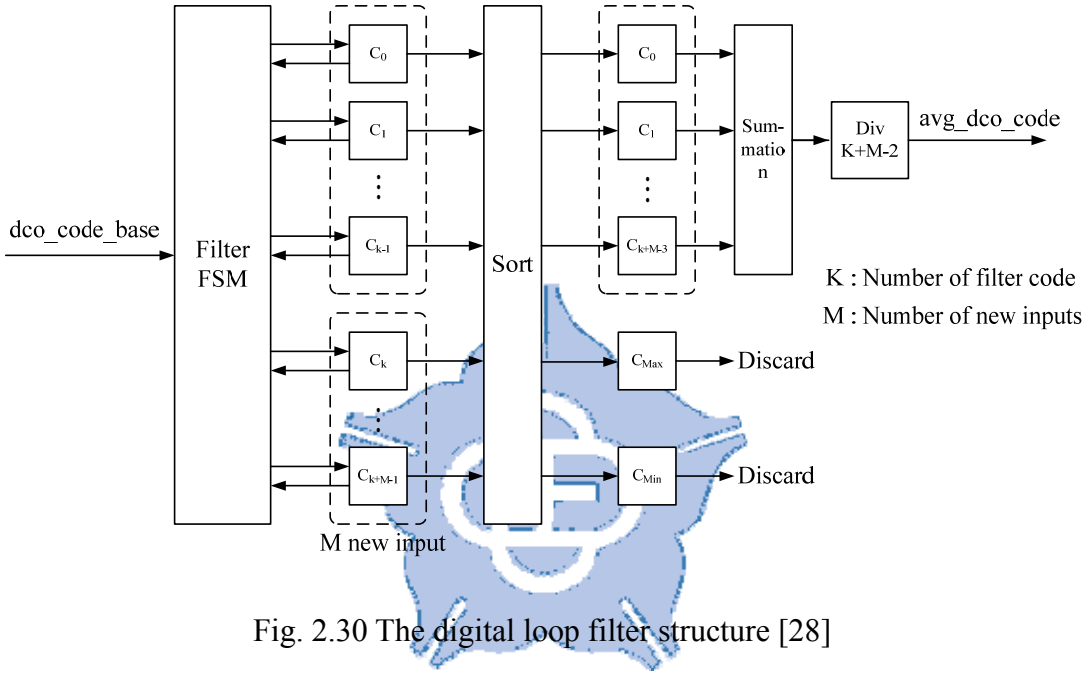


Fig. 2.30 The digital loop filter structure [28]

The digital loop filter [28] is shown in Fig. 2.30. The digital loop filter receives the dco_code_base from the ADPLL controller and is sent into the finite state machine of digital loop filter. In the beginning of the digital loop filter, the finite state machine latches K input into registers in K reference clock cycles. After initiation state, in each reference clock cycle the digital loop filter continuing latches the new dco_code_base into registers and renews the values which are stored in registers. The digital loop filter discards both maximum and minimum dco_code_base, and then generates the baseline frequency code avg_dco_code by averaging the C_0,C_1,…,C_{K+M-3}. Therefore, the baseline frequency is updated by the digital loop filter.

When the ADPLL controller detects the phase polarity from the PFD, the controller sends the dco_code_base to the digital loop filter. Then, the digital loop filter updates the avg_dco_code and is sent back to the ADPLL controller to reduce the phase error and increase the stability of the ADPLL loop.

2.6 Dithering Technology

2.6.1 Dithering Theorem

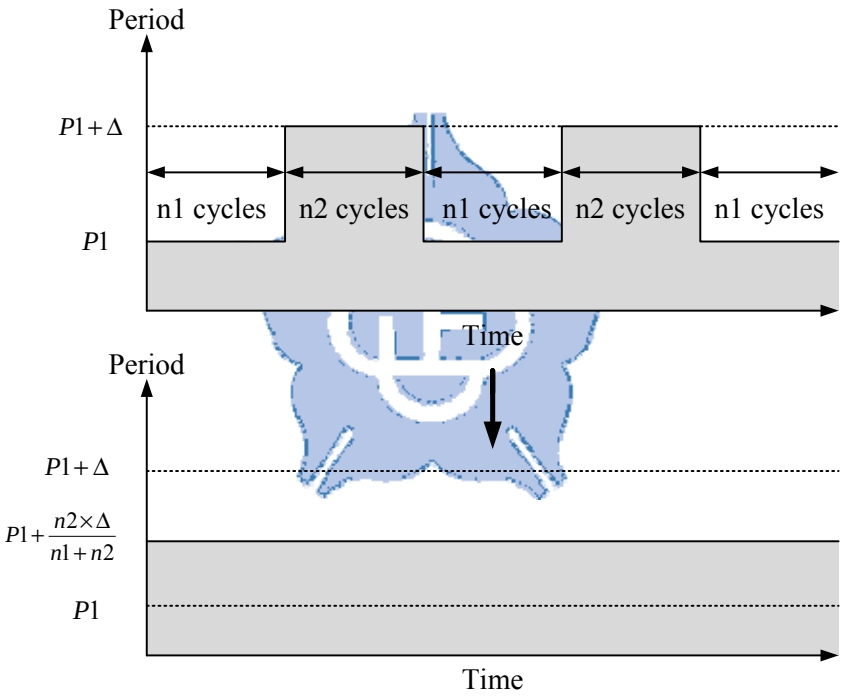


Fig. 2.31 Dithering Technology

The proposed ADPLL uses DCO dithering technology by Sigma-Delta Modulator (SDM) to improve the DCO equivalent resolution. Fig. 2.31 explains the dithering technology. The x axis is the time and y axis is the period of DCO output clock. In the top half of the figure, the DCO output generates n1 cycles of period P1, and generates n2 cycles of period P1+Δ in sequential. Hence, the average period of

the DCO is $\frac{P1\times n1+(P1+\Delta)\times n2}{n1+n2}=P1+\frac{n2\times\Delta}{n1+n2}$. By adjusting the value of n1 and n2, we can precisely control the average period of the DCO output cycle. By mixing the DCO output period P1 and period P1+Δ, the DCO equivalent resolution has improved from the original Δ to $\frac{\Delta}{n1+n2}$, and the result is shown in the bottom half of the Fig. 2.31. In video display applications, the frequency multiplication factor of clock generator is from 800 to 2592, hence, in one reference clock period (HSYNC) it has up to N = 2592 pixel clock cycles.

The architecture of ADPLL with SDM can significantly improve the DCO equivalent resolution, which can reduce the phase error between the reference clock (HSYNC) and pixel clock of the last output time. Because the SDM is used, the real resolution of the DCO doesn't need to be femto-second level. The DCO dithering technology makes the DCO circuit easier to be designed and reduces the circuit complexity. Therefore, the proposed ADPLL architecture will use the SDM with DCO circuit to solve the problem of the reference clock phase is difficult to tracked in high frequency multiplication factor PLL.

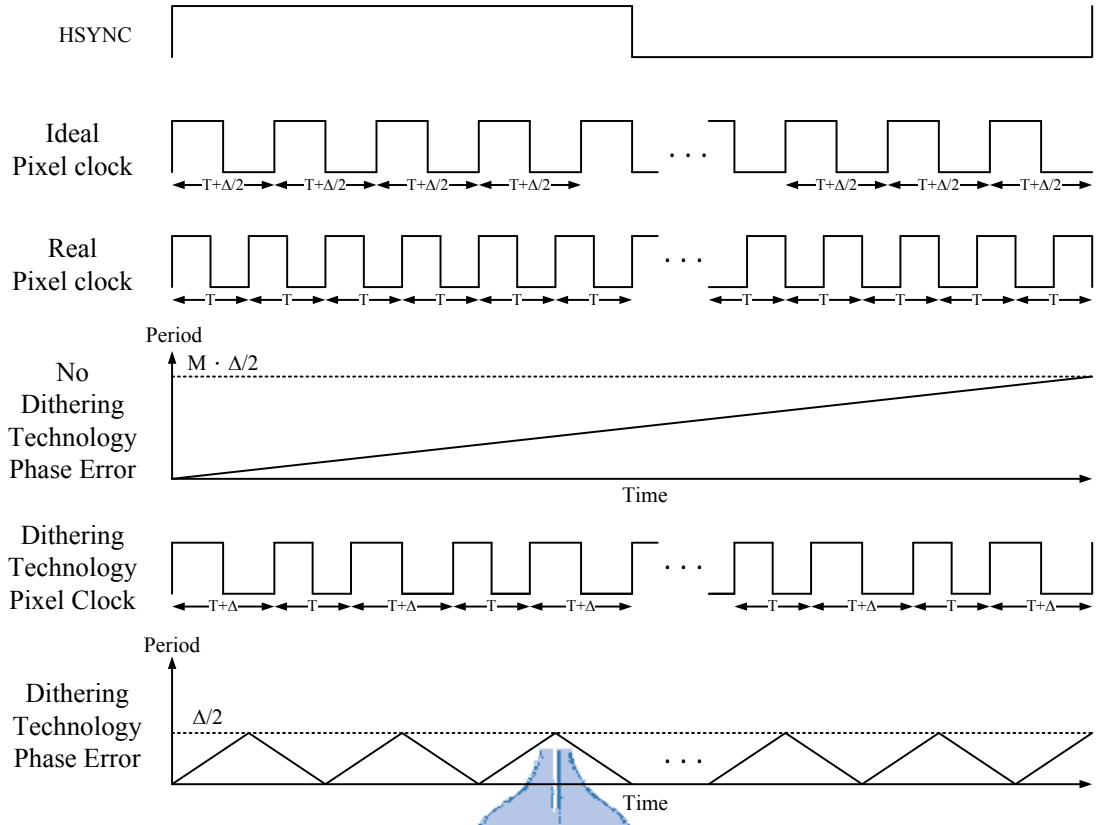


Fig. 2.32 Phase Error reduction by using dithering technology

The proposed ADPLL uses DCO dithering technology to reduce the phase error between the reference clock (HSYNC) and pixel clock as shown in Fig. 2.32. In Fig. 2.32, the frequency multiplication factor is M, and the DCO resolution is Δ. Suppose the ideal period of output pixel clock cycle is T+Δ/2. In one reference clock (HSYNC), if we assume that the period of all pixel clock cycles are T, and according to the deviation of frequency, the phase error will continue to be accumulated. Before the next positive edge of reference clock (HSYNC), the total accumulated phase error becomes M · Δ/2. Assume that using the SDM to control the DCO output period between T and T+Δ alternately, the accumulated phase error problem can be solved, and the phase error can be limited less than Δ/2. From the above discussion, the architecture of proposed ADPLL with the SDM will significantly improve the phase tracking ability than conventional PLLs.

2.6.2 Working Principle

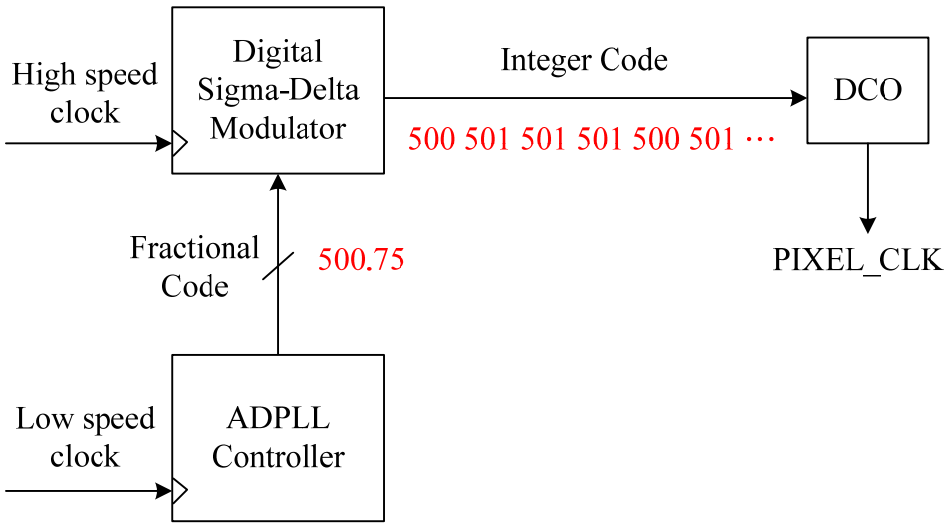


Fig. 2.33 The working principle of Sigma-Delta Modulator [29]

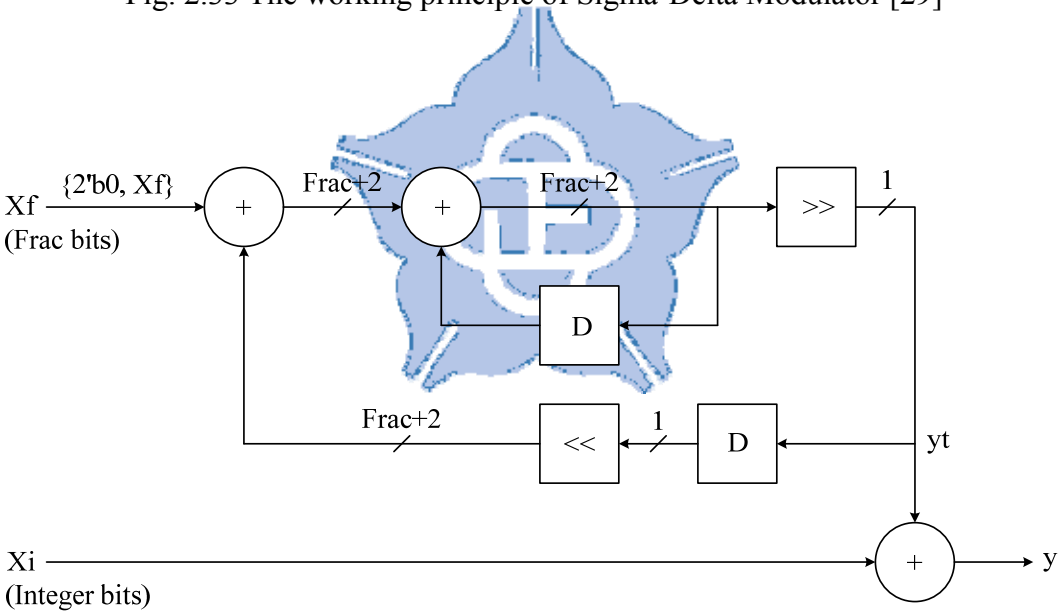


Fig. 2.34 First-order Sigma-Delta Modulator

In the proposed ADPLL, the first-order SDM is applied to implement the dithering technology. Fig. 2.33 shows the working principle of the dithering technology and Fig. 2.34 shows the architecture of the first-order SDM. After the ADPLL controller enters into the Fine-Fraction SAR state, the SDM is turned on, and the ADPLL controller sets the fractional code to control the SDM. Then the SDM

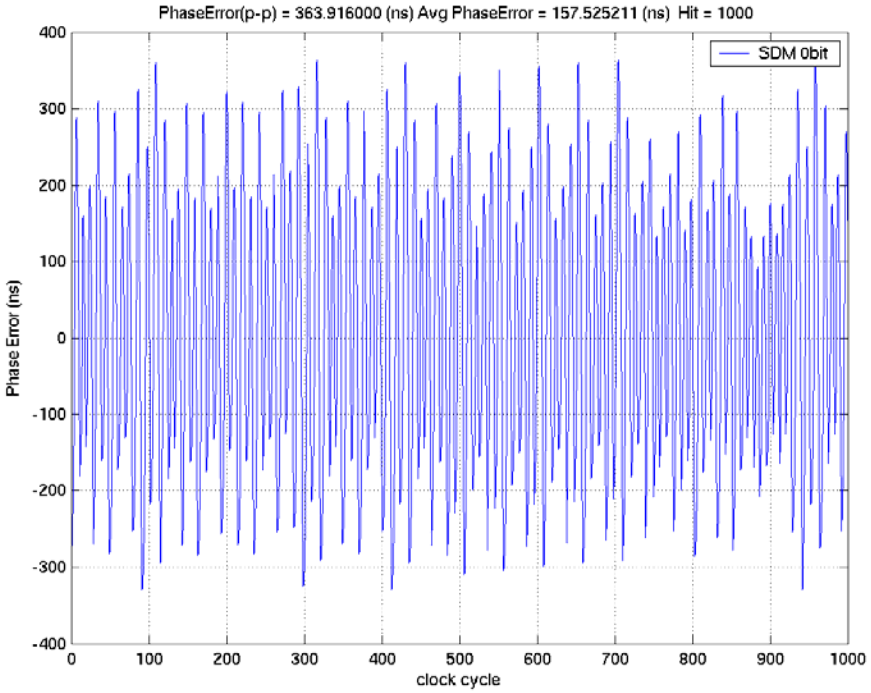
receives the fractional code to generate a series high speed integer code to tune the DCO frequency. According to the operation of SDM, the average of series high speed integer code is near to the fractional code. The SDM is triggered by high speed clock to over sample the data, and the ADPLL controller is triggered by low speed clock. Therefore, when the SDM is used, the DCO equivalent resolution has been improved, and the accumulation of the phase error can be decreased.

2.6.3 Simulation Result

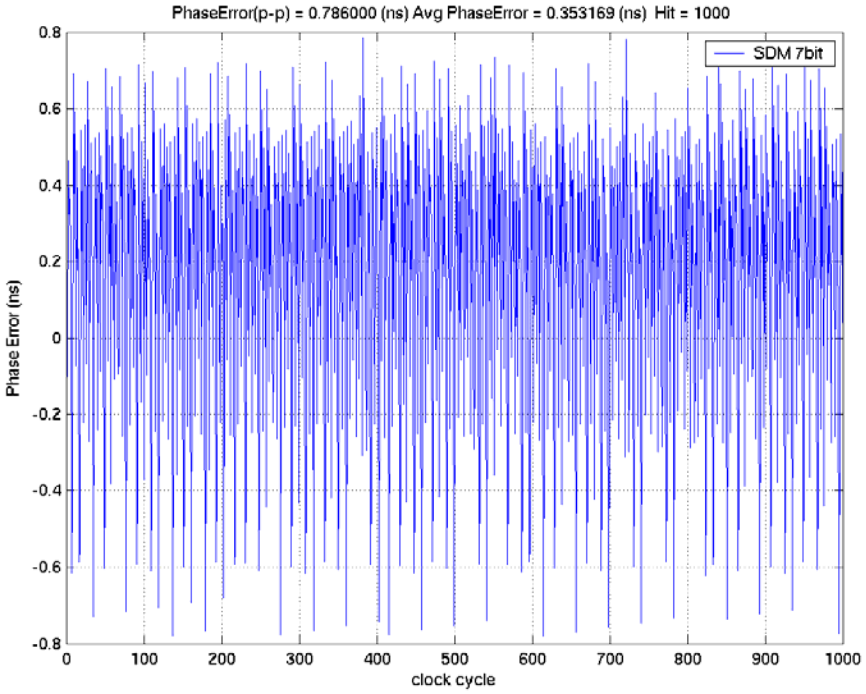
Fig. 2.35 shows the simulation results of Sigma-Delta Modulation with different fractional bits. If we assume that the ADPLL with ideal input HSYNC clock (no HSYNC jitter) in WUXGA mode. The detailed information is listed in Table 2.10. From Table 2.10, the peak-to-peak phase error is 363.916ns with 0bit SDM, 0.786ns with 7bit SDM, 0.544ns with 8bit SDM, and 0.346ns with 9bit SDM, respectively. The simulation shows that when there are more SDM fractional bits, the better performance is. The reason is that when the fractional bit is increased, the equivalent DCO resolution will be decreased.

Table 2.10 Peak-to-peak and average phase error with different SDM fractional bits in WUXGA mode

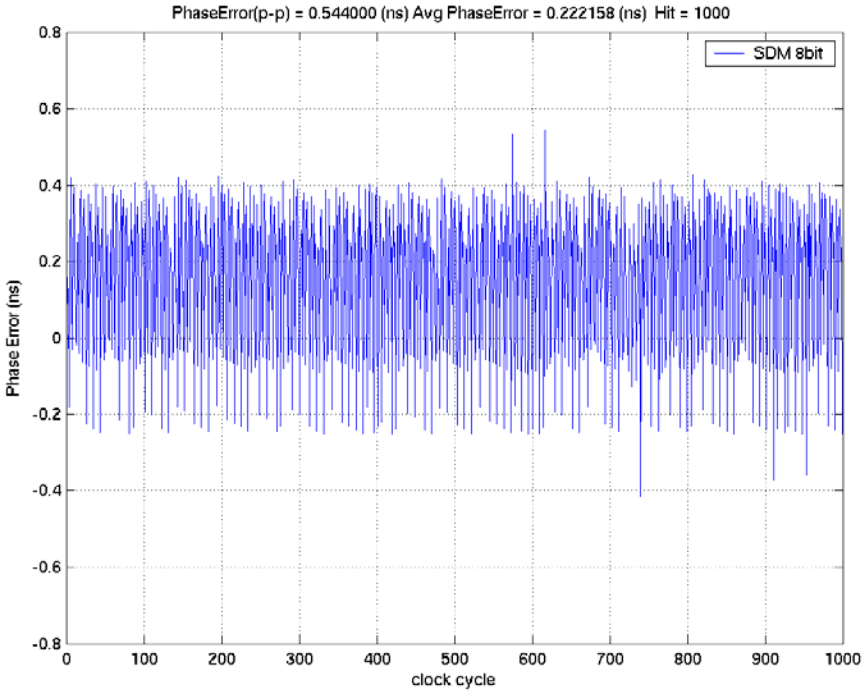
	0 bit	7 bit	8 bit	9 bit
Peak-to-Peak Phase Error (ns)	363.916	0.786	0.544	0.346
Average Phase Error (ns)	157.525	0.353	0.222	0.093



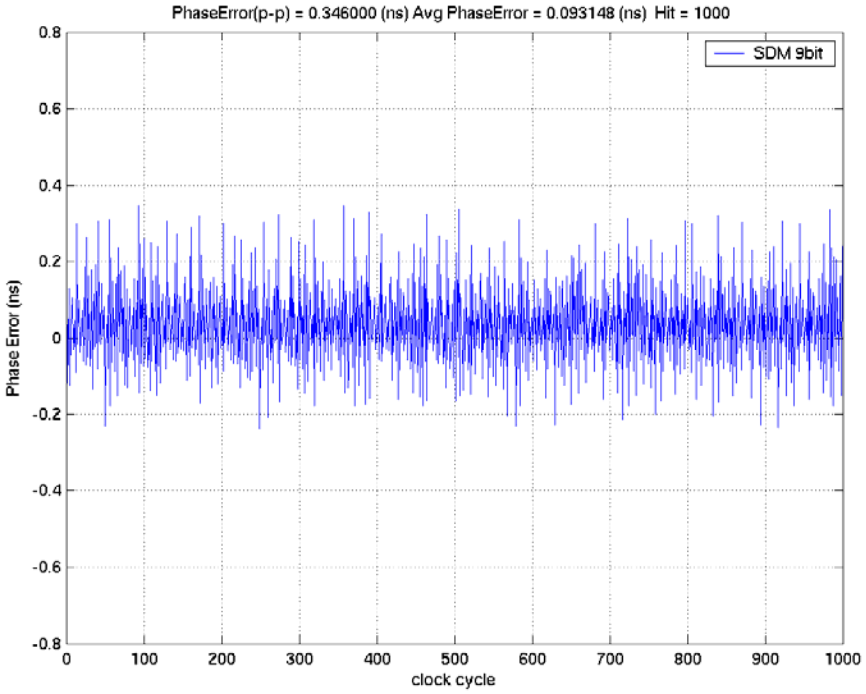
(a) 0bit Sigma-Delta Modulation



(b) 7bit Sigma-Delta Modulation



(c) 8bit Sigma-Delta Modulation



(d) 9bit Sigma-Delta Modulation

Fig. 2.35 Sigma-Delta Modulation with different fractional bits

2.7 Fast Phase Tracking Technology

2.7.1 Time-to-Digital Converter for Fast Phase Tracking

Although the proposed ADPLL uses the SDM DCO to significantly decrease the phase drift, but it still can not meet the specification requirements. For the impact of HSYNC jitter, we need an improvement circuit to further solve this problem. For the issue of the reference clock jitter, it caused the phase error between HSYNC and HSOUT and can't be decreased. As a result in the proposed architecture of the ADPLL uses the time-to-digital converter (TDC) for fast phase tracking.

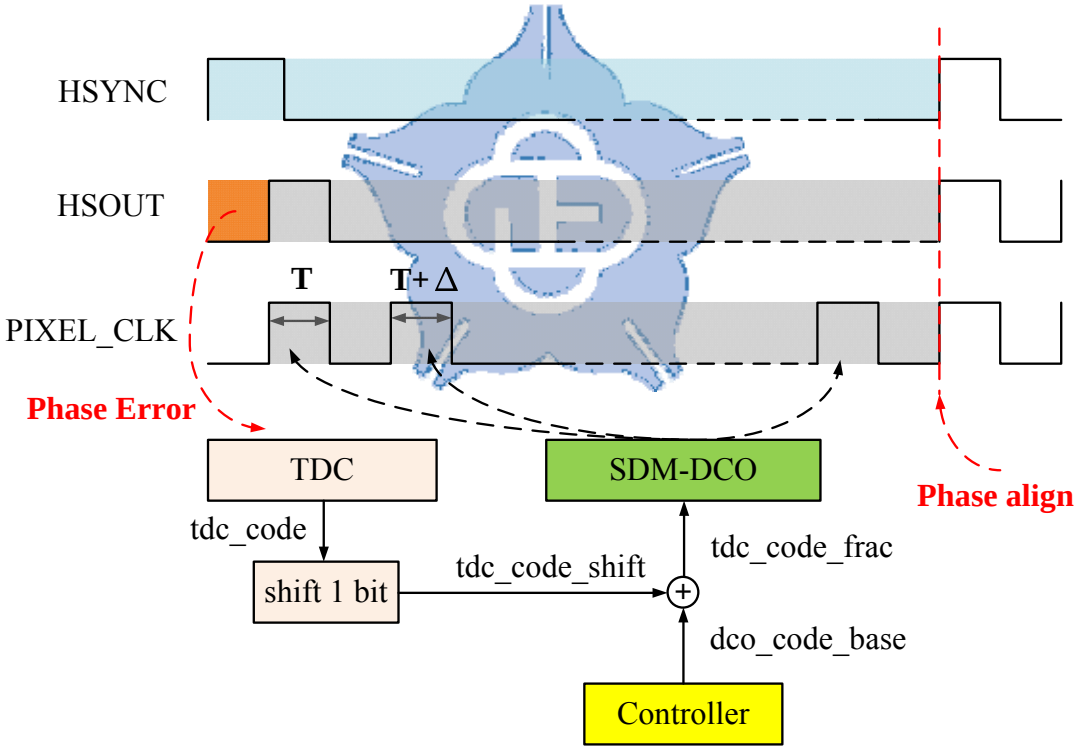


Fig. 2.36 The TDC working principle

Fig. 2.36 shows the working principle of the proposed TDC. When the ADPLL frequency searching is locked, the ADPLL controller enters the Fast Phase Tracking State. The first step is using the TDC to quantize the phase error between HSYNC and

HSOUT into a digital code (tdc_code). Then, the tdc_code is divided by two and then sent to SDM DCO. The SDM DCO will adjust the percentage of the T+Δ period and T period according to the phase error. The TDC can compensate the phase error caused by the reference clock jitter. Therefore, before the next positive edge of HSYNC, the phase error caused by the previous reference clock jitter has been completely compensated. So it can align the phase of HSYNC and HSOUT, and reduces the phase error of output. Moreover, because the phase error will be immediate compensated by TDC circuit before the next positive edge of HSYNC, the phase error will not accumulate to next clock cycle. Therefore, we can expect to significantly reduce the phase error in the non-ideal working environment.

2.7.2 Structure

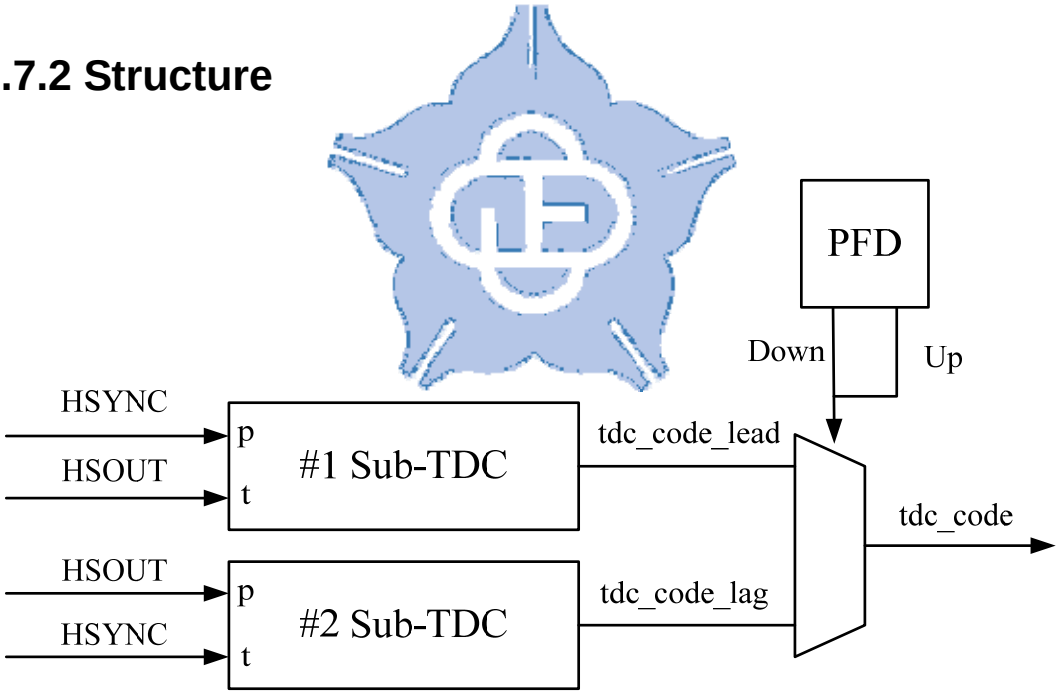


Fig. 2.37 The entirety TDC structure

Fig. 2.37 shows the entire TDC structure. The TDC is composed of two duplicate sub-TDCs and 2-to-1 multiplexer. The first sub-TDC is used to quantize the phase error when HSYNC is leading HSOUT. On the contrary, the second sub-TDC is used to quantize the phase error when HSYNC is lagging HSOUT. Then according to Up

and Down information from the PFD, the multiplexer selects which the outputs of sub-TDCs to be the TDC code (tdc_code).

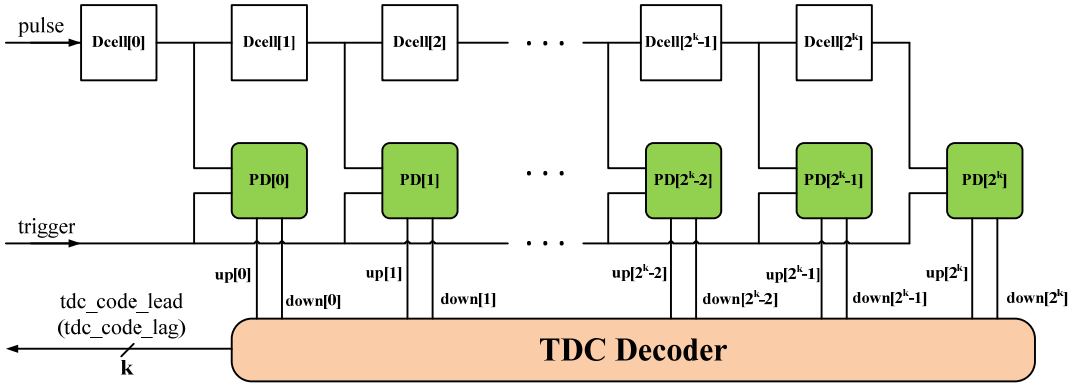


Fig. 2.38 The detailed structure of the sub-TDC

The delay chain of buffers [30] is a well-known method to realize a TDC. However, the delay chain of buffers structure can't quiltize the time interval smaller than a buffer delay. However, the resolution is limited by buffer delay and metastability window of the Flip-Flop. We modify the traditional TDC [30] , the detail circuit of the proposed sub-TDC is shown in Fig. 2.38.

Input signal (pulse) passed through a string of non-inverting delay and the PD samples the output of each delay cell (Dcell) sent to the TDC decoder to generate the TDC code (tdc_code). To solve the problem of metastability window of flip-flop, the TDC replaces the Flip-Flop with phase detector (PD). The PD is discussed in section 2.3.2.2, it can provide very small dead zone thus the resolution of the TDC can be improved. The advantages of the proposed TDC are better recognition rate and have fine resolution.

2.7.3 Problem of TDC Loop Gain and Interpolation-type DCO Solution

TDC can significantly reduce the phase error between HSYNC and HSOUT. But how to use the TDC code (tdc_code) to compensate the phase error is an important issue. This is because the TDC and the DCO is not the same circuit, therefore they will have different resolution with PVT variations. As a result, we must find a way to map the TDC code to the DCO code in the controller, and we cannot directly add the TDC code to the DCO code. It has to determine a suitable gain called TDC loop gain to make the TDC code and the DCO code to be corresponding. In conventional approaches, the suitable TDC loop gain were decided by the circuit simulation and multiplied by the TDC quantification code to as the ideal TDC code.

Based on the above mentioned problems, we do not want to determine the TDC loop gain by circuit simulation. To solve this problem, the proposed TDC uses the delay cell of interpolation-type DCO coarse-tuning stage to as the Dcell of two sub-TDCs circuit. Therefore, the issue of both the TDC circuit and the DCO circuit mismatch problem can be eliminated. Because the DCO is a loop system, the delay line will pass through the positive half cycle and negative half cycle, but the pulse signal in sub-TDCs circuit only passes through a single delay-line one time. Therefore, in the proposed TDC circuit, the value of tdc_code is divided by 2, and the controller can directly use this value to do operation.

2.7.4 Simulation Result

The number of delay cells in sub-TDC circuit is very important. The length of delay chain will affect the entirety power consumption and area. In our work, we use different Dcells to compare which number of Dcell has minimum phase error in four view modes. In simulation parameters, the numbers of Dcell are 256, 128, 64, 32, 16, 8, 4, 2, 1, and turn off TDC, respectively. Table 2.11 and Table 2.12 show the maximum phase error and average phase error in four display modes. The TDC is simulated in UltraSIM SPICE simulation. The simulation result shows that when it uses 16 Dcells to as its delay chain length, the phase error is more suitable in four display modes. Therefore, the proposed ADPLL uses 16 Dcells to form the delay chain of the sub-TDC circuit.

Table 2.11 Maximum phase error of proposed ADPLL in XGA to WUXGA

(ns) 1K Hits	Maximum Phase Error									
	256 Dcell	128 Dcell	64 Dcell	32 Dcell	16 Dcell	8 Dcell	4 Dcell	2 Dcell	1 Dcell	TDC off
XGA	2.25	1.84	2.19	1.90	2.57	5.36	3.20	3.73	4.52	4.52
SXGA	1.33	1.33	1.39	1.37	1.69	2.57	2.77	3.02	3.61	3.84
UXGA	14.07	7.67	4.77	3.16	2.02	2.08	2.74	3.79	4.74	4.74
WUXGA	14.29	7.33	6.26	2.28	1.99	2.50	2.87	3.69	3.88	1.73

Table 2.12 Average phase error of proposed ADPLL in XGA to WUXGA

(ns) 1K Hits	Average Phase Error									
	256 Dcell	128 Dcell	64 Dcell	32 Dcell	16 Dcell	8 Dcell	4 Dcell	2 Dcell	1 Dcell	TDC off
XGA	0.53	0.48	0.53	0.54	0.54	0.59	0.69	0.78	0.98	0.98
SXGA	0.43	0.43	0.43	0.43	0.45	0.51	0.64	0.75	1.01	0.98
UXGA	10.77	5.14	2.08	0.88	0.60	0.51	0.61	0.77	1.08	1.08
WUXGA	8.58	2.06	0.83	0.63	0.50	0.51	0.61	0.77	0.90	0.90

The TDC performance is shown in Table 2.13. In HSPICE simulation, the resolution in TT corner, FF corner, SS corner are 43.496ps, 32.120ps, and 69.078ps, respectively.

Table 2.13 Summary of the TDC performance

	TT Corner	FF Corner	SS Corner
Resolution (ps)	43.496	32.120	69.078
Range (ps)	704.820	518.587	1125.356

Chapter 3 Experimental Results

3.1 Chip Implementation

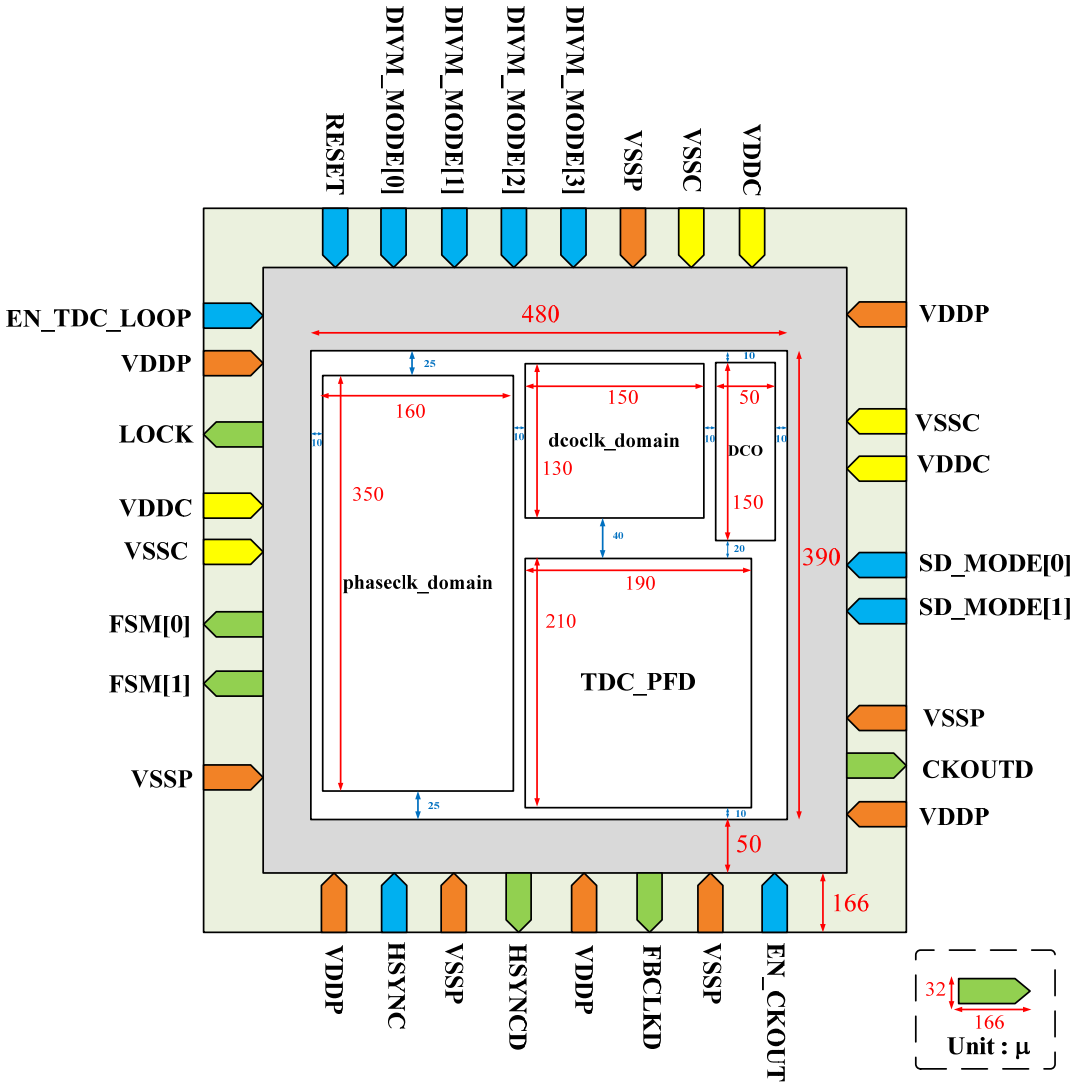


Fig. 3.1 Floorplanning and I/O planning

Fig. 3.1 shows proposed ADPLL chip floorplanning and I/O planning, in the proposed chip 16 I/O PADs and 16 power PADs are used. Table 3.1 is the detail I/O description.

Table 3.1 Table I/O PADs description

Input	Bits	Function	
RESET	1	Set chip to initial	
HSYNC	1	Input clock	
EN_CKOUT	1	Enable pixel clock to output	
EN_TDC_LOOP	1	Enable TDC loop to work	
SD_MODE	2	Set the number bits of SDM fractional code	
		Value	Fractional Code
		0	9 bits Fractional Code
		1	8 bits Fractional Code
		2	7 bits Fractional Code
		3	SDM Turn Off
DIVM_MODE	4	Set the Multiplication Factor of ADPLL	
		Value	Multiplication Factor
		1	XGA 1376
		2	SXGA 1688
		3	UXGA 2160
		4	WUXGA 2592
		5	16
		6	32
		7	64
		8	128
		9	256
		10	512
		11	1024
		12	2048
		13	4096
		14	5600
Output	Bits	Function	
HSYNCD	1	Reference clock	
FBCLKD	1	Feedback clock	
CKOUTD	1	Pixel clock	
FSM	2	Finite state machine state	
LOCK	1	Phase lock signal	
Power Pad	Pairs	Function	
VDDC+VSSC	3	CORE Power Pad	
VDDP+VSSP	5	Pad Power Pad	

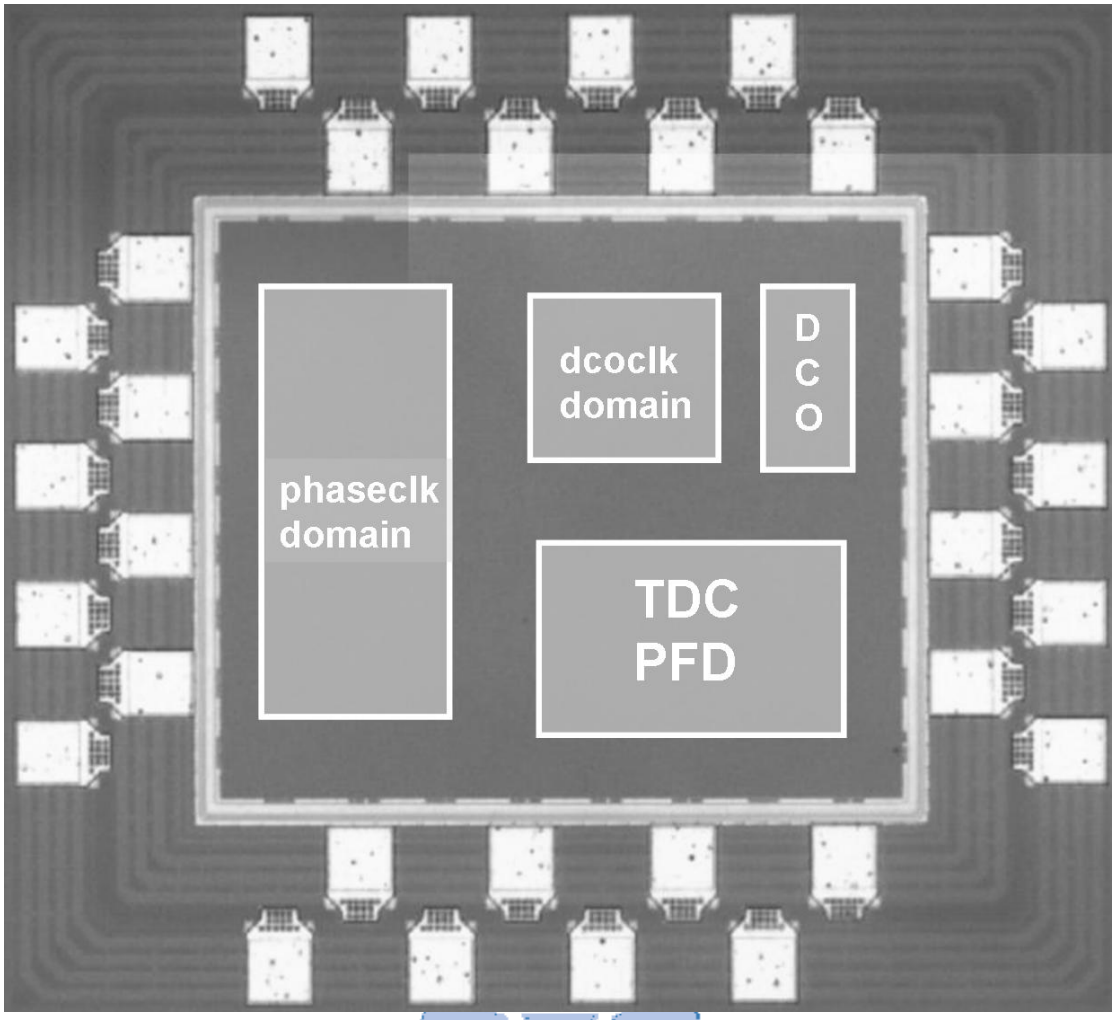


Fig. 3.2 The microphotograph of the proposed ADPLL chip

The microphotograph of the proposed ADPLL chip is shown in Fig. 3.2. This chip is fabricated by UMC 65nm 1P10M standard performance (SP) CMOS process. The chip size is 910×820 μm^2 and the core size is 580×490 μm^2. The layout is divided into four blocks there are phaseclk domain, dcoclk domain, TDCPFD, and DCO, respectively. The phaseclk domain block contains the ADPLL controller and the digital loop filter. The dcoclk domain block contains the decoder of the DCO, the Sigma-Delta Modulator, and the frequency divider. The TDCPFD block contains the time-to-digital converter (TDC), the Phase/Frequency Detector (PFD), and the Phase Detector (PD). The last block is the DCO block, which is placed nearby the dcoclk domain, because the DCO control signals run through the decoder to the DCO block.

3.2 Jitter Behavioral Models Discussion

Because the reference clock (HSYNC) is not a stable clock, the jitter of HSYNC can be as high as 1.06ns [9], and it will affect the overall ADPLL performance. To discuss the influence of the reference clock jitter, four different jitter behavioral models are designed to simulate the actual jitter environment. According to the [9], four different jitter models are designed for worst case, and the peak-to-peak value of the reference clock (HSYNC) is set to ± 1.2 ns. In the following, the jitter behavioral models are divided into four categories.

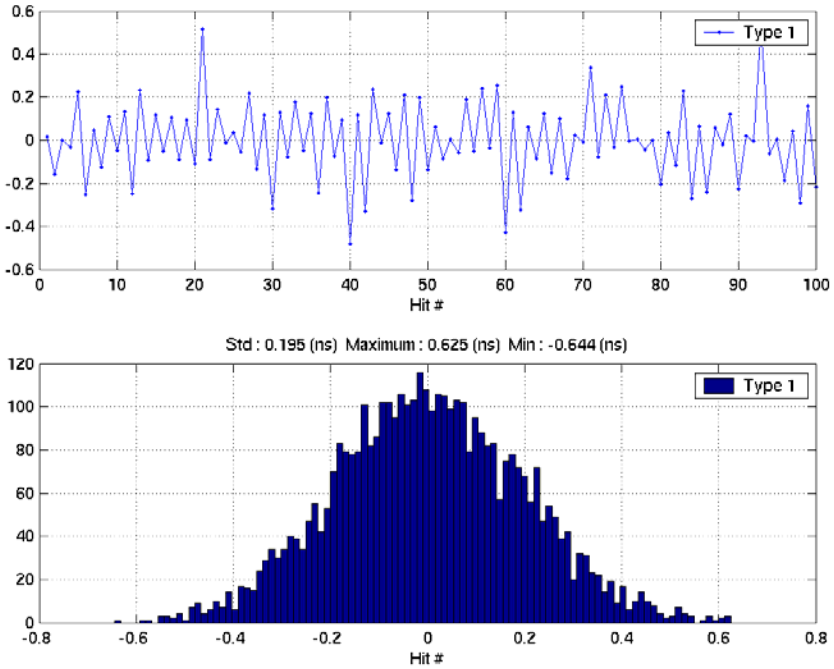
Type 1 : Normal distribution, Fast variation

Type 2 : Normal distribution, Medium variation

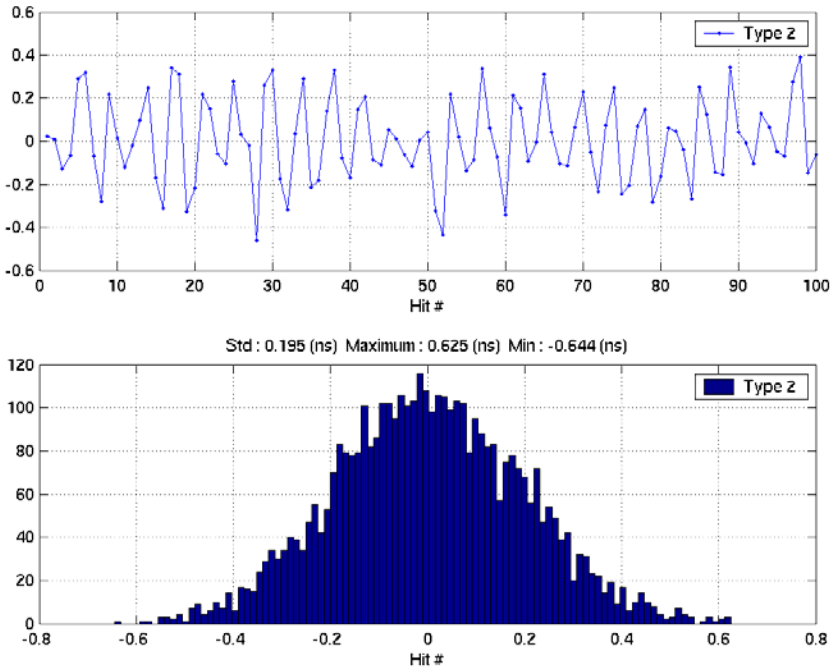
Type 3 : Normal distribution, Slow variation

Type 4 : Uniform distribution, Irregular variation

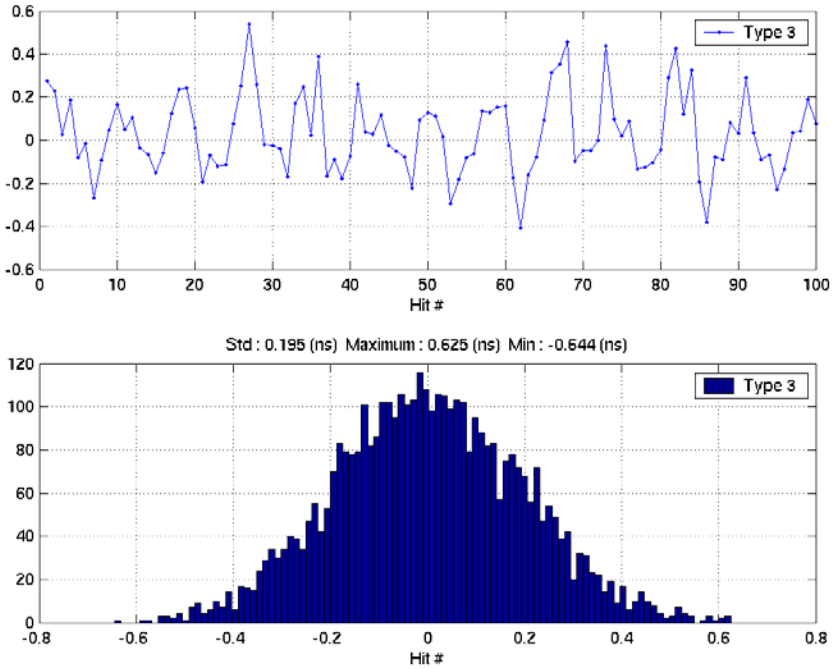
In the following, the "+" symbol represents the value of positive jitter, and the "-" symbol represents the value of negative jitter, respectively. In Type 1, the jitter variations is drastic changed and its jitter form is (+,-,+,-,…,+,-). In Type 2, the jitter form is (+,+,-,-,+,+,…,-,-). In Type 3, the jitter variations changes slowly and its jitter form is (+,+,+,+,-,-,-,-,+,+,+,+,…,-,-,-,-). In Type 4, the jitter is a irregular form. From Type 1 to Type 3, these distributions of jitter are normal distribution, but in Type 4 is uniform distribution. Fig. 3.3 shows the jitter histogram and jitter distribution. The proposed ADPLL uses this four type jitter models to do circuit simulation, and the simulation result is shown in Section 3.3.1.1.



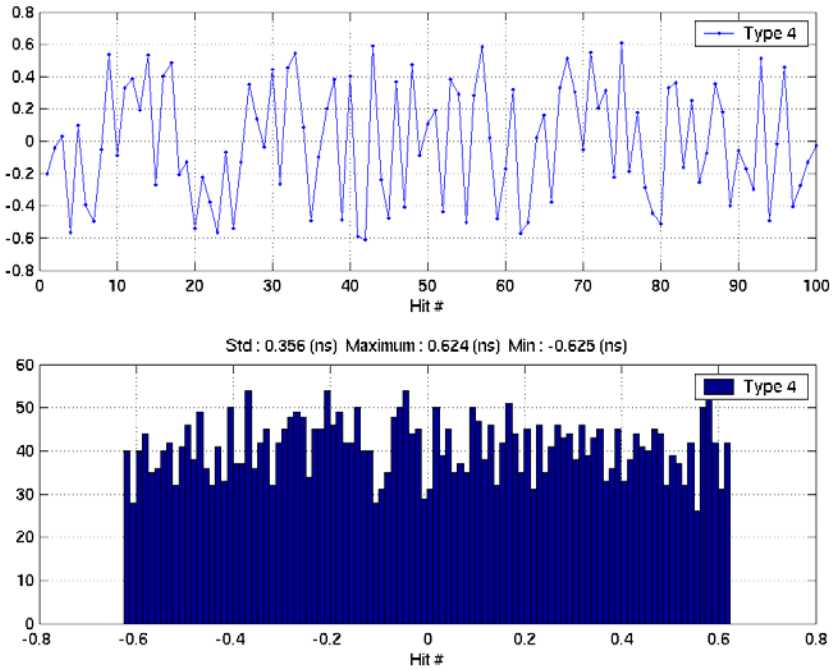
(a) Type 1, Normal distribution, Fast variation



(b) Type 2, Normal distribution, Medium variation



(c) Type 3, Normal distribution, Slow variation



(d) Type 4, Uniform distribution, Irregular variation

Fig. 3.3 Different jitter models and its distribution

3.3 Overall Simulation

Table 3.2 The specification of proposed ADPLL

Mode	HSYNC Frequency (kHz)	Multiplication Factor	Pixel Clock (PIXEL_CLK) Frequency (MHz)
XGA	68.677	1376	94.500
SXGA	79.976	1688	135.000
UXGA	75.000	2160	192.000
WUXGA	74.556	2592	193.250

Table 3.2 shows the specification of proposed ADPLL. In proposed ADPLL, it will use these four modes to do circuit simulation, and these modes are XGA, SXGA, UXGA, and WUXGA, respectively.

3.3.1 Simulation in Verilog Behavior Model

3.3.1.1 Different Jitter Behavioral Models

The phase error performance of proposed ADPLL is simulated in four jitter models. If we assume that the proposed ADPLL parameters are TDC on and SDM on with 1.2ns jitter. Fig. 3.4 and Fig. 3.5 show the comparison of both maximum and average phase error in different jitter behavioral models. However, from Fig. 3.4 to Fig. 3.5, the Type 4 jitter behavioral model has the worst performance in maximum and average phase error. The detailed information is listed in Table 3.3. Actually, the real HSYNC jitter environment is similar to Type 2 jitter behavioral model. Therefore, the phase error performance of proposed ADPLL is controlled less than 30% of the pixel clock period with Type 1 to Type 3 jitter behavioral models. But proposed ADPLL will simulate on the worst case by using the Type 4 jitter behavioral model.

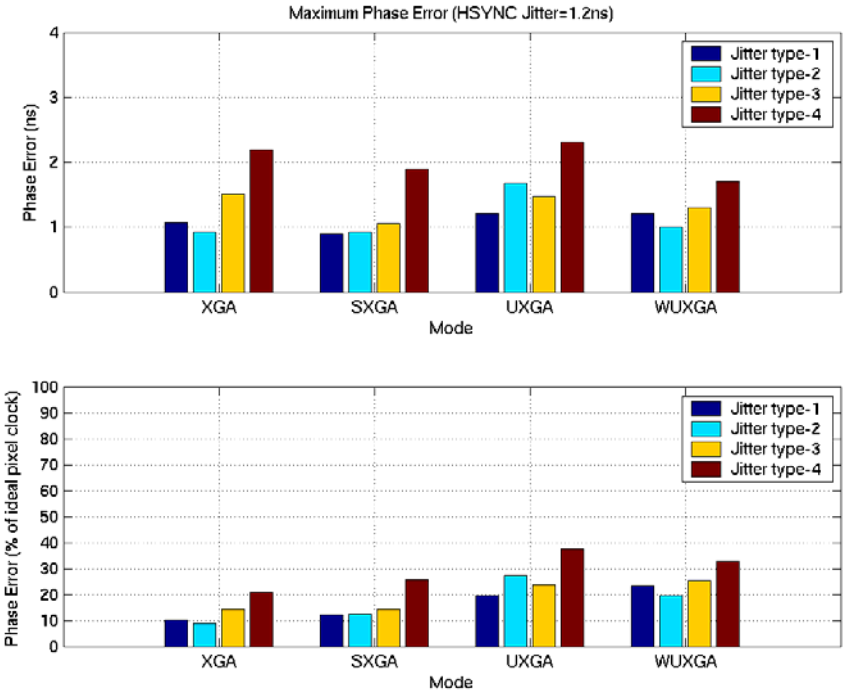


Fig. 3.4 The comparison of the maximum phase error in different jitter models

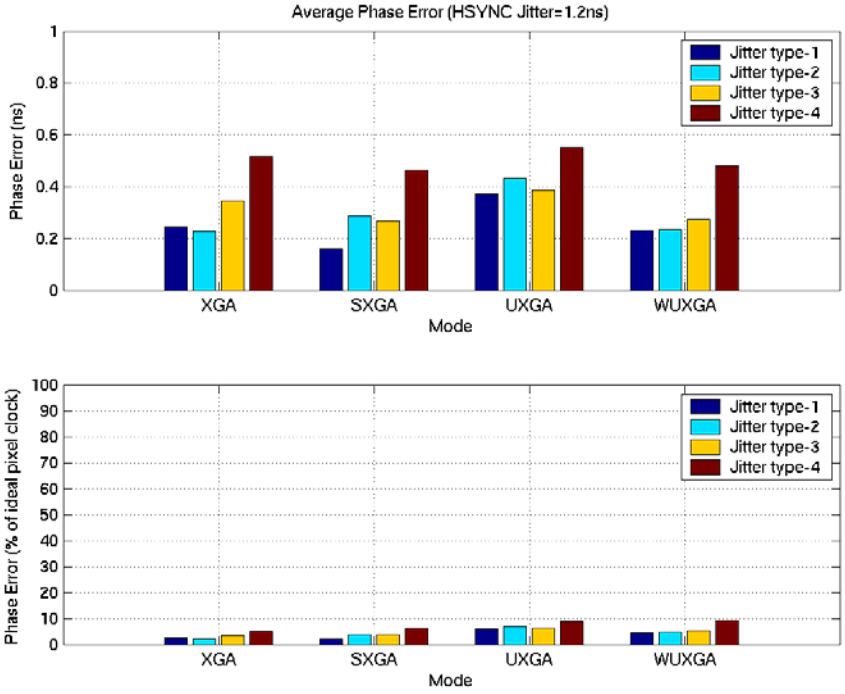


Fig. 3.5 The comparison of the average phase error in different jitter models

Table 3.3 Maximum and average phase error in different jitter behavioral models

Mode	Maximum Phase Error							
	Type 1		Type 2		Type 3		Type 4	
	(ns)	(%)	(ns)	(%)	(ns)	(%)	(ns)	(%)
XGA	1.063	10.05	0.931	8.80	1.512	14.29	2.189	20.69
SXGA	0.897	12.11	0.931	12.57	1.056	14.26	1.898	25.62
UXGA	1.213	19.65	1.673	27.10	1.464	23.72	2.317	37.53
WUXGA	1.211	23.40	1.009	19.50	1.304	25.20	1.699	32.83
Mode	Average Phase Error							
	Type 1		Type 2		Type 3		Type 4	
	(ns)	(%)	(ns)	(%)	(ns)	(%)	(ns)	(%)
XGA	0.245	2.32	0.228	2.15	0.345	3.26	0.515	4.87
SXGA	0.161	2.17	0.286	3.86	0.267	3.60	0.466	6.29
UXGA	0.371	6.01	0.432	7.00	0.387	6.27	0.553	8.96
WUXGA	0.232	4.48	0.235	4.54	0.275	5.31	0.480	9.28

3.3.1.2 Time-to-Digital Converter

From Fig. 3.6 to Fig. 3.7 show the maximum phase error and average phase error between HSYNC and HSOUT without TDC. As well, in Fig. 3.8 and Fig. 3.9 show the result with TDC. In four figures, the left half shows the phase error versus different input jitters, and the right half shows the percentage of ideal pixel clock period versus different input jitters.

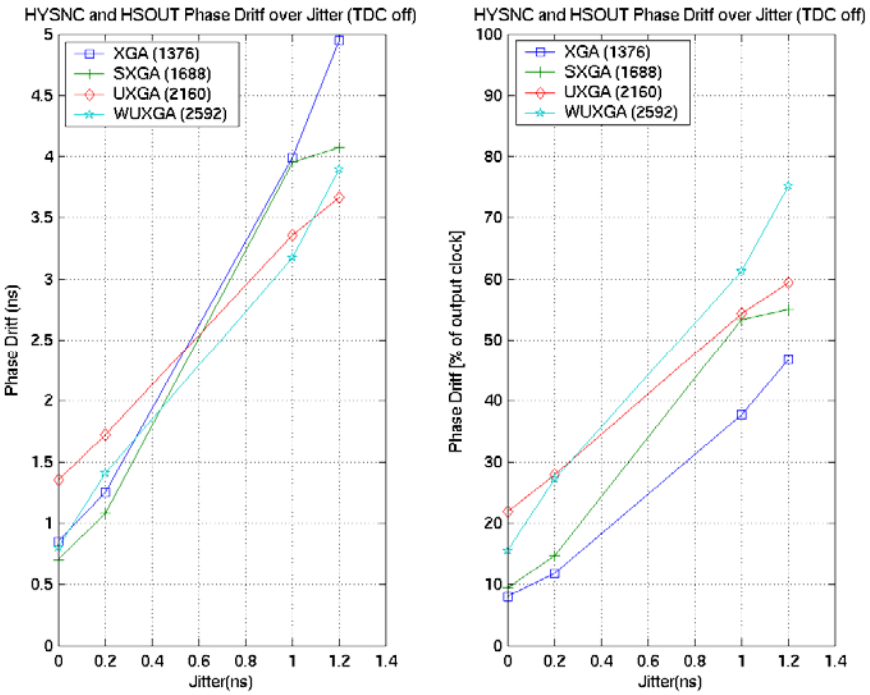


Fig. 3.6 Without TDC, the maximum phase error.

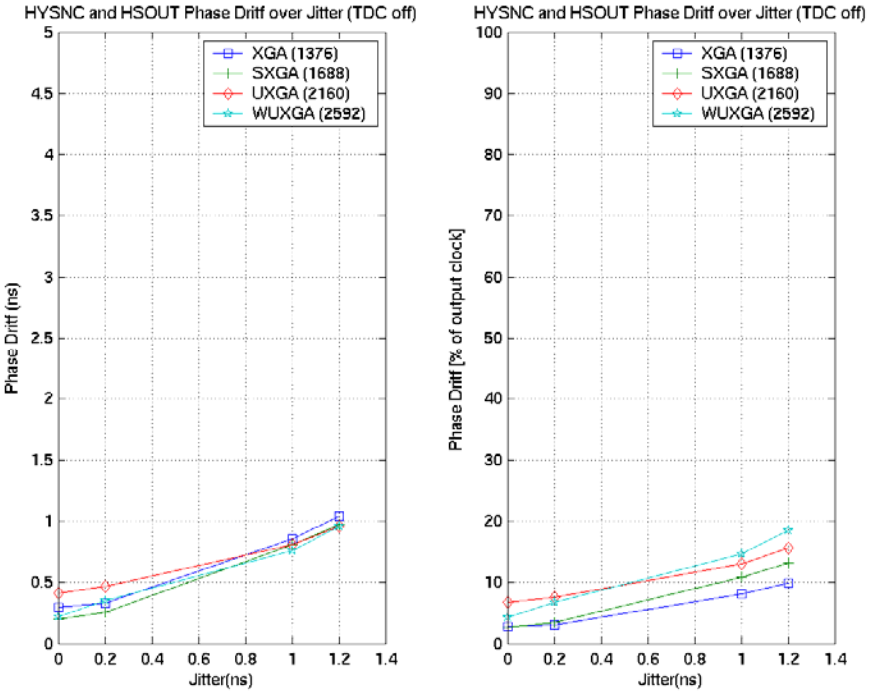


Fig. 3.7 Without TDC, the average phase error.

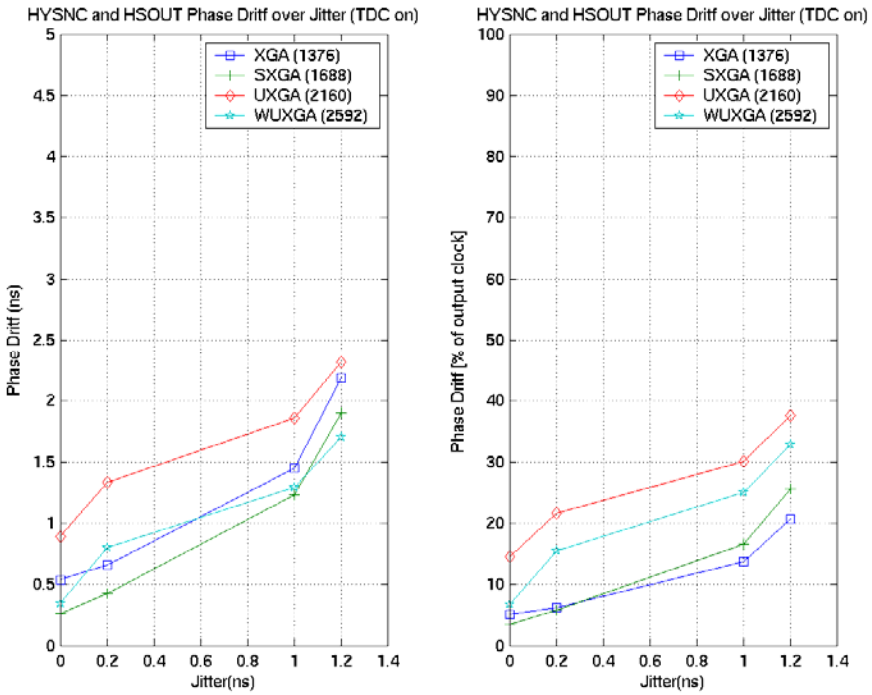


Fig. 3.8 With TDC, the maximum phase error.

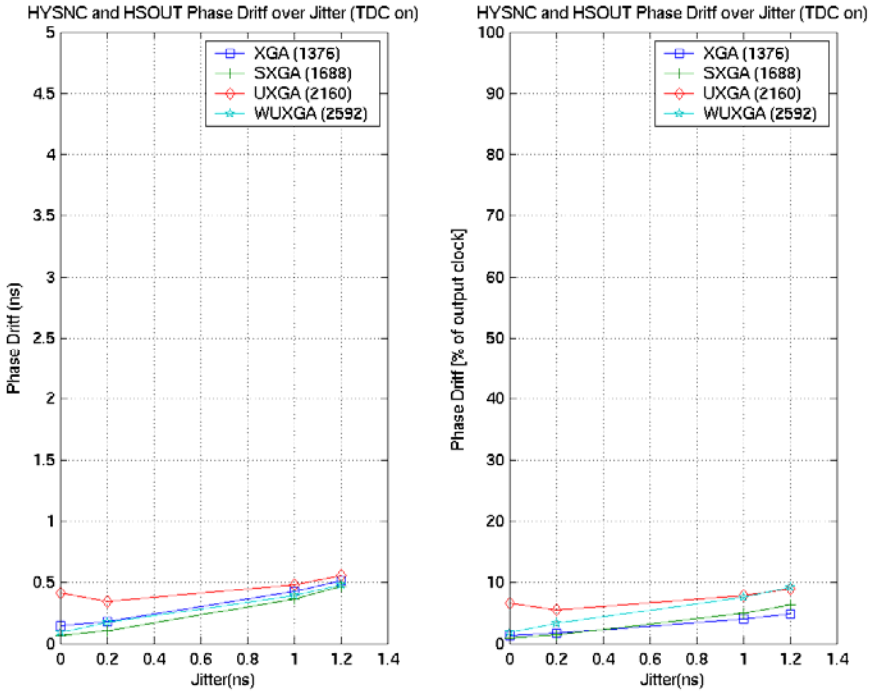


Fig. 3.9 With TDC, the average phase error.

The detailed information is listed in Table 3.4 and Table 3.5. Table 3.4 and Table 3.5 show the maximum phase error and the average phase error with and without TDC. In each table, the shadowed rows represent the system with TDC and unshadowed rows represent the system without TDC. For each data followed in parentheses represents the percentage of the ideal pixel clock. In WUXGA mode and without TDC, the maximum phase error is 3.90ns, and with TDC the maximum phase error is 1.70ns. Therefore, when the proposed ADPLL adopts the TDC to compensate the phase error, the phase error can be improved almost 45%, reduced from 75.29% to 32.89%.

Table 3.4 Maximum phase error with and without TDC in XGA to WUXGA

(ns)	TDC	Jitter 1.2 ns	Jitter 1.0 ns	Jitter 0.2 ns	Jitter 0.0 ns
XGA	OFF	4.95 (46.78%)	3.99 (37.72%)	1.25 (11.80%)	0.85 (8.03%)
	ON	2.19 (20.69%)	1.45 (13.68%)	0.66 (6.24%)	0.54 (5.08%)
SXGA	OFF	4.08 (55.04%)	3.95 (53.35%)	1.08 (14.61%)	0.70 (9.45%)
	ON	1.90 (25.62%)	1.23 (16.58%)	0.43 (5.76%)	0.26 (3.51%)
UXGA	OFF	3.67 (59.37%)	3.36 (54.43%)	1.73 (27.94%)	1.35 (21.89%)
	ON	2.32 (37.53%)	1.86 (30.12%)	1.34 (21.63%)	0.89 (14.47%)
WUXGA	OFF	3.90 (75.29%)	3.18 (61.35%)	1.41 (27.23%)	0.81 (15.54%)
	ON	1.70 (32.89%)	1.30 (25.04%)	0.80 (15.48%)	0.35 (6.69%)

Table 3.5 Average phase error with and without TDC in XGA to WUXGA

(ns)	TDC	Jitter 1.2 ns	Jitter 1.0 ns	Jitter 0.2 ns	Jitter 0.0 ns
XGA	OFF	1.04 (9.86%)	0.85 (8.07%)	0.33 (3.10%)	0.30 (2.81%)
	ON	0.51 (4.86%)	0.42 (4.01%)	0.18 (1.69%)	0.14 (1.35%)
SXGA	OFF	0.97 (13.14%)	0.81 (10.91%)	0.25 (3.41%)	0.20 (2.72%)
	ON	0.47 (6.29%)	0.37 (4.94%)	0.11 (1.47%)	0.07 (0.88%)
UXGA	OFF	0.96 (15.56%)	0.81 (13.08%)	0.47 (7.59%)	0.41 (6.69%)
	ON	0.55 (8.96%)	0.48 (7.83%)	0.34 (5.58%)	0.41 (6.67%)
WUXGA	OFF	0.96 (18.55%)	0.76 (14.67%)	0.35 (6.75%)	0.22 (4.28%)
	ON	0.48 (9.27%)	0.39 (7.59%)	0.17 (3.31%)	0.09 (1.80%)

The comparison results of the ADPLL with and without TDC are shown in bar chart of Fig. 3.10 and Fig. 3.11. In the top half of both figures, these show the maximum phase error and the average phase error, and in the bottom of both figures show the percentage of ideal pixel clock period. The assumption jitter of this simulation is ± 1.2 ns. When the proposed ADPLL uses TDC to compensate the phase error, the performance of proposed ADPLL can be improved a lot.

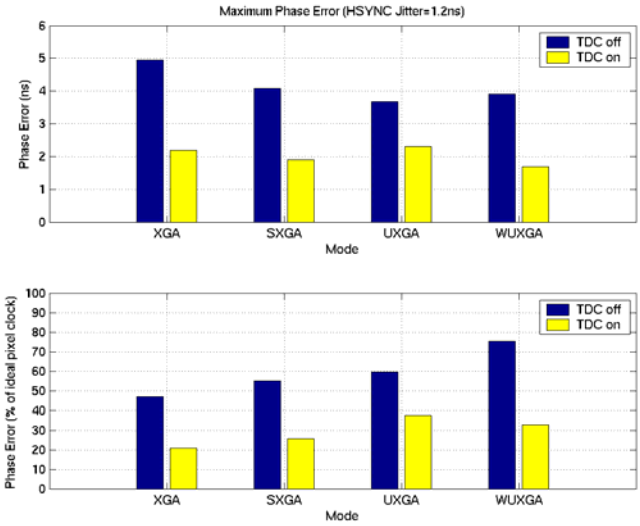


Fig. 3.10 Maximum phase error of proposed ADPLL with and without TDC

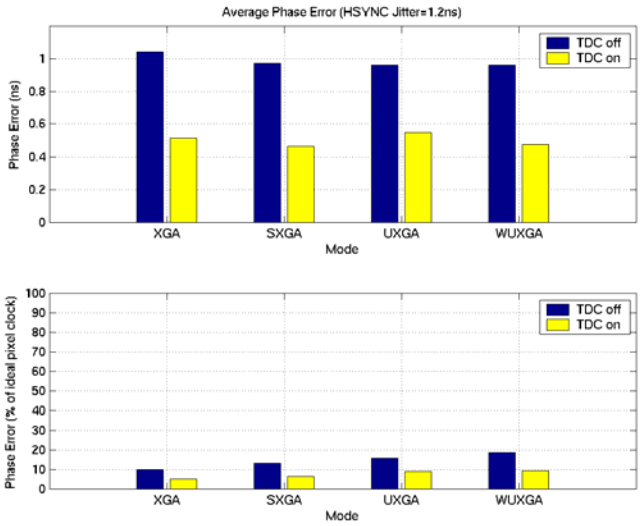


Fig. 3.11 Average phase error of proposed ADPLL with and without TDC

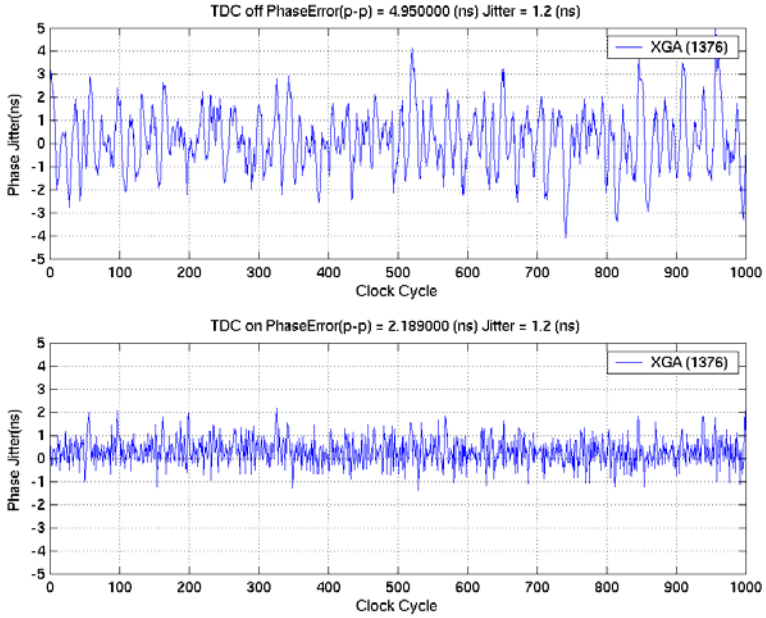


Fig. 3.12 Jitter performance with and without TDC in XGA mode

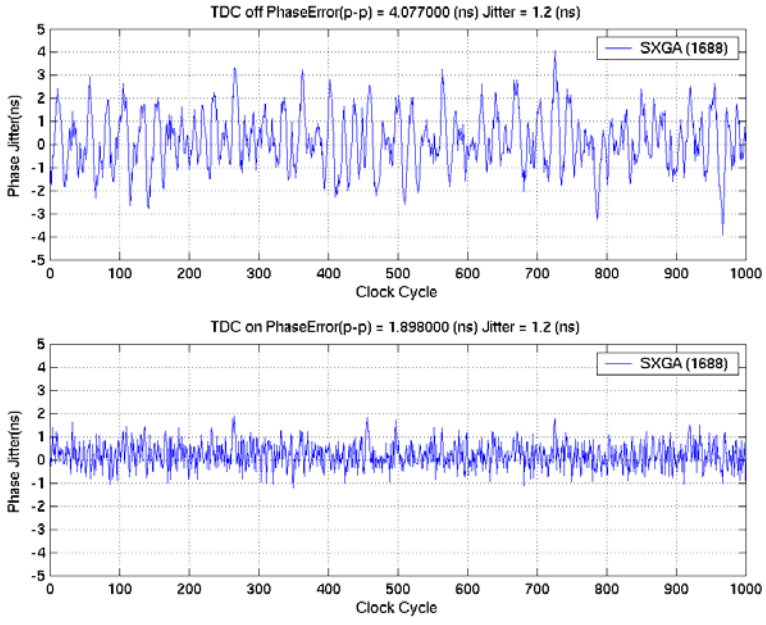


Fig. 3.13 Jitter performance with and without TDC in SXGA mode

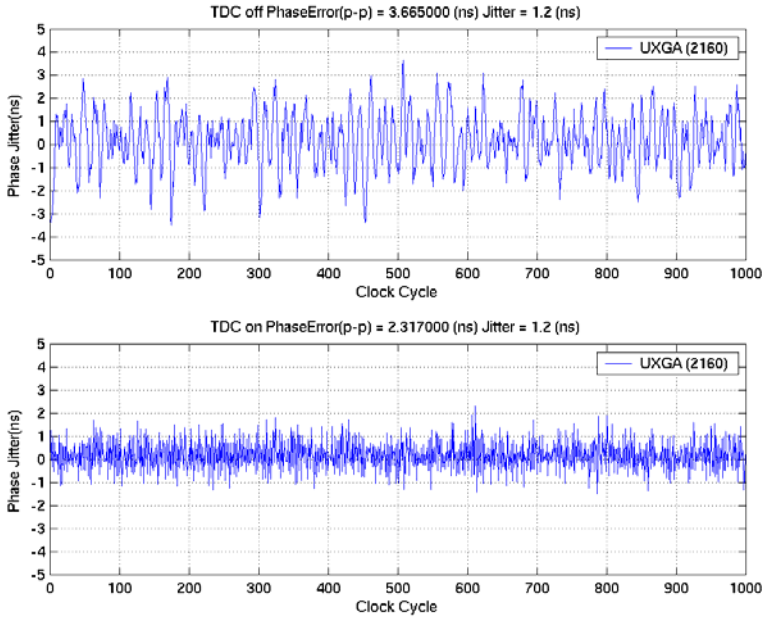


Fig. 3.14 Jitter performance with and without TDC in UXGA mode

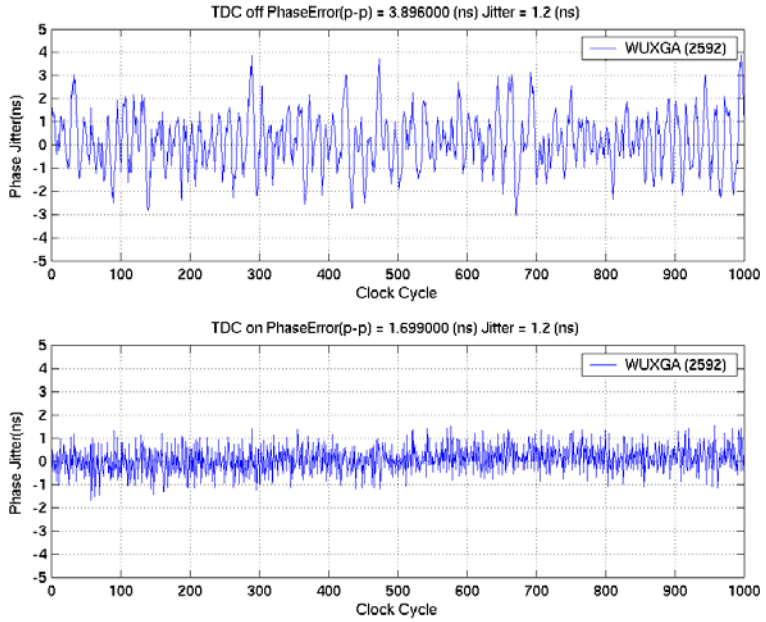


Fig. 3.15 Jitter performance with and without TDC in WUXGA mode

From Fig 3.12 to Fig. 3.15, these shows the tracking jitter performance of proposed ADPLL loop in different modes. The x axis is the clock cycle count and the y axis is the phase error.

3.3.1.3 Sigma-Delta Modulator

The influence of proposed ADPLL with and without SDM is simulated in different display mode (XGA to WUXGA). In the top half of both Fig. 3.16 and Fig. 3.17, these show the maximum phase error and average phase error, and in the bottom of both figures show the percentage of the ideal pixel clock period.

We assume that the proposed ADPLL turns on TDC, with 1.2ns jitter, with Type 4 jitter behavioral model. From Fig. 3.16, in XGA view mode and without the SDM, the phase error is 7201ns (68054%), and it is decreased to 2.189ns (20.69%) with the SDM. When the proposed ADPLL uses SDM, the phase error has been extensively decreased. We can see that if SDM DCO is not used, the phase error can't be tracked correctly. The detailed information is listed in Table 3.6.

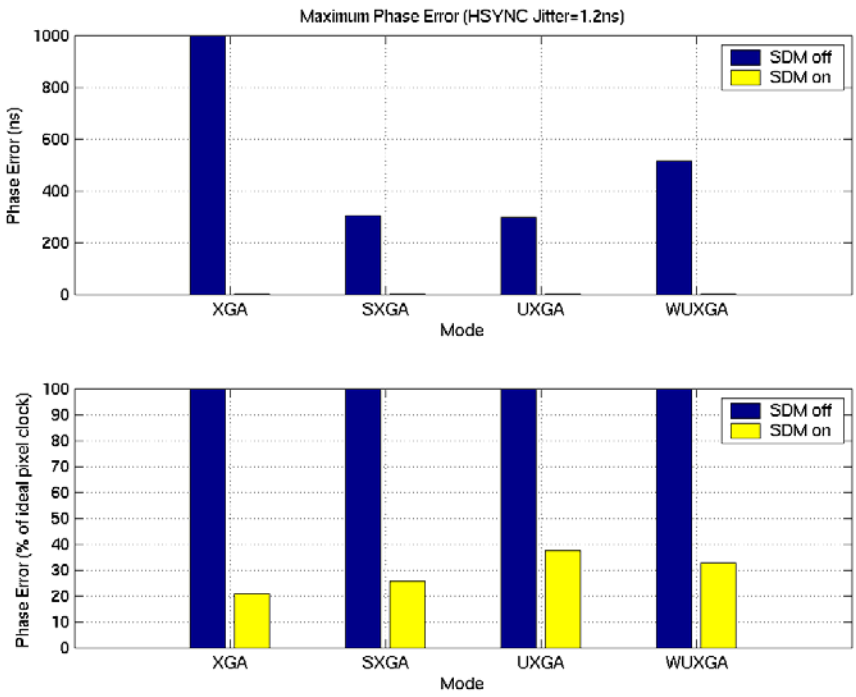


Fig. 3.16 Maximum phase error of the proposed ADPLL with and without SDM

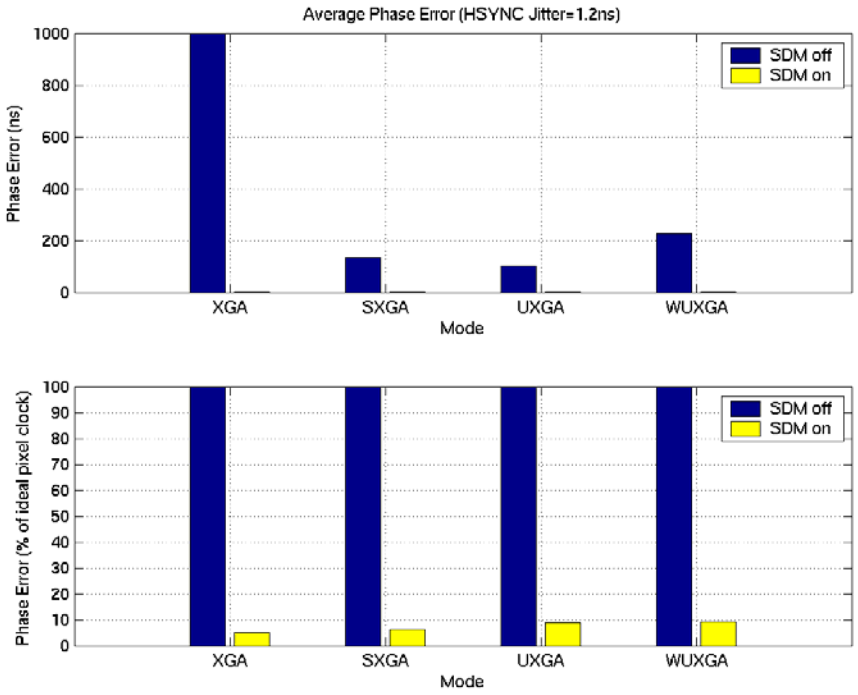


Fig. 3.17 Average phase error of the proposed ADPLL with and without SDM

Table 3.6 Maximum and average phase errors with and without SDM

Mode	Maximum Phase Error				Average Phase Error			
	SDM OFF		SDM ON		SDM OFF		SDM ON	
	(ns)	(%)	(ns)	(%)	(ns)	(%)	(ns)	(%)
XGA	7201.57	68054	2.189	20.69	2825.34	26699	0.515	4.87
SXGA	303.617	4099	1.898	25.62	133.002	1795	0.466	6.29
UXGA	299.803	4856	2.317	37.53	102.247	4856	0.553	8.96
WUXGA	515.874	9968	1.699	32.83	228.509	4415	0.480	9.28

3.3.2 Simulation in AMS

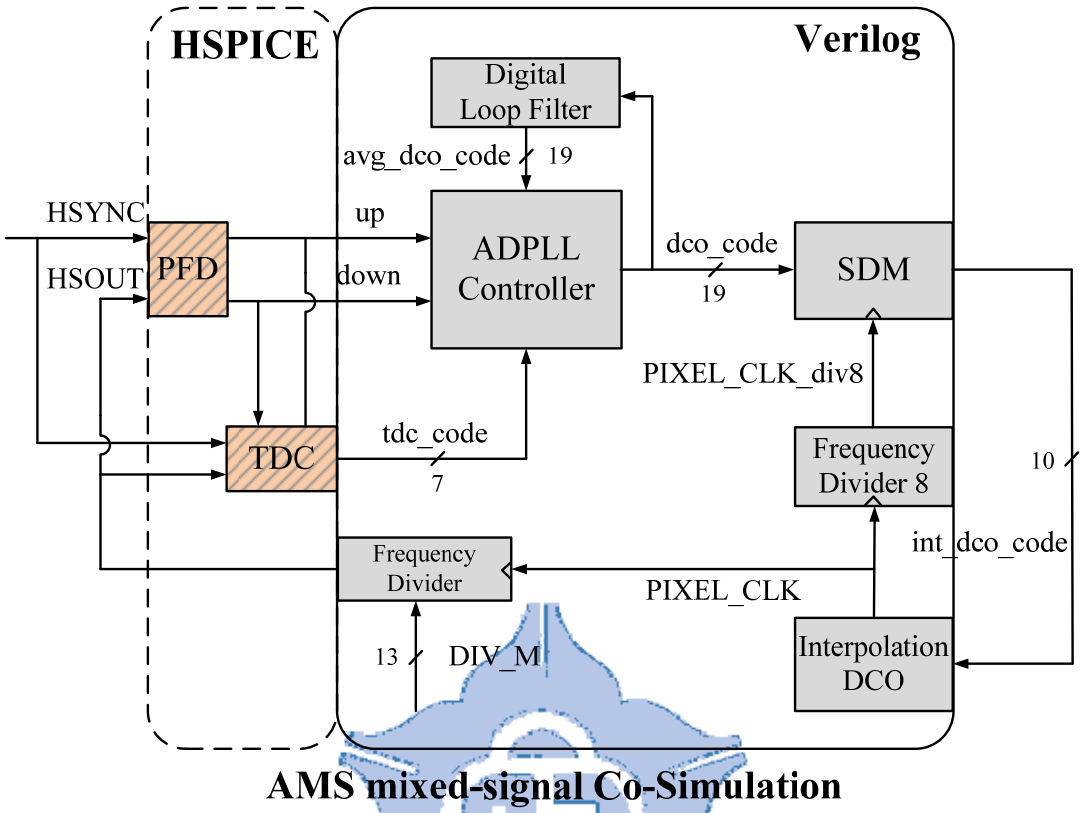


Fig. 3.18 Simulation mode of the proposed ADPLL

Table 3.7 Simulation Mode of the proposed ADPLL components

ADPLL Components	Simulation Mode
ADPLL Controller	Verilog
Sigma-Delta Modulation	Verilog
Digital Loop Filter	Verilog
Interpolation DCO	Verilog
Frequency Divider	Verilog
PFD	HSPICE
TDC	HSPICE

Because the reference clock frequency of proposed ADPLL is very low (68.677kHz to 74.556kHz), and the maximum output frequency of the DCO is up to 193.250MHz. In the circuit simulation, in order to accommodate the DCO simulation accuracy, the overall ADPLL simulation time will become very long and unacceptable. Therefore, a mixed-mode simulation is needed, the digital circuit and DCO use Verilog simulation, TDC, and PFD are used SPICE to do circuit simulation. The proposed ADPLL uses the Cadence AMS-Ultra simulator to do mixed-signal co-simulation to speed up the simulation time. Fig. 3.18 shows the simulation mode of the proposed ADPLL components and Table 3.7 shows the detailed simulation information. In AMS mixed-signal co-simulation, we divide the ADPLL components into HSPICE mode and Verilog mode. In the proposed AMS simulation, TDC, and PFD are simulated by SPICE and the other digital circuits and the DCO are simulated by NC-Verilog. The DCO is the most time consuming block in this AMS simulation, so the DCO must be simulated by NC-Verilog, or the simulation will become too long. This is because the DCO operation frequency is up to 193.250MHz, but the other circuit operates at reference clock rate which is hundred kHz. In order to maintain the accuracy of the DCO, we use the DCO post-layout simulation results to create the frequency look up table. Therefore, we can still have enough accuracy for DCO circuit in AMS mixed-signal co-simulation. We assume that the proposed ADPLL turns on both TDC and SDM, with 1.2ns jitter and with Type 4 jitter behavioral model.

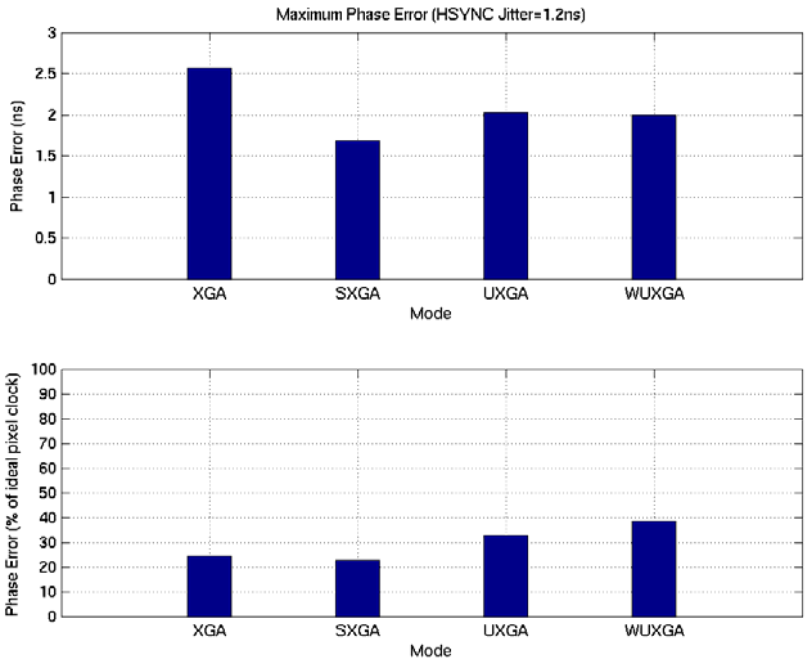


Fig. 3.19 The maximum phase error in AMS simulation

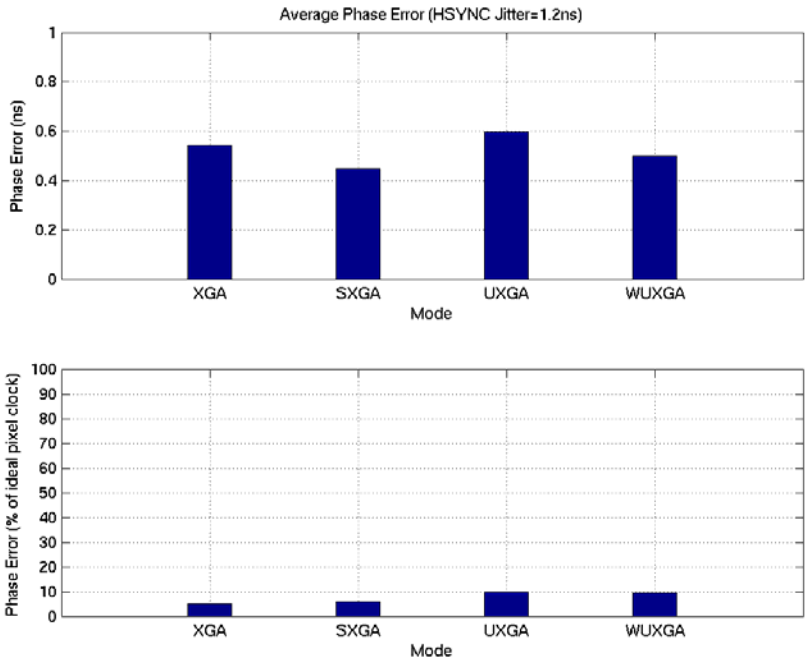


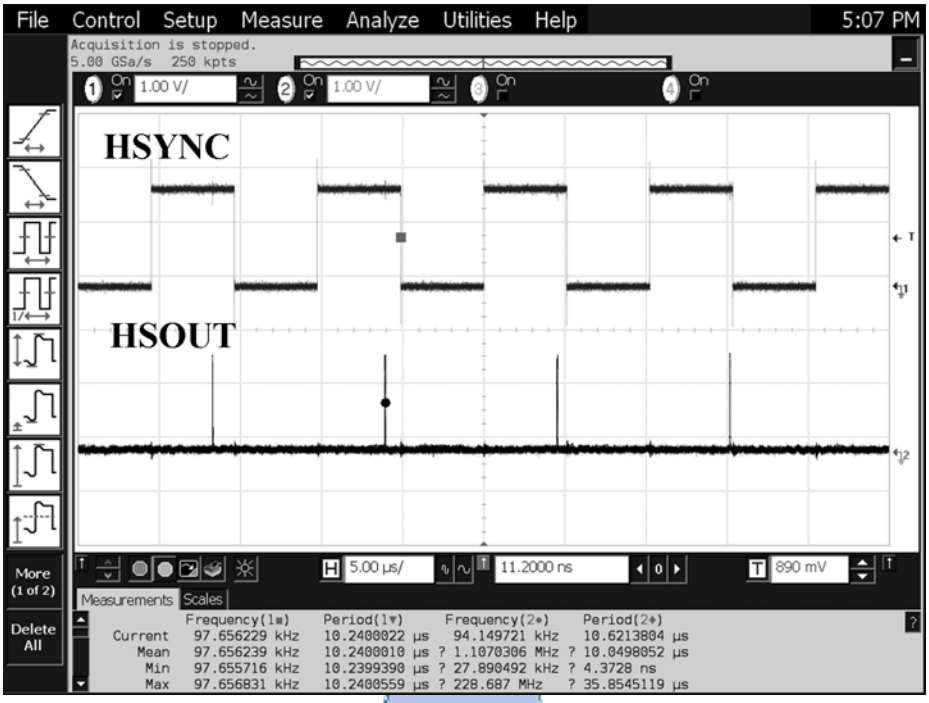
Fig. 3.20 The average phase error in AMS simulation

Table 3.8 Maximum and average phase errors in AMS simulation

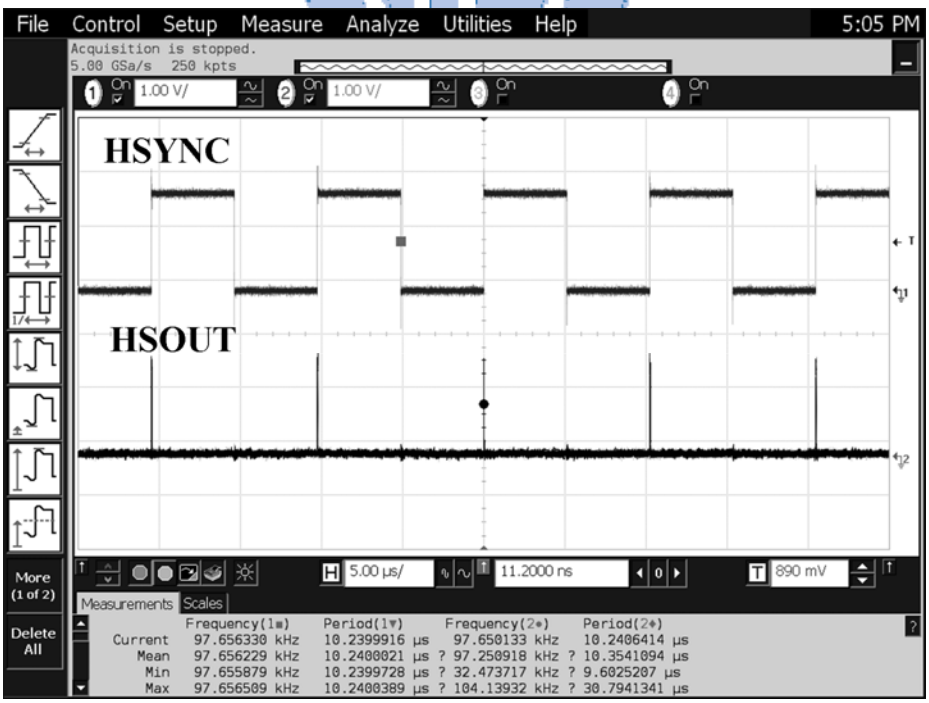
Mode	Maximum Phase Error (ns)		Average Phase Error (ns)	
	(ns)	(%)	(ns)	(%)
XGA	2.571	24.30%	0.541	5.11%
SXGA	1.685	22.75%	0.447	6.03%
UXGA	2.020	32.72%	0.598	9.69%
WUXGA	1.991	38.47%	0.500	9.66%

Fig. 3.18 and Fig 3.19 show the result of maximum and average phase error in AMS simulation, and Table 3.7 shows the detailed information. In Fig. 3.18, the maximum phase error in four modes are 2.571ns (24.30%), 1.685ns (22.75%), 2.020ns (32.72%), and 1.991ns (38.47%), respectively. Although the performance of the maximum phase error beyond one third of the pixel clock period requirement in both UXGA mode and WUXGA mode, but the performance of the average phase error is controlled in 0.500ns (9.66%) in WUXGA mode. To compare the simulation result between AMS simulation (Table 3.7) and Verilog simulation (Table 3.4 and Table 3.5), in AMS simulation, the average phase error performance is better than Verilog simulation, and the maximum phase error performance in both simulation are almost the same.

3.4 Chip Measurement

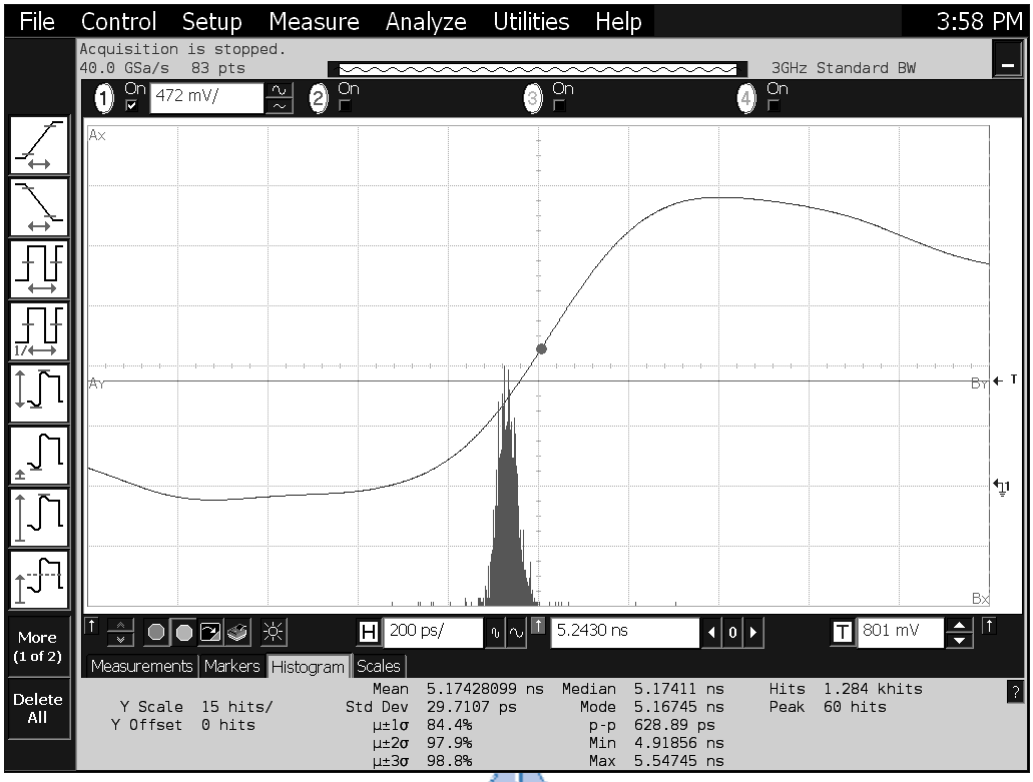


(a) Without SDM

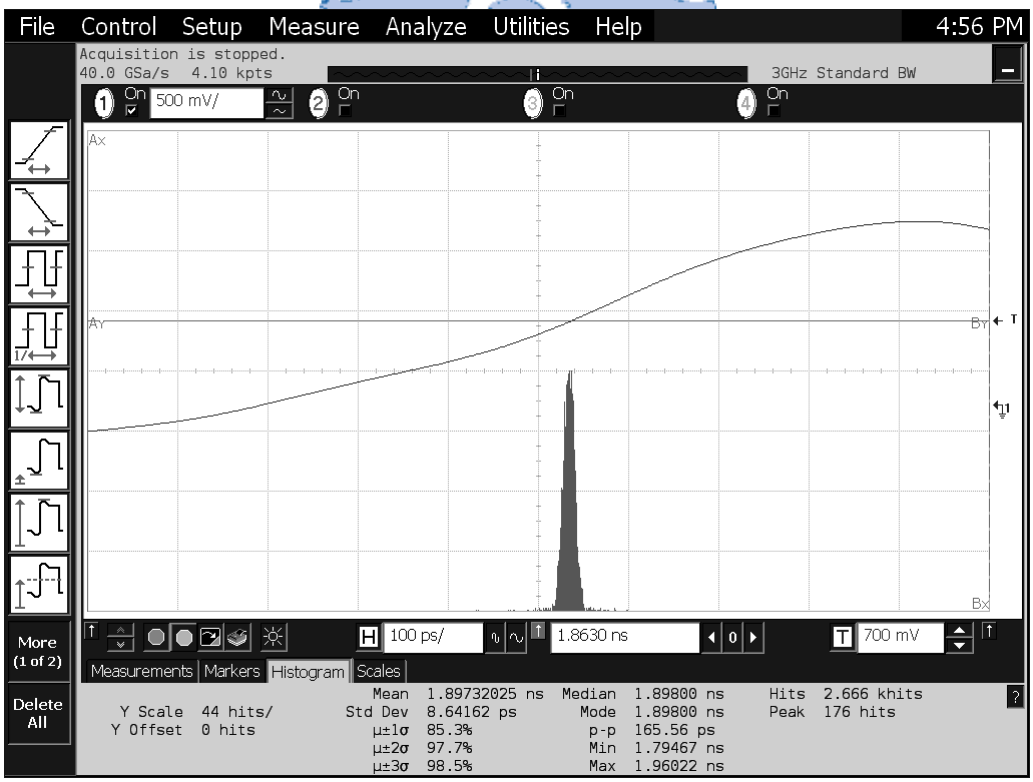


(a) With SDM

Fig. 3.21 The locking procedure of the ADPLL in 5600 multiplication factor



(a) The jitter histogram of the pixel clock in WUXGA mode (@193.26MHz)



(b) The jitter histogram of the pixel clock in 5600 multiplication factor @527.06MHz)

Fig. 3.22 The jitter histogram of the proposed ADPLL

Fig. 3.21 shows the locking procedure of the proposed ADPLL in 5600 multiplication factor in Fast Phase Tracking state. In Fig. 3.21(a), the proposed ADPLL turns off the SDM, and in Fig. 3.21(b) the proposed ADPLL turns on the SDM. If the ADPLL turns off the SDM, the ADPLL has a large phase error between HSYNC and HSOUT. However, when the proposed turns on the SDM, the HSOUT almost aligns HSYNC.

Fig. 3.22 shows the jitter measurement result of the pixel clock. The HSYNC signal in our measurement environment is very noisy. Both root-mean-square (rms) and peak-to-peak jitter of HSYNC signal are 39.03ps and 391.08ps respectively. The HSYNC jitter will affect the overall ADPLL performance. In Fig. 3.22(a), it shows measured jitter histogram operates at 193.26MHz in WUXGA mode. The rms jitter is 29.71ps. In Fig. 3.22(b), it shows the measured jitter histogram operates at 527.06MHz in 5600 multiplication factor. The rms jitter is 8.64ps.

Table 3.9 Measurement Results of the proposed ADPLL

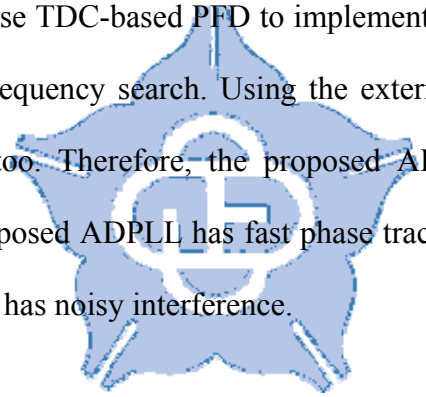
Frequency Multiplication Factor	1376 (XGA)	1688 (SXGA)	2160 (UXGA)	2592 (WUXGA)	5600 (TEST)
HSYNC Period (μs)	14.56	12.50	13.33	13.41	10.24
HSYNC Freq. (kHz)	68.68	79.98	75.00	74.56	97.66
Pixel Clock Period (ns)	11.09	7.41	6.17	5.17	1.90
Pixel Clock Freq. (MHz)	90.14	134.97	161.98	193.26	527.06
Pixel Clock $Jitter_{rms}$ (ps)	78.31	41.12	33.94	29.71	8.64

Table 3.9 summarizes the measurement results of the proposed ADPLL. We measure five different modes, the multiplication factor are 1376, 1688, 2160, 2592, and 5600 respectively. The rms jitter in five different modes are 78.31ps, 41.12ps, 33.94ps, 29.71, and 8.64ps respectively. In XGA mode, the pixel clock frequency is 90.14MHz, and in TEST mode is up to 527.06MHz.

Table 3.10 Performance Comparisons

Performance Indices	**Proposed**	**TVLSI'09[32]**	**JSSC'06 [11]**
Process	65nm CMOS	0.18μm CMOS	0.18μm CMOS
Approach	All-Digital	All-Digital	All-Digital
Phase Align	TDC-Based PFD	Bang-bang PFD	No
Area	0.07mm^2	0.14mm^2	0.16mm^2
Power	0.848mW (@193MHz) 1.813mW (@520MHz)	26.7mW (@600MHz)	15mW (@378MHz)
Input Range	35.71kHz~12.5MHz	30.3kHz~100MHz	19.26kHz~60MHz
Output Range	90.14~527.06MHz	62~616MHz	2.4~378MHz
Multiplication Factor	16~5600	1~2046	4~13888
$Jitter_{RMS}$	78.31ps (@90.14MHz) 8.64ps (@527.06MHz)	7.28ps (@600MHz)	76ps (@134.77MHz)
Performance Indices	**JSSC'04 [31]**	**JSSC'03 [12]**	**[10]**
Process	0.6μm CMOS	0.35μm CMOS	0.13μm CMOS
Approach	Mixed-Mode	All-Digital	Mixed-Mode
Phase Align	TDC-Based PFD with external crystal	Bang-bang PFD	TDC-Based PFD with external crystal
Area	1.8mm^2	0.71mm^2	0.2mm^2
Power	180mW	100mW (@500MHz)	N/A
Input Range	N/A	N/A	N/A
Output Range	10~80MHz	45~510MHz	Max.1GHz
Multiplication Factor	N/A	1~255	1~4096
$Jitter_{RMS}$	40ps (@78MHz)	22ps (@ 450MHz)	32.4ps (@78.73MHz)

Table 3.10 summarizes the proposed ADPLL chip performance. In Table 3.10, [32] uses the bang-bang PFD to implement the ADPLL. This ADPLL only uses leading or lagging information from PFD to compensate the phase error. However, it assumes that the HSYNC is ideal, it doesn't have reference jitter to affect the ADPLL performance. In Fig. 3.6, if peak-to-peak HSYNC jitter is up to 1.2ns and without TDC, [32] will have large phase error. So, this approach doesn't have phase tracking ability when the reference signal has jitter. But in the proposed ADPLL, if we assume that the ADPLL doesn't have HSYNC jitter interference (0ns HSYNC jitter) and turns off the TDC, the proposed ADPLL still have phase tracking ability. In [11][12], such ADPLLs are only frequency synthesizer, they don't have phase tracking ability. In [10][31], those ADPLLs use TDC-based PFD to implement ADPLL, but they have to use external crystal for frequency search. Using the external crystal means that the design cost is increase, too. Therefore, the proposed ADPLL is implemented by TDC-based PFD. The proposed ADPLL has fast phase tracking ability and has small phase error when HSYNC has noisy interference.



Chapter 4 Conclusion and Future Works

In this dissertation, we proposed a fast phase tracking and high frequency multiplication factor ADPLL.

The interpolation-type DCO is used to solve the DCO non-monotonic problem and to solve the problem of TDC loop gain. Therefore, we can control the DCO directly than using the built-in self-calibration circuit.

The proposed ADPLL uses dithering technology to improve the equivalent DCO resolution to reduce the tracking jitter. The resolution is reduced from 17.205ns to 33.604fs. Therefore, the design difficulty of high frequency multiplication factor ADPLL can be reduced.

When the input has large jitter, the proposed ADPLL utilizes the TDC and the SDM to compensate the phase error. The maximum phase error is controlled less than 1.991ns in WUXGA mode with the worst jitter model, when the ADPLL frequency multiplication factor is 2592.

For this ADPLL, we tape-out two test chips to verify our proposed method and the circuit techniques. The first test chip is to solve the DCO non-monotonic problem, and the second test chip is implemented to verify the overall ADPLL performance for video display applications. Two test chips are implemented by UMC 65nm 1P10M

standard performance (SP) COMS process. The chip area including I/O Pads of proposed ADPLL is 910×820 mm^2.

Because the reference frequency is very low, the simulation time is too long and not unacceptable. In the future, we hope to create the more accurately model in RTL level to simulate the performance in post-layout simulation to reduce the simulation time.

Although the interpolator circuit can solve the non-monotonic problem, but the linearity of the interpolator circuit is worst, and it will have more power consumption. Therefore, we hope to increase the linearity of the interpolator circuit in future.

A built-in self test (BIST) circuit for PLLs is becoming an important issue, so we can use the BIST circuit to detect whether our design is work correctly. Therefore, we hope to design the BIST circuit for on-chip jitter measurement to increase the circuit testability in future.

Reference

[1] Che-Fu Liang, Shin-Hua Chen, and Shen-Iuan Liu, "A Digital Calibration Technique for Charge Pumps in Phase-Locked Systems," in *IEEE Journal of Solid-State Circuits*, Vol. 43, pp. 390-398, Feb. 2008.

[2] Ashok Sqaminathan, Kevin J. Wang, and Ian Galton, "A Wide-Bandwidth 2.4GHz ISM Band Fractional-N PLL With Adaptive Phase Noise Cancellation," in *IEEE Journal of Solid-State Circuits*, Vol. 42, pp. 2639-2650, Dec. 2007.

[3] Kyoungho Woo, Yong Liu, Eunsoo Nam, and Donhee Ham, "Fast-Lock Hybrid PLL Combining Fractional-N and Integer-N Modes of Differing Bandwidths," in *IEEE Journal of Solid-State Circuits*, Vol. 43, pp. 379-389, Feb. 2008.

[4] Akinori Matsumoto, Shiro Sakiyama, Yusuke Tokunaga, Takashi Morie, and Shiro Dosho, "A Design Method and Developments of a Low-Power and High-Resolution Multiphase Generation System," in *IEEE Journal of Solid-State Circuits*, Vol. 43, pp. 831-843, Apr. 2008.

[5] John G. Maneatis, Jaeha Kim, Iain McClatchie, Jay Maxey, and Manjusha Shankaradas, "Self-Biased High-Bandwidth Low-Jitter 1-40-4096 Multiplier Clock Generator PLL," in *IEEE Journal of Solid-State Circuits*, Vol. 38, pp. 1795-1803, Nov. 2003.

[6] "Monitor Timing Specifications," in *VESA and Industry Standards and Guidelines for Computer Display Version 1.0*, Revision 10, Oct. 2004.

[7] Jean-Baptiste Bequeret, Yann Deval, Olivier Mazouffre, Anne Spataro, Pascal Fouillat, Eric Benoit, and Jean Mendoza, "Clock Generator using factorial DLL for Video Applications," *in Proceeding of IEEE Conference on Custom Integrated Circuits*, pp. 485-488, May. 2001.

[8] Analog Bits, "Video Capture PLL for HDTV and high-end flat panel monitor," in Datasheet. (http://www.analogbits.com/pdf/Video_Capture_PLL.pdf)

[9] Matrox Graphics, "UltraSharp Display Output Technology," in User Guides. (http://www.watch.impress.co.jp/pc/docs/2002/0515/us_displ.pdf)

[10] Guang-jun Xie, and Cheng Wang, "An All-Digital PLL for Video pixel Clock Regeneration Applications," in *Proceeding of WRI Congress on Computer Science and Information Engineering*, Vol. 3, pp. 392-396, 2009.

[11] Pao-Lung Chen, Ching-Che Chung, and Chen-Yi Lee, "A Clock Generation with Cascaded Dynamic Frequency Counting Loops for Wide Multiplication Range Applications," in *IEEE Journal of Solid-State Circuits*, Vol. 41, pp. 1275-1285, Jun. 2006.

[12] 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.

[13] Jose A. Tierno Alexander V. Rylyakov, and Daniel J. Friedman, "A Wide Power Supply Range, Wide Tuning Range, All Static CMOS All Digital PLL in 65nm SOI," in *IEEE Journal of Solid-State Circuits*, Vol. 43, pp. 42-51, Jan. 2008.

[14] Robert Bogdan Staszewski, Chih-Ming Hung, Dirk Leipold, and Poras T. Balsara, "A First Multigigahertz Digitally Controlled Oscillator for Wireless Application," in *IEEE Transactions on Microwave Theory and Techniques*, Vol. 51, pp. 2154-2164, Nov. 2003.

[15] Robert Bogdan Staszewski, Khurram Muhammad, Dirk Leipold, Chih-Ming Hung, Yo-Chuol Ho, John L. Wallberg, Chan Fernando, Ken Maggio, Roman Staszewski, Tom Jung, Jinseok Koh, Soji John, Irene Yuanying Deng, Vivek Sarda, Oscar Moreira-Tamayo, Valerian Mayega, Ran Katz, Ofer Friedman, Oren Eytan Eliezer, Elida de-Obaldia, and Poras T. Balsara, "All-Digital TX Frequency Synthesizer and Discrete-Time Receiver for Bluetooth Radio in 130-nm CMOS," in *IEEE Journal of Solid-State Circuits*, Vol. 39, pp. 2278-2290, Dec. 2004.

[16] 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," in *IEEE Journal of Solid-State Circuits*, Vol. 30, pp. 412-422, Apr. 1995.

[17] Abhijith Arakali, Srikanth Gondi, and Pavan Kumar Hanumolu, "Low Power Supply Regulation Techniques for Ring Oscillators in Phase-Locked Loops Using a Split-Tuned Architecture," in *IEEE Journal of Solid-State Circuit*, Vol. 44, pp. 2169–2181, Aug. 2009.

[18] Chao-Ching Hung, and Shen-Iuan Liu, "A Leakage-Compensated PLL in 65-nm CMOS Technology," in *IEEE Transaction on Circuits and System II: Express Briefs*, Vol. 56, pp. 525-529, Jul. 2009.

[19] Thomas Olsson, and Peter Nilsson, "A digitally controlled PLL for SoC applications," in *IEEE Journal of Solid-State Circuit*, Vol. 39, pp. 751-760, May 2004.

[20] Robert Bogdan Staszewski, Chih-Ming Hung, Dirk Leipold, and Poras T. Balsara, "A first multigigahertz digitally controlled oscillator for wireless applications," in *IEEE Transactions on Microwave Theory and Techniques*, Vol. 51, pp. 2154-2164, Nov. 2003.

[21] Pao-Lung Chen, Ching-Che Chung, and Chen-Yi Lee, "A portable digitally controlled oscillator using novel varactors," in *IEEE Transaction on Circuits and System II: Express Briefs*, Vol. 52, pp. 233-237, May 2005.

[22] Duo Sheng, Ching-Che Chung, and Chen-Yi Lee, "An Ultra-Low-Power and Portable Digitally Controlled Oscillator for SoC Applications," in *IEEE Transaction on Circuits and System II: Express Briefs*, Vol. 54, pp. 954-958, May 2007.

[23] Duo Sheng, Ching-Che Chung, and Chen-Yi Lee, "An all digital spread spectrum clock generator with programmable spread ratio for SoC applications," *in Proceeding of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 850-853, Nov. 2008.

[24] Hsuan-Jung Hsu, Chun-Chieh Tu, and Shi-Yu Huang, "A high-resolution all-digital phase-locked loop with its application to built-in speed grading for memory," *in Proceeding of IEEE Symposium on VLSI Design Automation and Test (VLSI-DAT)*, pp. 267-270, Apr. 2008.

[25] Byoung-Mo Moon, Young-June Park, and Deog-Kyoon Jeong, "Monotonic Wide-Range Digitally Controlled Oscillator Compensated for Supply Voltage Variation," in *IEEE Transaction on Circuits and System II: Express Briefs*, Vol. 55, pp. 1036-1040, Oct. 2008.

[26] Duo Sheng, Ching-Che Chung, and Chen-Yi Lee, "A Low-Power and Portable Spread Spectrum Clock Generator for SoC Applications," accepted by *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2010.

[27] Kwang-Hee Choi, Jung-Bum Shin, Jae-Yoon Sim, and Hong-June Park, "An Interpolating Digitally Controlled Oscillator for a Wide-Range All-Digital PLL," in *IEEE Transaction on Circuits and System II: Express Briefs*, Vol. 56, pp. 2055-2063, Sep. 2009.

[28] Chen-Yi Lee, and Ching-Che Chung, "Digital Loop Filter for All-Digital Phase-Locked Loop Design," US patent 7,696,832 B1, Apr.13, 2010.

[29] Robert Bogdan Staszewski, Dirk Leipold, Chih-Ming Hung, and Poras T. Balsara, "A first digitally-controlled oscillator in a deep-submicron CMOS process for multi-GHz wireless applications," in *Proceeding of IEEE Radio Frequency Integrated Circuits (RFIC) Symposium*, pp. 81–84, June 2003.

[30] Robert Bogdan Staszewski, Sudheer Vemulapalli, Prasant Vallur, John Wallberg, and Poras T. Balsara, "1.3V 20ps time-to-digital converters for frequency synthesis in 90-nm CMOS," in *IEEE Transaction on Circuits and System II: Express Briefs*, Vol. 53, pp. 220-224, Mar. 2006.

[31] Liming Xiu, Wen Li, Jason Meiners, Rajitha Padakanti, "A novel all digital PLL with software adaptive filter," in *IEEE Journal of Solid-State Circuits*, Vol. 39, pp. 476-483, Mar. 2004.

[32] Hsuan-Jung Hsu, and Shi-Yu Huang, "A Low-Jitter ADPLL via a Suppressive Digital Filter and an Interpolation-Based Locking Scheme," in *IEEE*

Transactions on Very Large Scale Integration (VLSI) Systems, 2010.