Sub-Picosecond Jitter Clock Generation for Time Interleaved Analog to Digital Converter

by

Jianping Gong

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Electrical and Computer Engineering

by

____

July 2019

APPROVED:

____

Professor John McNeill, Advisor, Dean of Engineering

____

Professor Ulkuhan Guler

____

Dr. Michael Coln, Fellow, Analog Devices Inc

Abstract

Nowadays, Multi-GHz analog-to-digital converters (ADCs) are becoming more and more popular in radar systems, software-defined radio (SDR) and wideband communications, because they can realize much higher operation speed through using many interleaved sub-ADCs to relax ADC sampling rates. Although the time interleaved ADC has some issues such as gain mismatch, offset mismatch and timing skew between each ADC channel, these deterministic errors can be solved by previous works such as digital calibration technique. However, time-interleaved ADCs require a precise sample clock to achieve an acceptable effective-number-of-bits (ENOB) which can be degraded by jitter in the sample clock. The clock generation circuits presented in this work achieves sub-picosecond jitter performance in 180nm CMOS which is suitable for time-interleaved ADC.

Two different test chips were fabricated in 180nm CMOS to investigate the low jitter design technique. The low jitter delay line in two chips were designed in two different ways, but both of them utilized the low jitter design technique. In first test chip, the measured RMS jitter is 0.1061ps for each delay stage. The second chip uses the proposed low jitter Delay-Locked Loop can work from 80MHz to 120MHz, which means it can provide the time interleaved ADC with 2.4GHz to 3.6GHz low jitter sample clock, the measured delay stage jitter performance in second test chip is 0.1085ps.

Acknowledgements

I would like to express my deepest gratitude to my Ph.D. advisor, Prof. John McNeill, who has mentored me during my whole Ph.D. period. He is the most inspiring and motivating advisor I have ever had. He has not only taught me the knowledge, but also the approach to analyze and solve the difficult problems, which will benefit me in the future. I am grateful that I had the opportunity to study under his guidance.

At Worcester Polytechnic Institute, I also want to thank Ph.D. student Sulin Li who has helped me with the FPGA coding. His expertise on digital programming accelerated my chip evaluation process. I want to thank Bob Brown, who has maintained Cadence and other EDA tools.

I also want to thank my family who has always supported me when I met any difficulties. Their love, patience and support has always encouraged me and gave me the confidence to finish my study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Scaling of CMOS Integrated Circuit (IC) technology into nanometer scale gate dimensions has enabled dramatic improvement in cost, performance, and levels of system integration. For analog functions, scaling can be a mixed blessing: cost is improved due to reduced die area but analog performance is often degraded [1]. Despite the performance challenges, the cost advantages of integrating systems in nanometer CMOS have motivated a wide variety of work in analog and mixed signal design.

Recently, advanced CMOS direct radio frequency (RF) sampling analog-to-digital converters (ADCs) [2] have been widely used in radar systems, software-defined radio (SDR), wideband communications, etc. Unlike the traditional intermediate frequency (IF) sampling, which downconverts the RF signal directly to baseband to relax ADC sampling rates, RF sampling ADC (shown in Fig 1.1) can convert the incoming high frequency analog signal directly into the digital domain. By using this method, the IF stage is removed to reduce power consumption and printed cir-

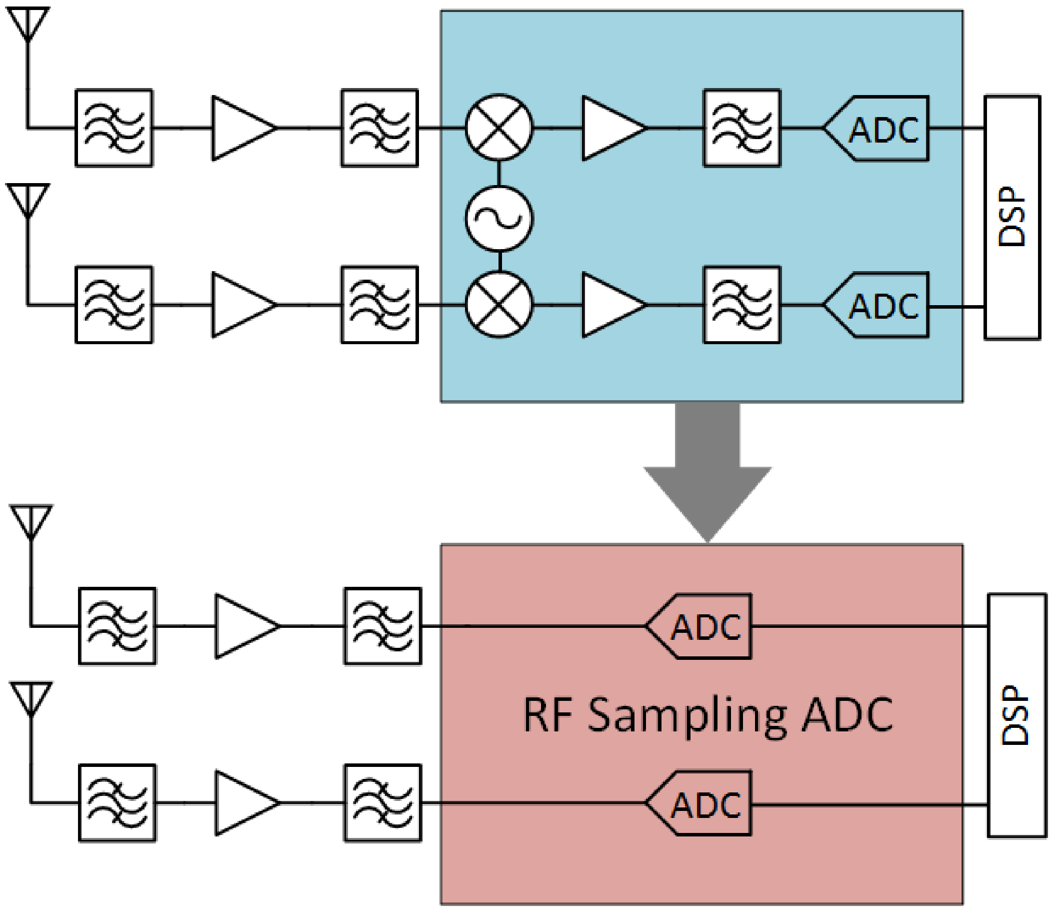


Figure 1.1: Traditional IF Sampling vs RF Sampling

cuit board (PCB) area, this is one of the main example applications of RF ADC. Because RF applications require high speed and moderate resolution data conversion, time-interleaved ADCs [3–18] are an attractive solution for RF sampling due to better power efficiency and low frequency sample clock, and can achieve a high sampling rate through multiple sub-ADCs operating at a lower sampling rate [19].

However, time-interleaved ADCs require a precise sample clock to achieve an acceptable effective-number-of-bits (ENOB) which can be degraded by jitter in the sample clock. generating a low-cost, low-jitter sample clock is a challenge for circuit designers. Although several ADCs [20–22] published to date can achieve acceptable performance with calibration of systematic clock phase mismatch [23], these work do not address the challenge to generate a precise sample clock for interleaved ADCs.

ADC performance can be characterized by ENOB, which is determined by the observed signal-to-noise ratio (SNR) in the frequency domain (FFT) ADC output

$$ENOB = \frac{SNR - 1.76dB}{6.02dB} \tag{1.1}$$

with SNR in dB. The SNR impact of a sample clock jitter σ_J with a full-scale sinusoidal input at frequency f_{SIG} is given by [24]

$$SNR = 20\log\left(\frac{1}{2\pi f_{SIG}\sigma_J}\right) \tag{1.2}$$

Equations (1.1) and (1.2) determine the required sampling clock uniformity necessary to achieve a desired ENOB. Note that ENOB can be degraded by either systematic clock errors (leading to spurs in the FFT) or random clock jitter (leading to increased noise floor in the FFT).

Phase-Locked-Loops (PLLs) and Delay-Locked-Loops (DLLs) have been widely used to generate and recover the clock in communication integrated circuits. Although PLLs are preferred for applications such as frequency synthesis, the DLL is an attractive option for clock phase generation because of

- its better jitter performance since the DLL avoids the jitter accumulation [28] characteristic of the PLL, and
- the unconditional stability [29] of the DLL feedback loop which has first order loop characteristics.

The main disadvantage of DLL for generating the ADCs sample clock is the nonuniformity of the sample clocks due to systematic mismatch in the DLL phase delays. The systematic mismatch can lead to spurs in the FFT, degrading SNR. Deterministic errors can be removed through calibration techniques such as "split ADC" architecture [5] which can be used to provide all-digital background calibration of aperture delay mismatch errors, thus making it possible to apply the low jitter DLL to generate the interleaved ADC sampling clocks.

In order to design a Delay-Locked Loop which is suitable for time interleaved ADC, two test chips were designed and fabricated. Because time interleaved ADC requires low jitter sample clock, the first chip was designed to investigate the low jitter theory, a low jitter delay line was fabricated and evaluated to prove the jitter theory. Based on the measured results of the first chip, the proposed low jitter Delay-Locked Loop was designed on the second chip to generate the sample clock of time interleaved ADC.

1.2 Contribution Overview

This work has investigated the jitter theory and proved a low-cost low-complexity way to generate sub-picosecond jitter clocks for time interleaved ADC. The RF applications which require high speed and moderate resolution data conversion, the time interleaved ADC is an attractive solution for the RF sampling due to its power efficiency and low frequency sample clock. This work provided a new way to generate the low jitter low frequency sample clock for the time interleaved ADCs.

In paper [24], the idea of investigating the low jitter theory has been presented, and the preliminary design has been proposed, some simulations have also been proposed. In this paper, the importance of the low jitter clock for time interleaved ADC has been prove from some simulations.

In order to investigate the low jitter theory, a low jitter delay line has been designed and fabricated. Paper [25] presented a design technique using a digitally-controlled delay line for the generation of sub-picosecond jitter ADC sampling clock phases from a low-cost low-frequency clock source. The measured results demonstrated the propagation delay and low jitter theory in the proposed delay line, this chip was an exploration for the low jitter clock design in the future.

Based on the results from work in [25], a low jitter delay locked loop was designed and simulated in paper [26]. In this paper, the new digital Delay-Locked Loop was proposed including the Bang-Bang phase detection, probability computing theory and new digital system architecture. This new digital architecture focused on the low jitter clock generation design.

According to the work results in [26], a new low jitter Delay-Locked Loop was designed and fabricated in 180nm CMOS, this work has some improvements based on the work in [26], including the ultra low jitter bang-bang phase detector, more digital modes in digital system and some changes in the delay line. The publication [27] of this work is in preparation. In order to investigate the jitter theory in Delay-Locked Loop, there are auto/manual modes in the digital system, which is suitable for the jitter measurement. Moreover, there are varactors at the output of each delay stage which can be used to prove the jitter theory. Dummy stages also put in the delay line in order to have matched output load. The chip measurement results proves that the proposed Delay-Locked Loop is an attractive solution the low-cost low-complexity ultra low jitter clock generation.

Besides the work mentioned above, the author have also finished other projects and have some publications listed below:

- J. Gong and J. A. McNeill, "Sub-Picosecond-Jitter Clock Generation for Interleaved ADC," *2015 IEEE Conference on PhD Research in Microelectronics (PRIME2015)*, Glasgow, Scotland, June 2015.

- J. Gong, S. Li, J. A. McNeill, "Sub-Picosecond-Jitter Clock Generation for Interleaved ADC with Delay-Locked-Loop in 28nm CMOS," *IEEE 2016 Int'l Symposium on Circuits and Systems(ISCAS 2016)*, Montreal, Quebec, Canada, May 2016.

- J.McNeill, J.Gong, R.Majidi.,"Design of a Sub-Picosecond-Jitter Delay-Lock-Loop for Interleaved ADC Sample Clock Synthesis," *IEEE International Midwest Symposium on Circuits and Systems*, pp. 1854-1865, Aug, 2012.

- S. Li, J. Gong, J. McNeill, "VCO-Based ADC With Digital Background Calibration in 65nm CMOS," *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)*, Montreal, QC, Canada

- J. McNeill, R. Majidi, J. Gong, C. Liu, "Lookup-table-based background linearization for VCO-based ADCs," *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*, San Jose, CA, USA

- J. Gong, Y. Kim, F. Lombardi, J. Han "Hardening a memory cell for low power operation by gate leakage reduction," *2012 IEEE International Symposium* *on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Austin, TX, USA

- J. McNeill, R. Majidi, J. Gong "Split ADC Background Linearization of VCO-Based ADCs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 49-58, Oct 2014.

- J. Gong, S. Li, J. McNeill "Sub-Picosecond Jitter Clock Generation for Interleaved ADC," *Applied Sciences*, In preparation.

1.3 Dissertation Organization

The following chapters of this dissertation are organized as follows: Chapter 2 will introduce some background knowledge which is related to this dissertation; different kinds of jitter and the relationship between jitter and propagation delay will be defined. Chapter 3 will illustrate the design details of the low jitter delay-line chip design. Chapter 4 will discuss the measurement results of the low jitter delay-line chip. According to the results from Chapter 4, the low jitter Delay-Locked Loop is designed in Chapter 5, and Chapter 5 will show the details of the design technique, probability theory behind the design, and digital system. Chapter 6 will discuss the measurement results of the low jitter Delay-Locked Loop. Chapter 7 summarizes and concludes this dissertation and proposes directions for future work.

Chapter 2

Background

2.1 Clock Jitter

Jitter is a time variant noise component that impacts the phase of a signal. The clock jitters are considered as small random errors [33]. There are two broad categories of jitter: random jitter (RJ) and deterministic jitter (DJ) [37]. Fig 2.1 shows the jitter component classification tree [30]. It can be seen that the random jitter is unbounded and deterministic jitter is bounded. The deterministic jitter can be further categorized into data-dependent jitter (DDj), Periodic jitter (Pj), and bounded uncorrelated to data jitter (BUJ). Jitter is the definition which is used to quantify the signal error in time domain; in frequency domain, phase noise is used to quantify the random fluctuations in the phase of a waveform. Fig 2.2 shows the jitter in time domain and Fig 2.3 shows the phase noise in frequency domain. In the figure, A(t) is the random amplitude change and $\varphi(t)$ is the random phase change. In Fig 2.2, it can be seen that the clock cycle of a perfect clock is always the same. However, in the real world, the perfect clock does not exist and there is always jitter in the real world clock. Because of the existence of the jitter, every clock cycle is different

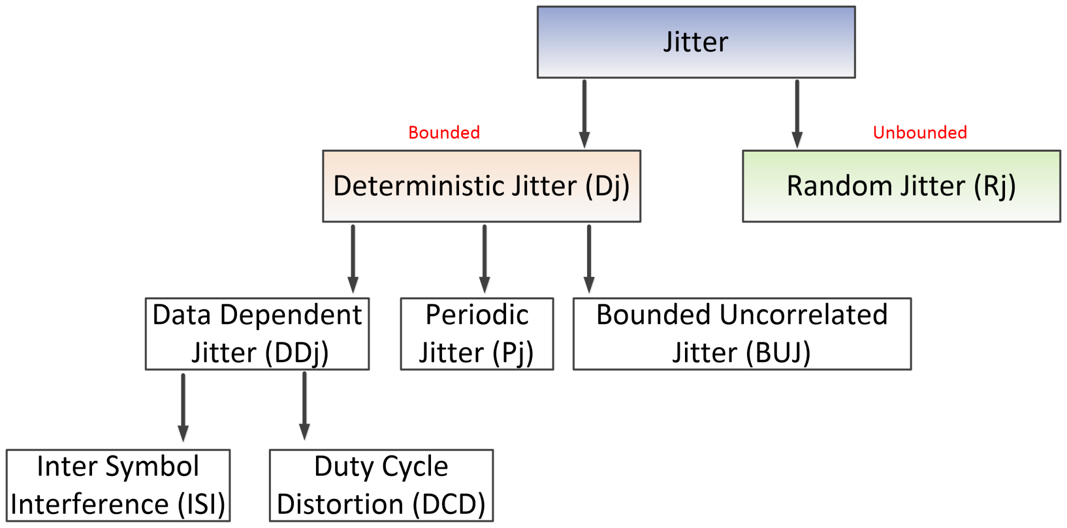

Jitter
Bounded
Unbounded
Deterministic Jitter (Dj)
Random Jitter (Rj)
Data Dependent Jitter (DDj)
Periodic Jitter (Pj)
Bounded Uncorrelated Jitter (BUJ)
Inter Symbol Interference (ISI)
Duty Cycle Distortion (DCD)

Figure 2.1: Jitter Component Classification Tree

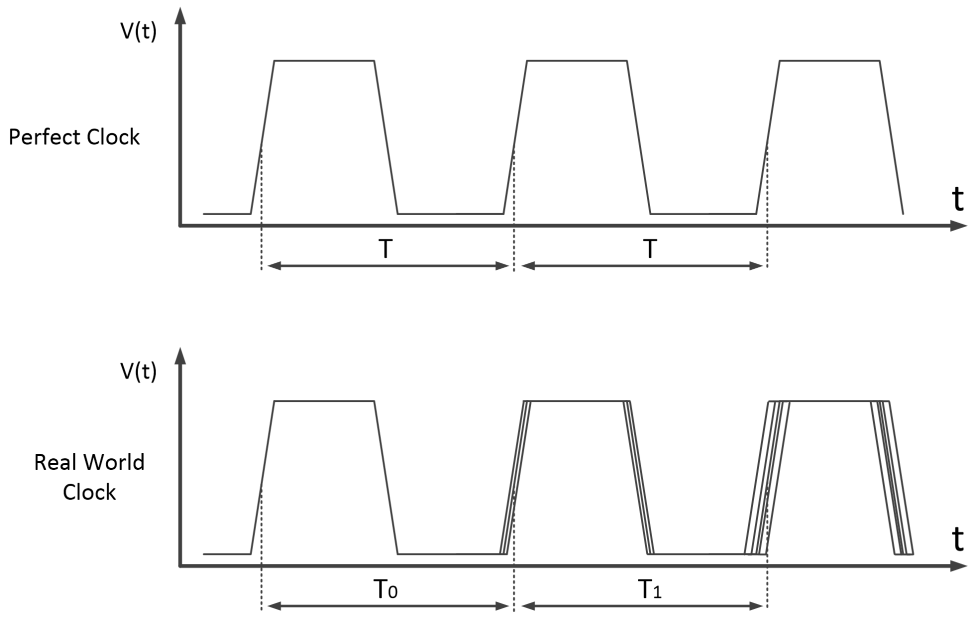


Figure 2.2: Jitter in Time Domain

from others; In Fig 2.2, T_0 and T_1 are different due to jitter. In Fig 2.3, it can be seen that the power of a perfect sinusoidal signal will be at one specific frequency; however, in Fig 2.3 due to the jitter, the frequency spectrum plot is a real sinusoidal clock with carrier frequency F_C plus sideband offset noise frequencies. It means that sometimes the clock will have a slightly larger frequency and sometimes the clock

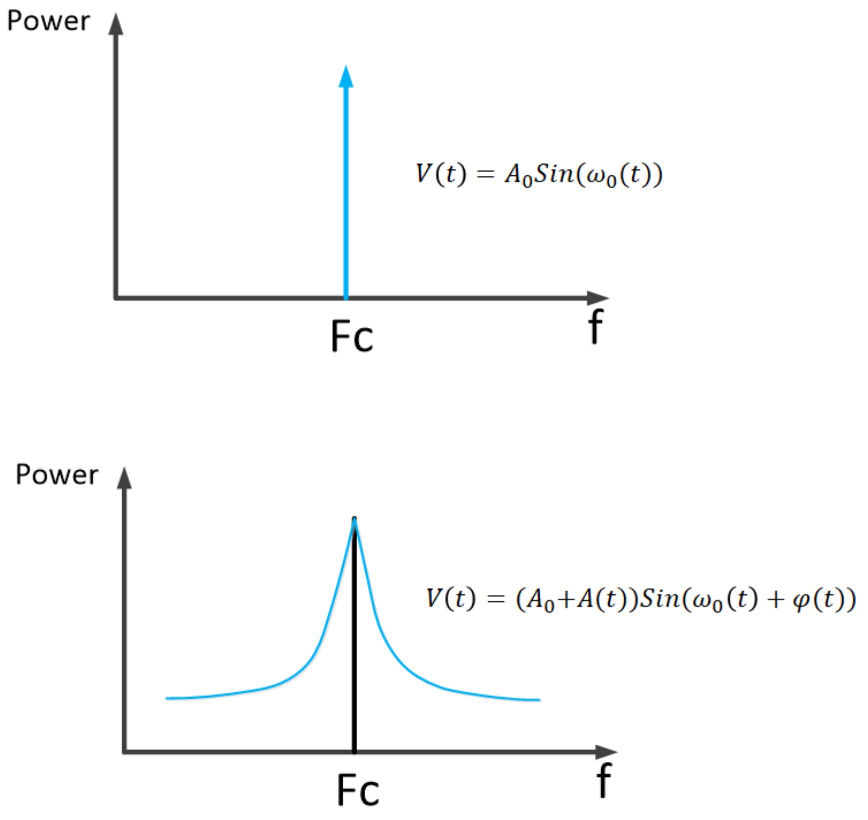

Power
$V(t) = A_0 Sin(\omega_0(t))$
Fc
f
Power
$V(t) = (A_0 + A(t))Sin(\omega_0(t) + \varphi(t))$
Fc
f

Figure 2.3: Jitter in Frequency Domain - Phase Noise

will have a slightly lower frequency. The small changes in frequency appear as phase shifts in the clock waveform, hence the name phase noise [86].

2.1.1 Random Jitter

Random jitter follows the Gaussian Distribution [34, 44, 45] and its instantaneous noise value is mathematically unbounded. The random jitter is primarily caused by thermal noise [31] and shot noise [32]. From Fig 2.4, it can be seen that the two tails of the Gaussian Distribution curve is very close to zero, but it will never fully reach zero, that is why the random jitter is unbounded, however the probability of a very large amount of random jitter on a given clock transition is very small, for example, from the Gaussian distribution, the probability of the jitter which is

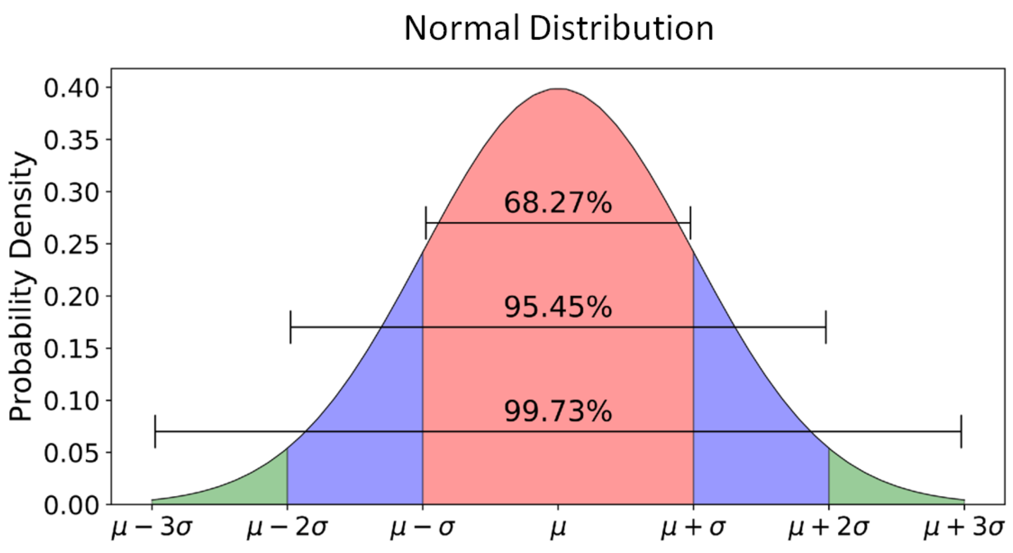


Figure 2.4: Normal Distribution of Random Jitter: probability of a large amount of random jitter is small, but not zero.

seven times of the standard deviation is one in a trillion. Moreover, the random jitter due to white noise on a given clock transition is not related to the previous clock transition, it means that in an electronic circuit, for example, the delay line of a Delay-Lock Loop, the random jitter due to white noise in each delay stage is uncorrelated to the signal from previous delay stage.

2.1.2 Deterministic Jitter

Deterministic jitter is not random and it is not intrinsic to every system. Deterministic jitter can be calculated from a complete understanding of the circuit and its environment, so it is predictable and reproducible, and it is usually measured in terms of a peak-to-peak value. Unlike the random jitter which is Gaussian in nature and unbounded, the deterministic jitter is bounded with minimum and maximum value. Deterministic jitter can be further categorized into data-dependent jitter (DDj), Periodic jitter (Pj), and bounded uncorrelated to data jitter (BUJ) (e.g.,

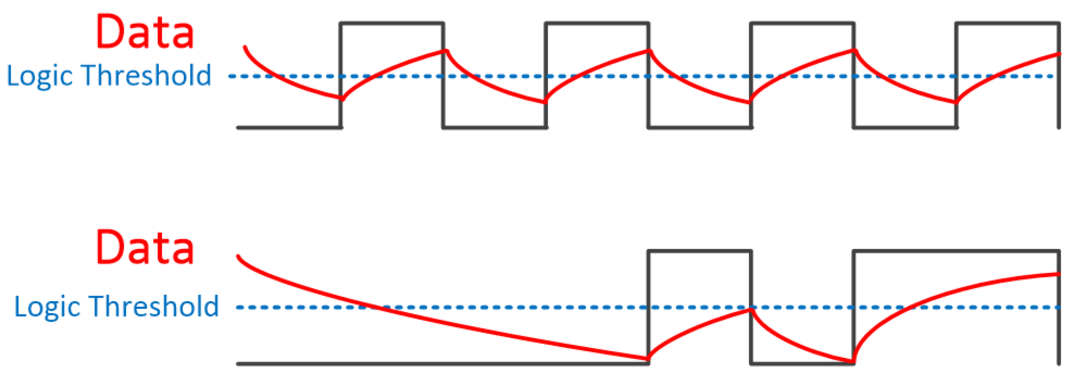


Figure 2.5: Inter Symbol Interference Example

crosstalk jitter or sinusoidal jitter) [42, 43].

Data dependent jitter is the signal variation which is caused by the changing of transmitted data pattern. For example, when the jitter of a rising edge of a clock which follows a sequence of bits, e.g., 11110000, differs from the jitter of rising edge of a clock which follows a different sequence of bits, e.g., 11001100, then this jitter is called data-dependent jitter. Data Dependent Jitter is caused primarily by phase errors in serialization clocks, channel filtering crosstalk, resistance and frequency response of the transmission path, and asymmetries in the rising end falling edges of the transmitted signal [38].

The data dependent jitter can be further categorized into Inter Symbol Interference (ISI) and Duty Cycle Distortion (DCD). Inter Symbol Interference (ISI) is the primary cause of data dependent jitter (DDj) and the situation is complicated by the correlation of Inter Symbol Interference and Duty Cycle Distortion [38]. The Inter Symbol Interference is usually due to the bandwidth limitation problem in either the transmitter or physical media; or caused by improper impedance termination [39]. Fig 2.5 shows an example of Inter Symbol Interference (ISI). In the Figure, if the data is 01010101, then after a long RC signal path, the data can also be 01010101, because the data can just cross the logic threshold voltage (half of

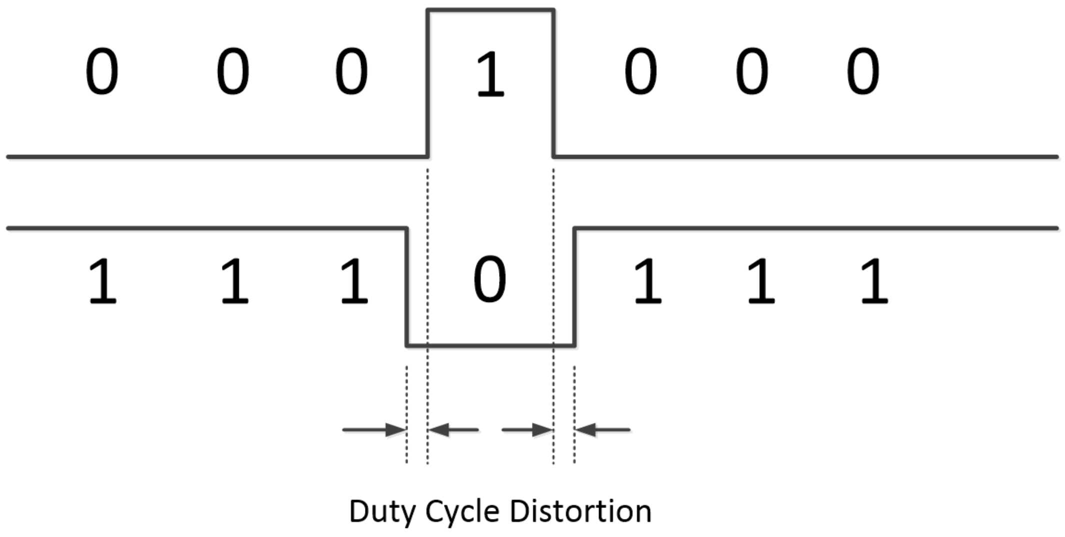


Figure 2.6: Duty Cycle Distortion Example

the high voltage), the final value does not need to reach the high voltage and low voltage, when the value can cross the logic threshold voltage, the logic value can be determined. However, if the data is 00001011, when the data pass a long RC signal path, the data logic value may be read incorrectly. Because there is a long string of identical bits 0, after a long RC data pass, the data can reach the low voltage rail value, if there is a logic 1 following the long string of 1s, there is no enough time for the signal to cross the logic threshold value, so the data 00001011 could be read as 10000001 when the data pass the long RC data path. The Duty Cycle Distortion (DCD) is is a measure of the asymmetry in the duty cycle of the transmitter [38]. Fig 2.6 shows the example of duty cycle distortion. In the Figure, the width of logic 1 in string 0001000 is different from the width of logic 0 in string 1110111. The duty cycle distortion will cause the clock rising edge and falling edge displacement, which may lead to digital circuit timing error.

Periodic jitter is caused by clocks or other periodic sources that can modulate the transmitted edges, usually it is due to crosstalk between neighboring lines resulting in bounded uncorrelated jitter in the data eye [35], so the periodic jitter is very easy

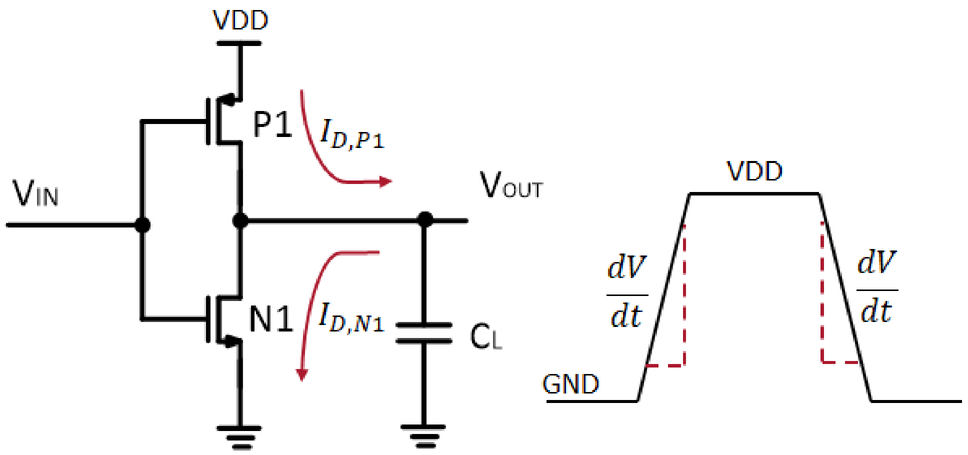


Figure 2.7: Schematic of Inverter

to be measured accurately and appears in the jitter - frequency spectrum as distinct peaks.

2.2 Propagation Delay

Fig 2.7 shows the schematic of a typical inverter, C_L is the load capacitance of the inverter, the C_L is the combination of the parasitic capacitance of PMOS and NMOS transistor, the capacitance between inverter output metal wire and ground, and the load capacitance which is from the next following circuit of the inverter.

In Fig 2.7, rising edge dV/dt is determined by available charging current $I_{D,P1}$, the available drain current from PMOS device P1. Similarly, falling edge dV/dt is determined by available discharging current $I_{D,N1}$ from NMOS device N1. Fig 2.8 shows the definition of propagation delay. The Low-to-High delay T_{PLH} and High-to-Low delay T_{PHL} are defined as the time difference between the inverter input and output signal when the signal voltage is 50% of the power supply. The propagation delay of an inverter is the average value of the Low-to-High delay and High-to-Low delay. Using the simplified propagation delay expressions in [87], we have

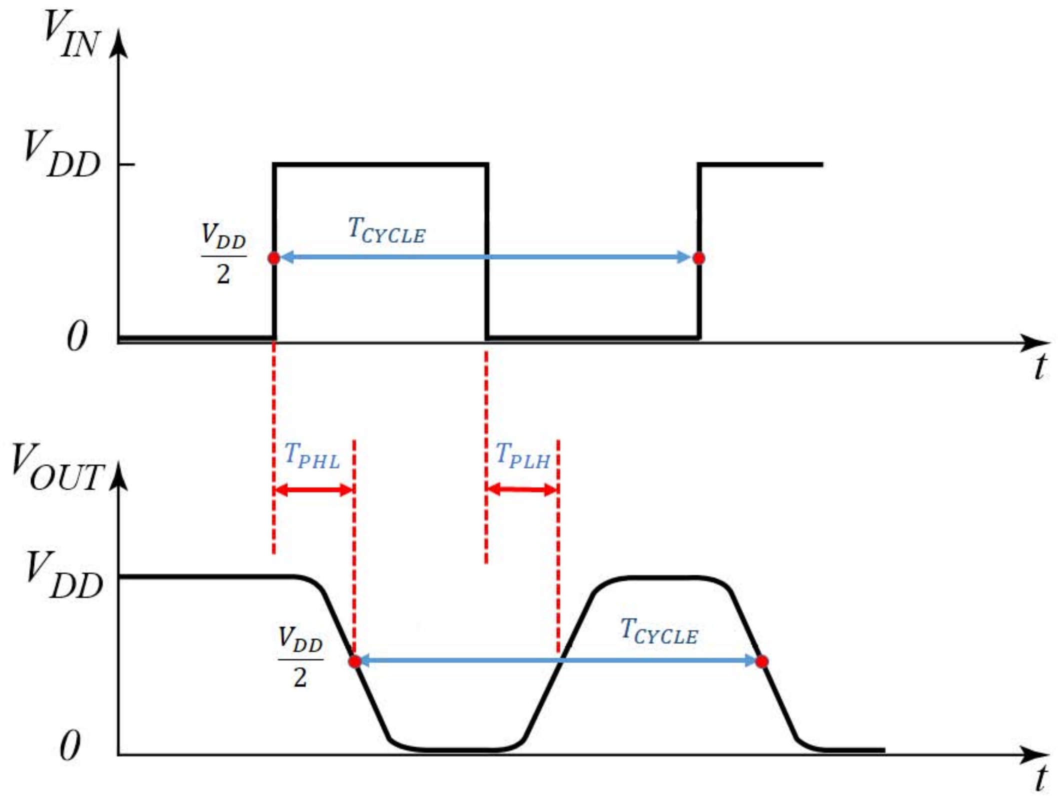


Figure 2.8: Propagation Delay of Inverter

$$T_{PLH} \approx \frac{C_L V_{DD}}{2I_{D,P1}} \tag{2.1}$$

and

$$T_{PHL} \approx \frac{C_L V_{DD}}{2I_{D,N1}} \tag{2.2}$$

to simplify the equation, we have used $I_{peak} = I_{D,N1} = I_{D,P1}$ and I_{peak} is the peak current of the $I_{D,N1}$ and $I_{D,P1}$. So we can have

$$t_{PD} = \frac{t_{PHL} + t_{PLH}}{2} = \frac{C_L V_{DD}}{2I_{peak}} \tag{2.3}$$

where C_L is load capacitance, $C_L{=}I_D/(dV/dt)$. In the analysis above, the load capacitance C_L is assumed identical in low-to-high and high-to-low transitions.

2.3 Figure of Merit κ

In [87] the jitter contribution of a noise source was described by the Figure of Merit κ. κ is the proportionality constant observed between the jitter standard deviation σ and the square root of delay time ΔT:

$$\sigma_{\Delta T} = \kappa\sqrt{\Delta T} \tag{2.4}$$

The jitter in equation (2.4) is the sum of individual gate jitter errors. If the jitter of each gate is uncorrelated then equation (2.4) can also be used for the single gate delay T_d and the corresponding RMS jitter σ_t:

$$\sigma_t = \kappa\sqrt{T_d} \tag{2.5}$$

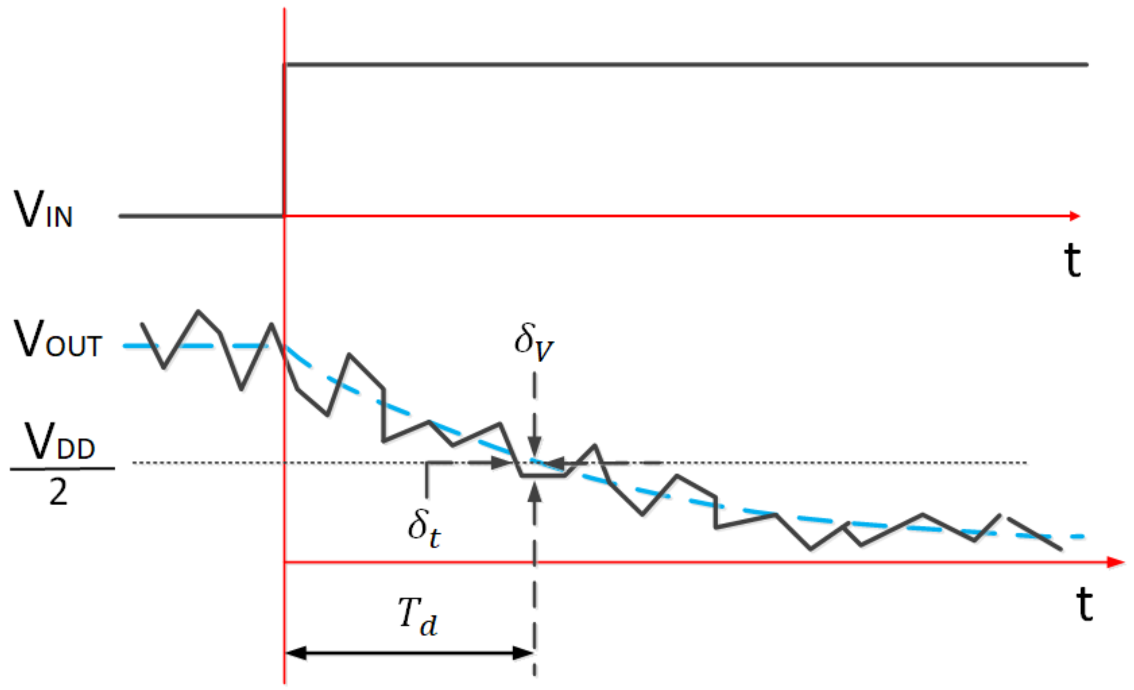


Figure 2.9: noise waveform of inverter

2.4 The impact of Peak Current on Clock Jitter

In a traditional ring oscillator, each delay stage is an inverter. In an inverter, any source of noise can affect its propagation delay and contribute the jitter to the inverter output. In the analysis of inverter noise, the noise comes from two sources: (1) the total of any resistive loading which is represented by R_L. (2) the current noise of switching element which is represented by $i_{n(SW)}$. Fig 2.9 shows the noise waveform of the inverter.

2.4.1 Load Element Noise

Fig 2.10 shows the load resistance model of a single ended inverter. To analyze the jitter of the single ended inverter, we simplify that:

- the gate delay T_d occurs at the logic threshold $V_{DD}/2$

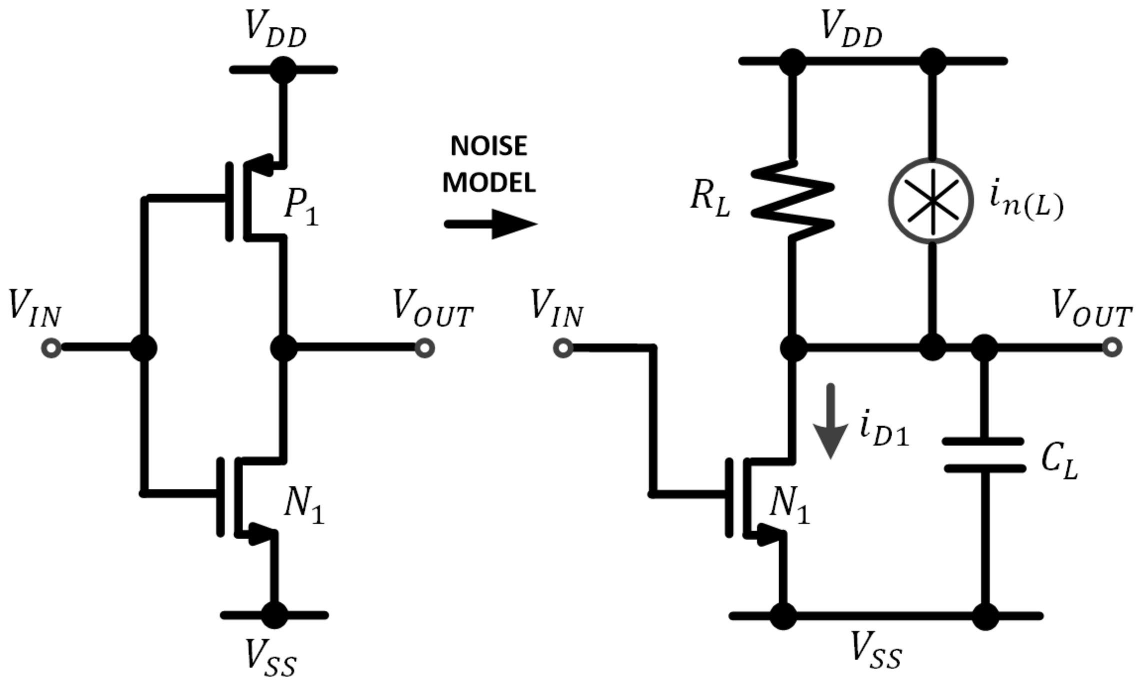


Figure 2.10: Load resistance noise model of single ended inverter

- the NMOS device NM1 turns on in the saturation region and the drain current can reach the maximum value i_{D1}=I_{PK} immediately.
- the total capacitive load at output is C_L, the total resistive load is R_L

So we can get:

$$V_{OUT}\left(t\right) = V_{DD}/2 = V_{DD} - I_{PK}R_L\left[1 - e^{T_d/R_LC_L}\right] \tag{2.6}$$

From the equation above:

$$T_d \approx \frac{C_LV_{DD}}{2I_{PK}} \tag{2.7}$$

The slope of V_{OUT} can be found by taking the time derivative of equation (2.6):

$$S\left(t\right) = \frac{dv}{dt} = -\frac{I_{PK}}{C_L}e^{-t/R_LC_L} \tag{2.8}$$

Substituting equation (2.8) into equation (2.7):

$$S\left(T_d\right) = -\frac{I_{PK}}{C_L}e^{-T_d/R_L C_L} = -\frac{I_{PK}}{C_L}e^{-V_{DD}/2I_{PK}R_L} \approx -\frac{I_{PK}}{C_L} \tag{2.9}$$

The output voltage noise from resistive load R_L is:

$$\sigma_v = \sqrt{kT/C_L} \tag{2.10}$$

Then we can get the jitter σ_t:

$$\sigma_t = \frac{\sigma_v}{S(T_d)} = \sqrt{\frac{kTC_L}{I_{PK}^2}} \tag{2.11}$$

From equation (2.5) and equation (2.11):

$$\kappa = \sqrt{2}\sqrt{\frac{kT}{I_{PK}V_{DD}}} \tag{2.12}$$

2.4.2 Switching Element Noise

Fig 2.11 shows the switching element noise model of a single ended inverter. The transistor N1 is modeled as an ideal noise free current source of value I_{PK} in parallel with a noise source $i_{n(SW)}$. From [87], the current noise source can be modeled as a series of pulses of duration dt. The RMS amplitude of the current pulses is given by

$$\sigma_{in} = \frac{i_{n(SW)}}{\sqrt{2dt}} \tag{2.13}$$

so the standard deviation of the corresponding amount of charge is

$$\sigma_q = \sigma_{in}dt = \frac{i_{n(SW)}}{\sqrt{2}}\sqrt{dt} \tag{2.14}$$

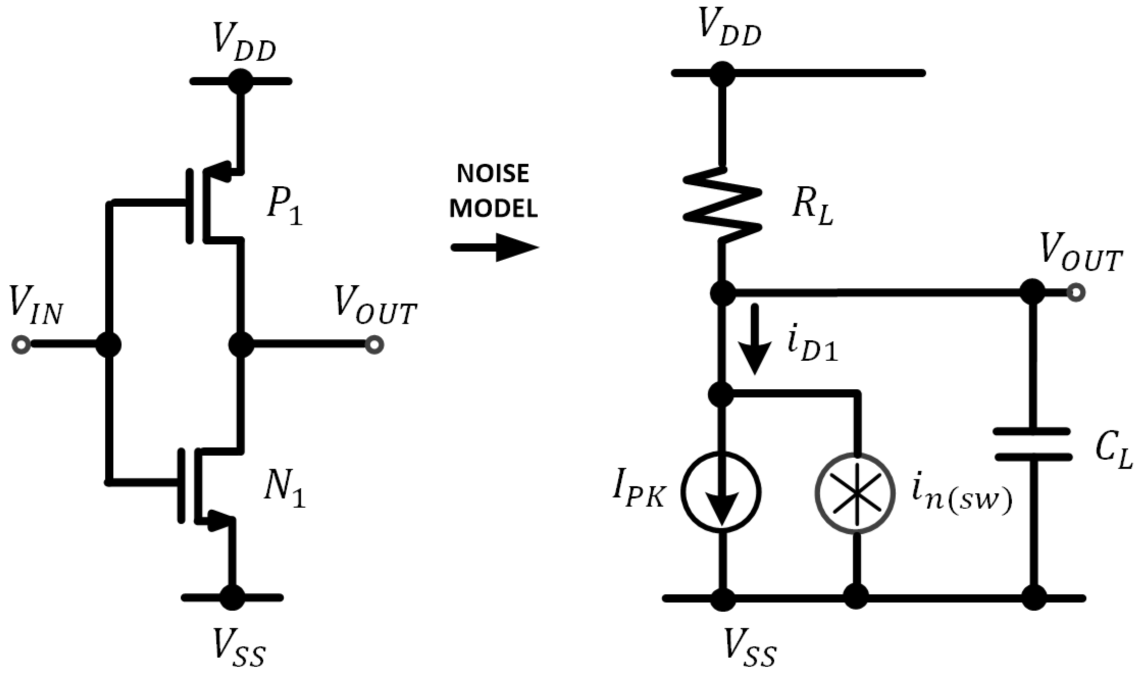


Figure 2.11: Switching element noise model of single ended inverter

To simplify, we assume that all charge is stored on the load capacitor C_L without loss from discharge through resistive load R_L, then the variance of the total charge is the sum of the individual variances. The special interest is the point at which the output voltage reaches the midpoint, this point is the switching threshold which is used to define the propagation delay, so the total charge integrated when the output voltage reaches the switching threshold $V_{DD}/2$ at time t = T_d in Fig 2.9 is given by

$$\sigma^2_{q(TOTAL\cdot)} = \int_0^{T_d} \frac{i^2_{n(SW)}}{2} dt = \frac{i^2_{n(SW)}}{2} Td \tag{2.15}$$

From the equation 2.7

$$T_d \approx \frac{C_L V_{DD}}{2 I_{PK}} \tag{2.16}$$

Substituting equation 2.16 into equation 2.15

$$\sigma^2_{q(TOTAL)} = \frac{i^2_{n(SW)}}{2}\frac{C_L V_{DD}}{2I_{PK}} \tag{2.17}$$

The RMS standard deviation of charge is

$$\sigma_{q(TOTAL)} = \frac{i_{n(SW)}}{2}\sqrt{\frac{C_L V_{DD}}{I_{PK}}} \tag{2.18}$$

The special interest is the point at which the V_{OUT} is crossing the threshold at time T_d in Fig 2.9, the voltage standard deviation at this point is

$$\sigma_{v(Td)} = \frac{\sigma_{q(TOTAL)}}{C_L} = \frac{i_{n(SW)}}{2}\sqrt{\frac{V_{DD}}{I_{PK}C_L}} \tag{2.19}$$

So the jitter σ_t is given by dividing the voltage standard deviation by slope S

$$\sigma_t = \frac{\sigma_{v(Td)}}{S} = \frac{i_{n(SW)}}{2}\sqrt{\frac{V_{DD}}{I_{PK}\cdot C_L}}\frac{C_L}{I_{PK}} = \frac{i_{n(SW)}}{2}\sqrt{\frac{V_{DD}C_L}{I^3_{PK}}} \tag{2.20}$$

So κ is given by

$$\kappa = \frac{\sigma_t}{\sqrt{Td}} = \frac{i_{n(SW)}}{2}\sqrt{\frac{V_{DD}C_L}{I^3_{PK}}}\sqrt{\frac{2I_{PK}}{V_{DD}C_L}} = \frac{1}{\sqrt{2}}\frac{i_{n(SW)}}{I_{PK}} \tag{2.21}$$

According to [46],

$$i_{n(s)} = \sqrt{4kT\gamma g_{m0}} \tag{2.22}$$

Where coefficient γ is 2/3 for long channel MOS devices and may need to be replaced by a larger value for submicron MOSFETs [47].

Substituting equation (2.23) into equation (2.21)

$$\kappa = \sqrt{\frac{2kT\gamma g_{m0}}{I^2_{PK}}} \tag{2.23}$$

From equation (2.23), it can be seen that jitter will decrease when I_{PK} increase for a given propagation delay. It can also be seen that γ will be different for long channel and short channel device.

- For long channel case, $\gamma = 2/3$ and

$$g_{m0} = \frac{2I_{PK}}{V_{DD} - V_t} \tag{2.24}$$

where V_t is the MOS threshold voltage and we assume that the gate is driven with a full swing input V_{GS}=V_{DD}, substitute equation (2.24) into equation (2.23), we have

$$\kappa = \sqrt{\frac{8}{3}}\sqrt{\frac{kT}{I_{PK}\left(V_{DD} - V_t\right)}} \tag{2.25}$$

- For short channel case, $\gamma > 2/3$. Moreover, due to velocity saturation, the expression for g_m is also different. For current, according to [48]

$$I_{PK} = v_{sat}C_{ox}\left(V_{GS} - V_t\right) \tag{2.26}$$

where v_{sat} is the saturated carrier velocity, C_{ox} is the gate oxide capacitance per unit area, and W is the effect channel width of the MOSFET. Given this expression for I_{PK}, g_{m0} is given by

$$g_{m0} = v_{sat}C_{ox}W \tag{2.27}$$

we assume that the gate is driven with a full swing input V_{GS}=V_{DD}, substituting equation (2.26) and equation (2.27) into equation (2.23) gives

$$\kappa = \sqrt{2\gamma}\sqrt{\frac{kT}{I_{PK}\left(V_{DD} - V_t\right)}} \tag{2.28}$$

2.5 Clock Generation

One of the critical tasks in designing a Time-Interleaved ADC is getting the low jitter time-interleaved clocks for sub-ADCs, because the clock of last sub-ADC in the Time-Interleaved ADC should be properly aligned to the ADC reference clock. Clock alignment is usually done using a feedback system that controls the phase. Moreover, the phase difference between each two adjacent sub-ADCs' clock should be the same, phase locked loop and Delay-Locked Loop are good candidates to generate the clocks for time interleaved ADC, because both Voltage Controlled Oscillator in Phase Locked Loop or Voltage Controlled Delay Line in Delay-Locked Loop can be designed using current starved inverter connecting in series, then the circuit in each stage can be the same.

2.5.1 Phase-Locked Loop

Phase-Locked Loop is a feedback system that generates an output signal whose phase is related to the phase of an input signal [46], this feedback system can compare the output phase with the input phase. There are many previous work [49–67] that are presented in this Phase Locked Loop related topics. In Figure 2.12, it can be seen that the input reference clock and PLL output clock are two inputs of the Phase Detector, the phase detector will compare the phase difference between two signals, the output of phase detector will drive the charge pump and generate the control voltage through loop filter. The voltage controlled oscillator will change the frequency when control voltage change. The frequency divider is optional, if we want a higher output frequency, then the frequency divider can be used.

From Figure 2.12, the impedance of the loop filter can be written as:

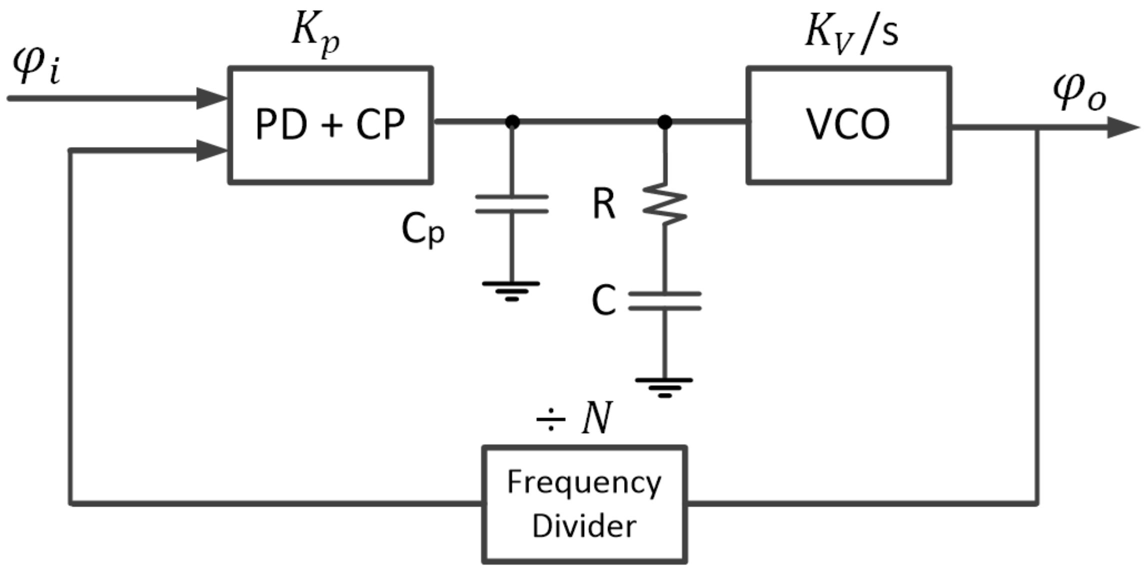


Figure 2.12: Block Diagram of Phase Locked Loop

$$Z(s) = \frac{sRC + 1}{s(sC_pCR + C_p + C)} \tag{2.29}$$

If the frequency divider can divide the frequency by N, the transfer function of the Phase Locked Loop can be written as:

$$H_\phi(s) = \frac{K_p \cdot Z(s) \cdot \dfrac{K_v}{s}}{1 + \dfrac{1}{N} K_p \cdot Z(s) \cdot \dfrac{K_v}{s}} \tag{2.30}$$

where $Z(s)$ is the impedance of the loop filter.

Phase-Locked Loop is an second or third order loop which may has stability issue; Moreover, Phase-Locked Loop will accumulate phase error.

2.5.2 Delay-Locked Loop

Delay-Locked Loop [59, 68–78] is a feedback system that generates output signals whose phase is delayed comparing to the input reference clock phase. The main

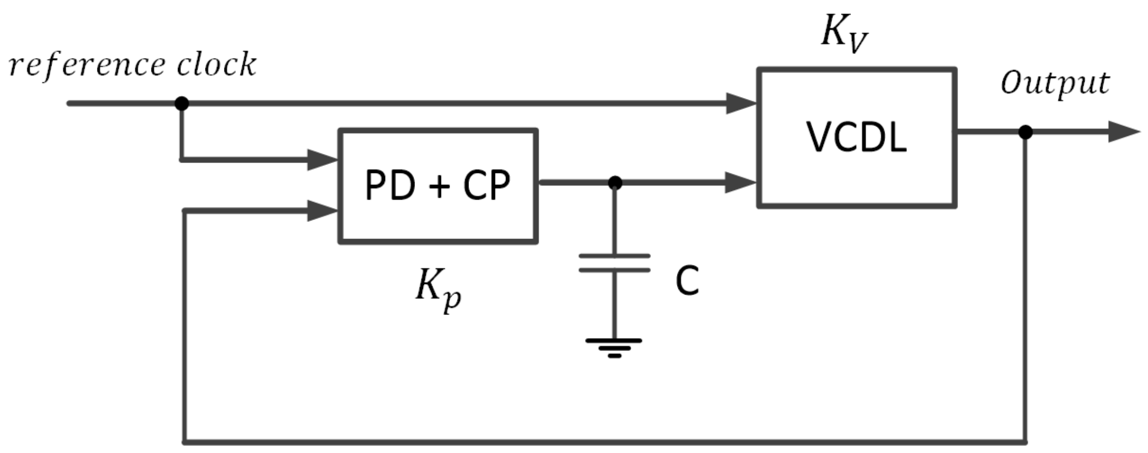


Figure 2.13: Block Diagram of Delay Locked Loop

difference from the PLL is the absence of voltage controlled oscillator and replaced by a voltage controlled delay line. In the Delay-Locked Loop, there is no frequency divider and the loop filter can be a capacitor.

Figure 2.13 shows the block diagram of a traditional Delay-Locked Loop. The phase detector detects the phase difference between reference clock and DLL output clock, then this phase difference will generate a control voltage through the charge pump and loop filter. The Voltage Controlled Delay Line can change the delay when the control voltage change, then this output voltage will be sent back to the phase detector until the last output in the delay line has the same phase with the reference clock.

From Figure 2.13, the transfer function of the Delay-Locked Loop can be written as:

$$H\left(s\right)=\frac{K_pK_v\dfrac{1}{sC}}{1+K_pK_v\dfrac{1}{sC}} \tag{2.31}$$

From equation (2.31), it can be seen that Delay-Locked Loop is a first order loop which is inherently stable; Moreover, Delay-Locked Loop does not accumulate jitter.

2.6 Time Interleaved ADC

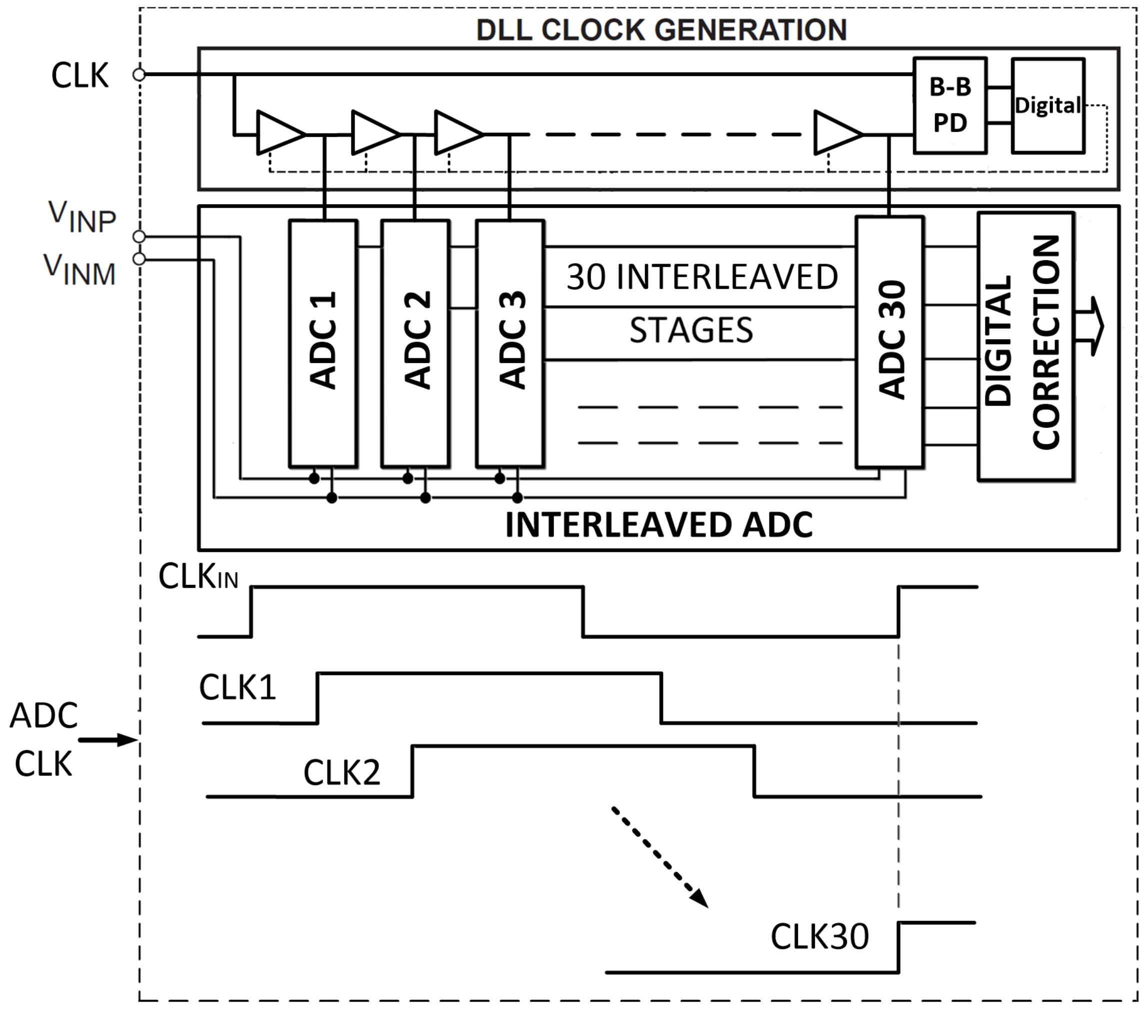


Figure 2.14: System Block of Time Interleaved ADC

Time-interleaved analog-to-digital converters [79–84] is a way to increase the overall system sample rate by using multiple identical ADCs in parallel. Fig 2.14 shows the system block of a 30-stage Time-Interleaved ADC, the reason for 30 stages will be explained in Chapter 5. Ideally the ADC clocks for the sub-ADCs have the same propagation delay between their adjacent clocks. In the time interleaved ADC, each ADC works at a lower sample rate, so if there are N sub-ADCs in the Time-Interleaved ADC, the Time-Interleaved ADC sampling rate is N times faster than each sub-ADC.

In Fig 2.15, if there are N sub-ADCs, and the given ADC has the continuous time input signal x(t) and the discrete time output signal y[n], the total sampling period of the time-interleaved ADC T_{total} should be $N \cdot T_{sub}$, where T_{sub} is the sampling period of each sub-ADC. So from the time interleaved ADC model in [85], the ith sub-ADC, where $i = 0, 1, ..., N-1$, has the sampling edges at

$$\begin{aligned} t_i\left[n\right] &= nT_{total} + iT_{sub} \\ &= \left(nN + i\right) \cdot T_{sub} \end{aligned} \tag{2.32}$$

From Fig 2.15 it can be seen that the input signal is uniformly sampled and the sample edge of each two consecutive clocks are offset by T_{sub}, so the output of the ith sub-ADC can be written as

$$\begin{aligned} y_i\left[n\right] &= x\left(t_i\left[n\right]\right) \\ &= x\left(\left(nN + i\right) \cdot T_{sub}\right) \end{aligned} \tag{2.33}$$

Because the final ADC output is generated through choosing different sub-ADC's output, according to equation 2.33, the final ADC output $y_i\left(n\right)$ can be derived as

$$y\left[n\right] = y_i\left[\frac{n-i}{N}\right] where\ i\ =\ n\ mod\ N \tag{2.34}$$

Define $Y_i\left[n\right]$ as the sub-ADC output $y_i\left[n\right]$ upsampled by N, then

$$Y_i\left[n\right] = \begin{cases} y_i\left[\frac{n-1}{N}\right] & if\ \frac{n-i}{N}\ is\ an\ integer \\ 0 & else \end{cases} \tag{2.35}$$

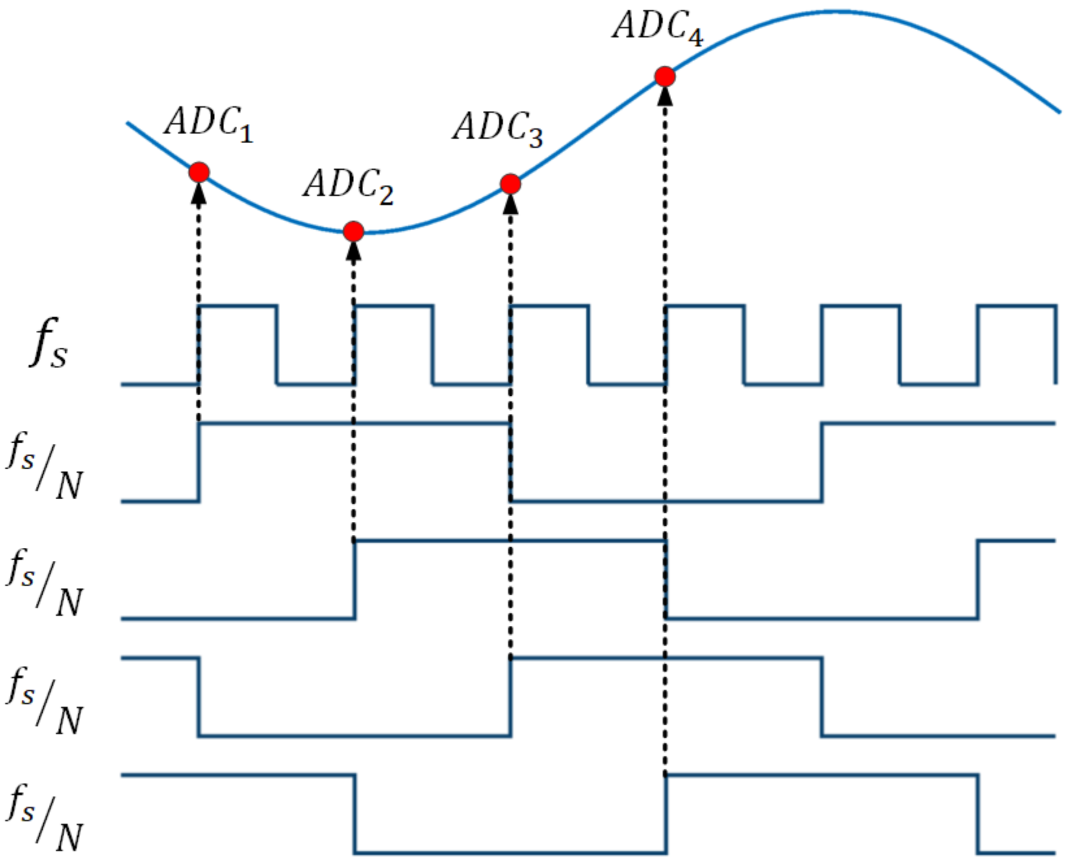


Figure 2.15: Example of Time Interleaved ADC. The sample rate of each sub ADC is f_s/N, the final sampling rate of the time interleaved ADC is f_s. In the example, N = 4

equation 2.35 can be simplified by using

$$\delta_i [n] = \sum_{k=-\infty}^{\infty} f [n - kN - i] \tag{2.36}$$

So the sub-ADC output can be expressed as

$$Y_i (n) = x (nT_{sub}) \cdot \delta [n] \tag{2.37}$$

Thus, the time-interleaved ADC output $y [n]$ in equation 2.34 can be written as

$$y [n] = \sum_{i=0}^{N-1} Y_i [n] \tag{2.38}$$

From equation (2.38), it can be seen that, the output of an ideal time-interleaved ADC can also be expressed as

$$y [n] = x (nT_{sub}) \tag{2.39}$$

This is the expected result from the beginning of this section.

Chapter 3

Low Jitter Delay-Line Chip Implementation

In this chapter, the design of the low jitter delay line is presented. The 10-stage, ultra low jitter voltage controlled delay line was implemented in 180nm IBM cmrf7sf CMOS technology to investigate the low jitter theory in [87]. In accordance with the jitter theory which is presented in [87], the transistors in the delay line are designed to achieve a peak current of 8.2 mA and the delay line can can provide a current step LSB of 1 μA. The input control of the delay line is a binary digital number, these input signals connect to the DAC in the delay line. These DACs are binary-weighted sized transistors which are connected in parallel. The target jitter performance of the delay line is 0.1 psec.

3.1 System Overview

Fig 3.1 shows the system diagram of the low jitter delay line test chip design. In the delay line, there are 10 delay stages. Each delay element in the delay line is

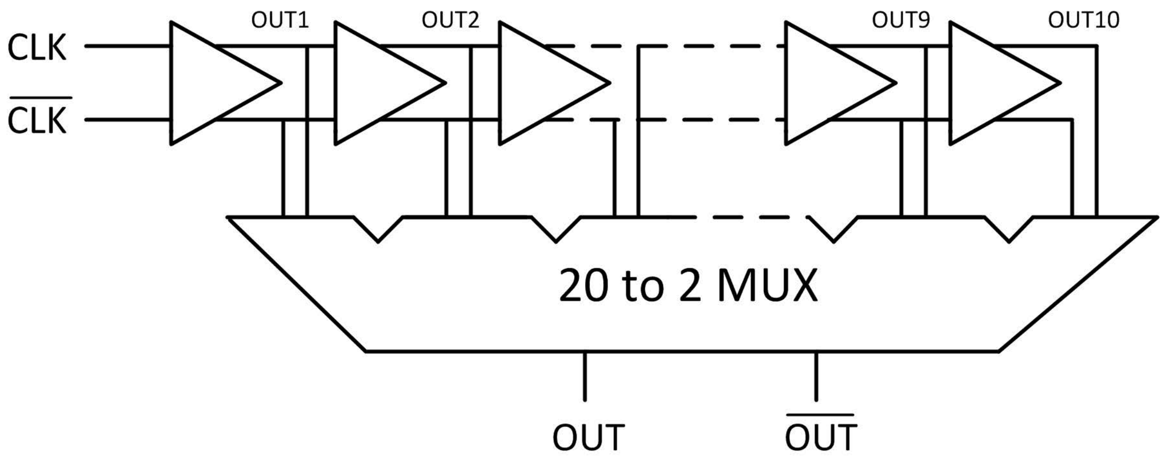


Figure 3.1: System design of the test chip

a pseudo-differential current-starved analog delay stage, so there are two current starved inverters in parallel. Because there are limited number of bond pads, a 20-to-2 multiplexer was used to choose the output of the delay line, so only two bonds are needed to test all 10 delay stages' outputs.

Table 3.1: Pin Description of The Test Chip

Pin Name	Input / Output	Analog / Digital	Pin Description
DLLVDD	Input	Analog	Power Supply for Delay Line, 1.8v
DLLVSS	Input	Analog	Power Supply for Delay Line, 0v
VDD!	Input	Analog	Power Supply for Digital Logic, 1.8v
GND!	Input	Analog	Power Supply for Digital Logic, 0v
IVDD	Input	Analog	Power Supply for Analog Buffer, 1.8v
IVSS	Input	Analog	Power Supply for Analog Buffer, 0v
DLLCLK	Input	Digital	Delay Line Input Clock
DLLCLKBAR	Input	Digital	Delay Lline Input Clock Bar
MUX	Output	Digital	Multiplexer Output
MUXBAR	Output	Digital	Multiplexer Output Bar
SHIFTRESET	Input	Digital	Reset signal for digital shift register
ENABLE	Input	Digital	Enable signal for digital shift register
SI	Input	Digital	Digital shift register serial input data
SHIFTCLK	Input	Digital	Shift Register Clock

Table 3.1 lists the pin descriptions of the designed test chip. Because the low jitter delay line power needs to be measured to explore the jitter theory in [87], the delay line has its own power supply DLLVDD and DLLVSS, separate from VDD! and GND! which are used by other parts of the circuit. There are 14 pins in total for the design test low jitter delay line, OUT and $\overline{OUT}$ are output pins which are used to choose the differential output clock from OUT1 to OUT10 in Fig 3.1.

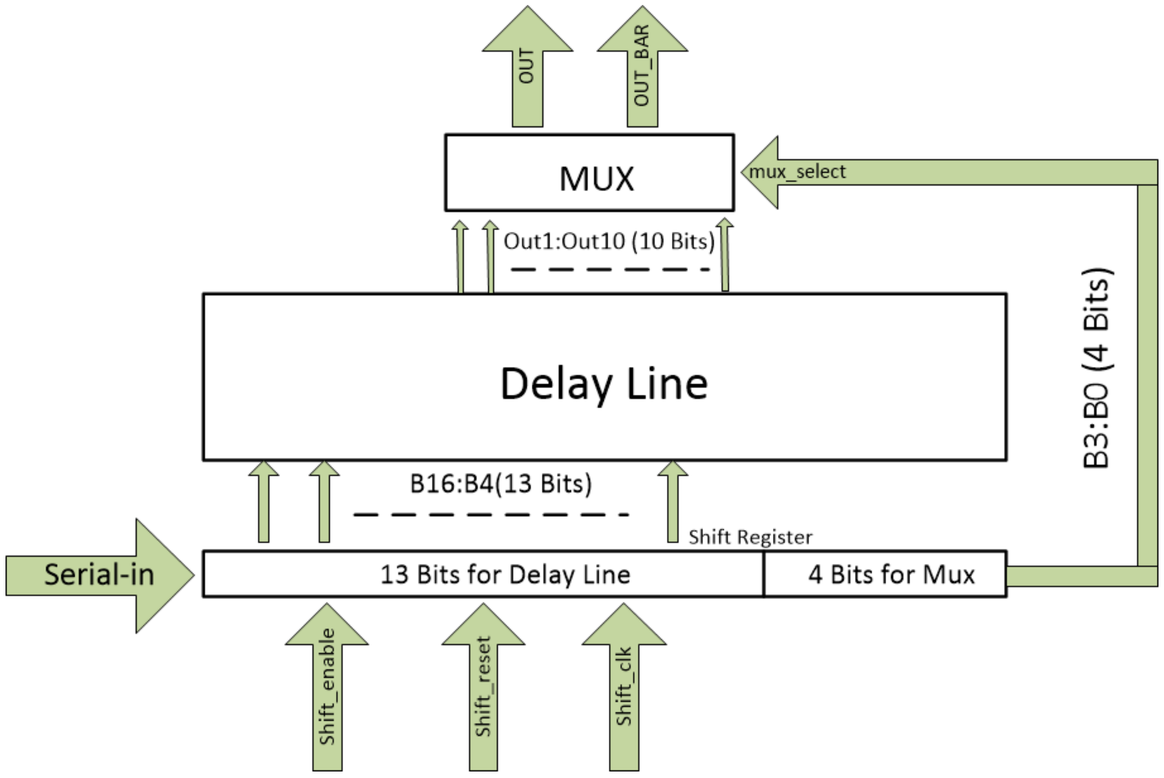


Figure 3.2: Digital control block diagram of chip

Fig 3.2 shows the digital control block diagram of the designed chip. There is a 17 bit shift register which are used to control the mux input and delay line control DAC. Bit 0 to Bit 3 are connected to the multiplexer. There is a binary to one-hot code decoder in the MUX block, so only one delay line output pair (OUT and OUT_BAR) can be selected each time. Bit 4 to Bit 16 of the shift register are used to control the delay of the delay line, because there is a 13 Bit DAC in each delay stage, the serial input can be connected directly to the DAC input. The detail of the delay stage will be explained in Section 3.2.

Fig 3.3 shows the chip layout of the low jitter delay line. The total size of the chip is 1.5mm x 1.5mm. Fig 3.4 is a magnified view of the delay line layout, showing the detail of the 10 stages in the delay line. The dimensions of the delay line are 948μm x 109μm.

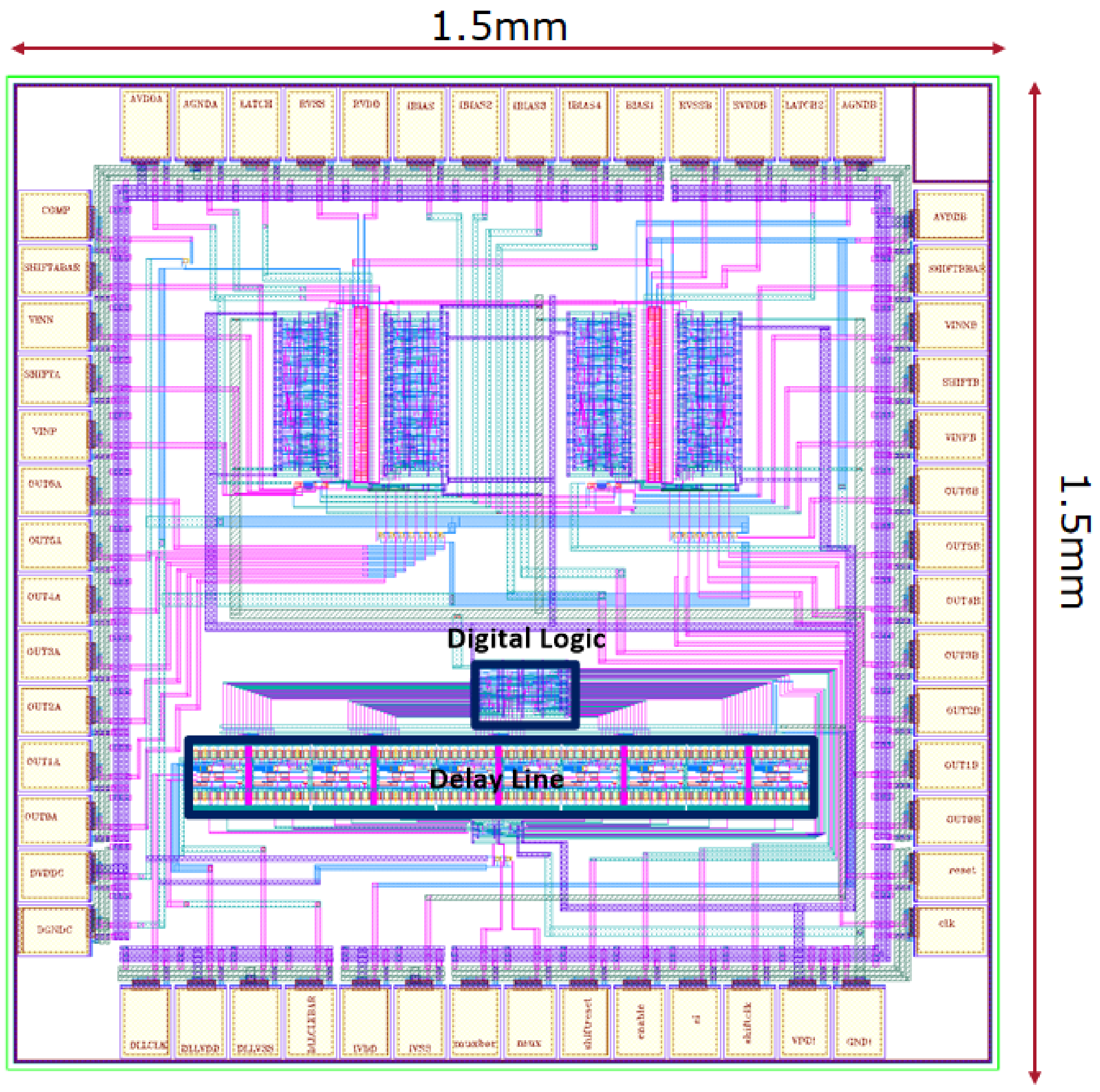


Figure 3.3: Layout of the Low jitter Delay Line

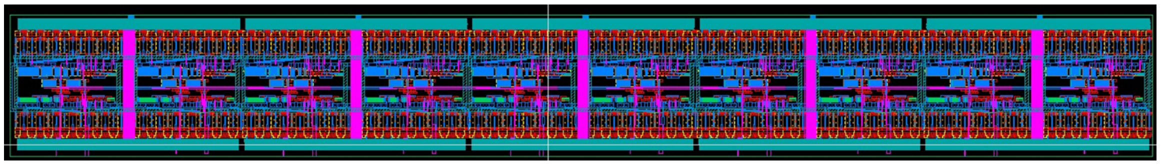


Figure 3.4: Zoom in layout of the delay line

3.2 Digital Controlled Delay Cell

Each delay stage of the delay line shown in Fig 3.1 is a pseudo-differential delay stage. Fig 3.5 shows the block diagram of each delay stage. Two delay cells are connected to $CLKi$ and $\overline{CLKi}$ respectively, the stage delay is set by digital control of the peak current i_{SW} available for the gate output transition. Two smaller sized, fixed current, cross coupled inverters connect to the outputs of the two delay cells to reinforce a 180^o phase relationship at the pseudo-differential outputs. Because the jitter will accumulate in each delay stage, with using the pseudo-differential architecture, the total number of delay stages are decreased to half, the jitter performance will improve. Moreover, the propagation delay of the delay line will decrease by half, the input frequency of delay line can be doubled.

The schematic of each delay cell is shown in Fig 3.6. The source current of the inverter PMOS/NMOS is controlled by a current source DAC. The DAC input control is a digital code which is from the shift register. Transistor sizing for the DAC is designed to provide a current step LSB of 1 μA, which corresponds to approximately 0.1 ps step resolution for the delay line. The required maximum current range of at least 4.5 mA from the DAC results from the jitter theory in [87], which is covered in more detail in Section 2.4. To explore the design space of the chip, more transistors are used to exceed the design requirement. The designed circuit in Fig 3.6 can achieve a maximum current of 8.2 mA. So when the 13 bit digital control input are all ones, the I_{PEAK} of the source current is 8.2 mA. These values for I_{MIN} and I_{MAX} result in the DAC resolution of 13 bits.

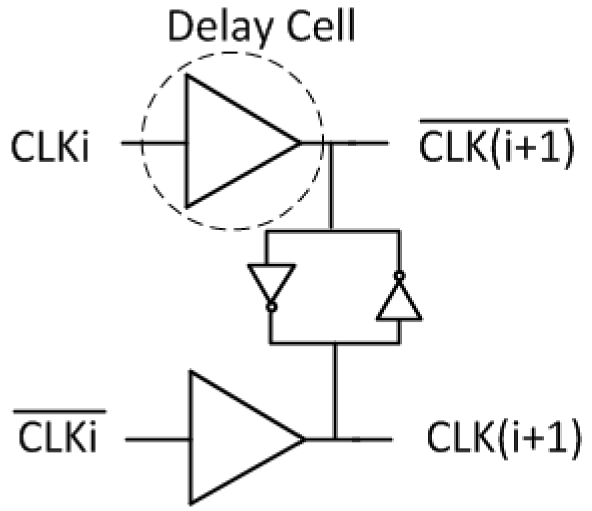


Figure 3.5: Schematic of Single Delay Stage.

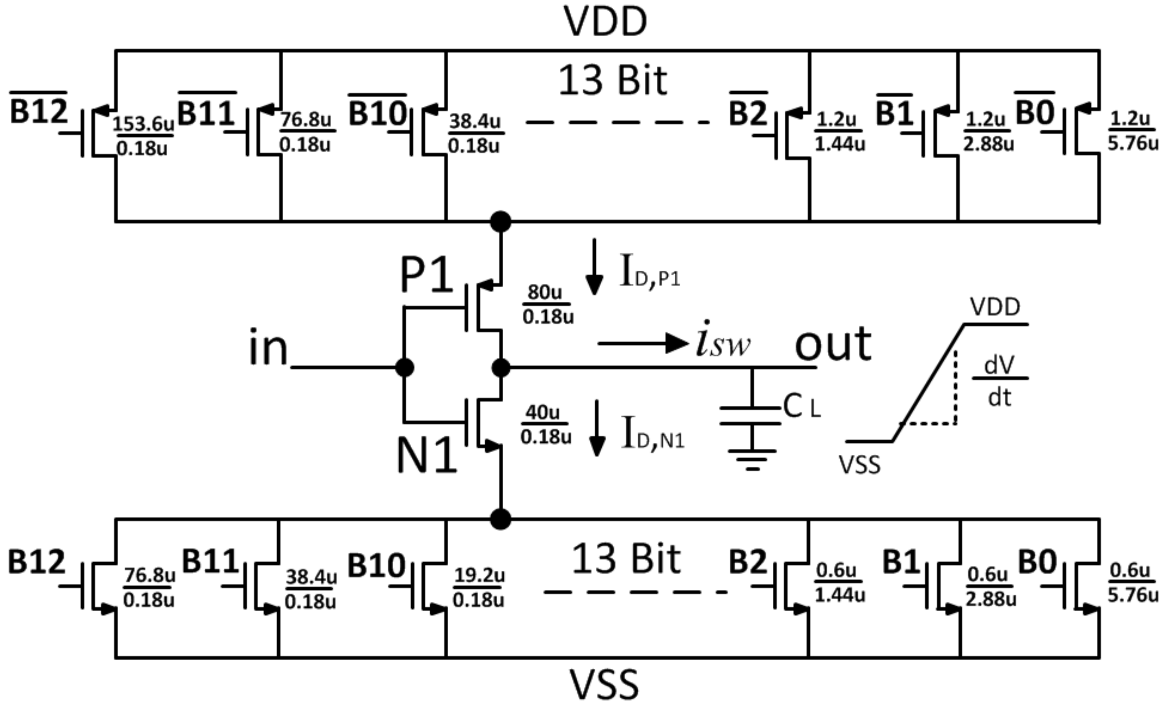

VDD
13 Bit
$\overline{\mathrm{B12}}$ 153.6u/0.18u
$\overline{\mathrm{B11}}$ 76.8u/0.18u
$\overline{\mathrm{B10}}$ 38.4u/0.18u
$\overline{\mathrm{B2}}$ 1.2u/1.44u
$\overline{\mathrm{B1}}$ 1.2u/2.88u
$\overline{\mathrm{B0}}$ 1.2u/5.76u
P1 80u/0.18u
$I_{D,P1}$
in
i_{sw}
out
C_L
N1 40u/0.18u
$I_{D,N1}$
VDD
dV/dt
VSS
B12 76.8u/0.18u
B11 38.4u/0.18u
B10 19.2u/0.18u
13 Bit
B2 0.6u/1.44u
B1 0.6u/2.88u
B0 0.6u/5.76u
VSS

Figure 3.6: Schematic of Single Delay Cell.

3.2.1 Propagation Delay

Fig 3.6 also shows a simple delay model of each current starved delay stage of the delay line. In this model, according to Section 2.2, the simplified propagation delay expression is

$$t_{PD} \approx \frac{C_L V_{DD}}{2I_{peak}} \tag{3.1}$$

where I_{peak} is the peak current of the drain current P1 and N1. From (3.1) we see that (to first order) the gate propagation delay is inversely proportional to the available current i_{SW} as controlled by the DAC.

3.2.2 Jitter in Delay Line

According to the theory in [87], as jitter accumulates in the DLL delay line the RMS jitter increases proportional to the square root of delay ΔT and can be characterized by a figure of merit κ:

$$\sigma_{\Delta T} = \kappa\sqrt{\Delta T} \tag{3.2}$$

given independent jitter errors in each stage. For an individual delay stage with delay $\Delta T = t_{PD}$, (2.5) gives for the individual stage jitter σ_{stage}

$$\sigma_{stage} = \kappa\sqrt{t_{PD}} \tag{3.3}$$

In the proposed delay line, there are 10 delay stages, and different jitter will be observed at each output. The jitter at output 1 will be a combination of contributions from the original input σ_{input} and the jitter contributed by stage 1 σ_{stage1}. Since these are independent, they will add in RMS fashion. Generalizing, the jitter of the nth output is given by

$$\sigma_{out(n)} = \sqrt{\sigma^2_{input} + \sigma^2_{stage1} + ... + \sigma^2_{stage(n)}} \tag{3.4}$$

Because the delay stages in the delay line are all the same, we assume the magnitude of the jitter contributed by each to be the same, so we can infer each stage jitter to be:

$$\sigma_{stage} = \sqrt{\frac{1}{10}\left(\sigma^2_{out10} - \sigma^2_{input}\right)} \tag{3.5}$$

For a total added jitter less than 1 psec from σ_{input} to σ_{out10}, (3.5) implies σ_{stage} is less than 0.32 psec RMS. From (3.3), this gives a required $\kappa = 3.2E-8\sqrt{\text{sec}}$ with $t_{PD} = 100$ psec.

And the total jitter of ten stages is:

$$\sigma_{total} = \sqrt{\frac{10}{9}\left(\sigma^2_{out10} - \sigma^2_{out1}\right)} \tag{3.6}$$

Chapter 4

Low Jitter Delay-Line Chip Testing and Analysis

4.1 Evaluation Board Design

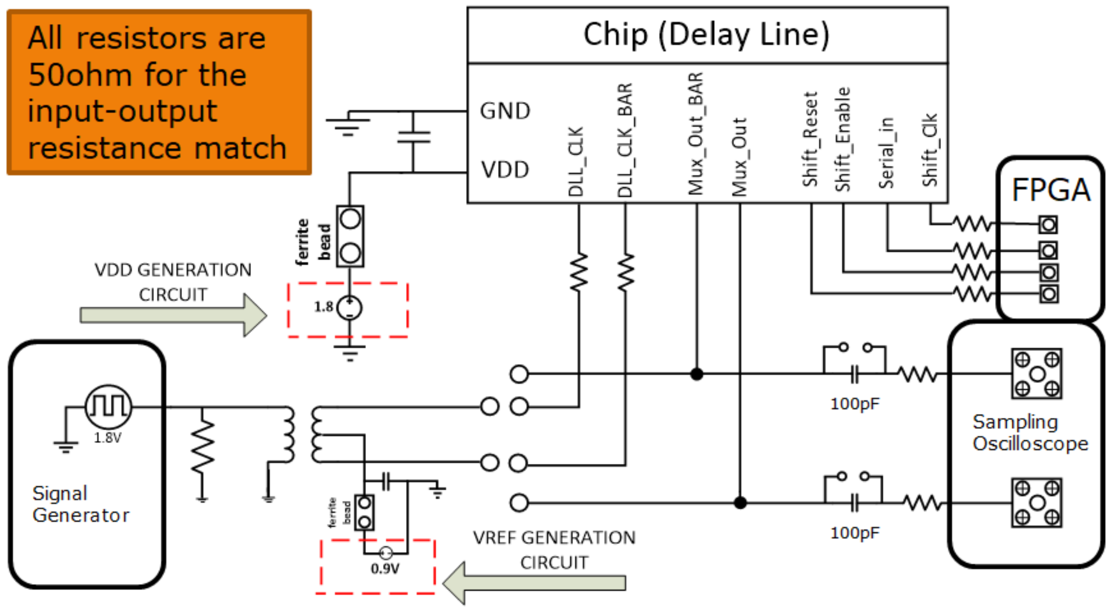


Figure 4.1: System Level Printed Circuit Board Design

Fig 4.1 shows the system level design of printed circuit board (PCB) for evaluating of the test chip. In the circuit, the signal generator is a crystal oscillator which generates a very stable and low jitter clock. A transformer is used to convert the single ended signal from the crystal oscillator into the differential clocks for DLLCLK and DLLCLK bar. This is required because of the pseudo differential architecture in the delay line. The center tap of the transformer connected to a 0.9 v voltage source to give a reference for the differential clocks. The 1.8 v power supply of the chip is generated by a linear regulator with low dropout voltage. The output of the

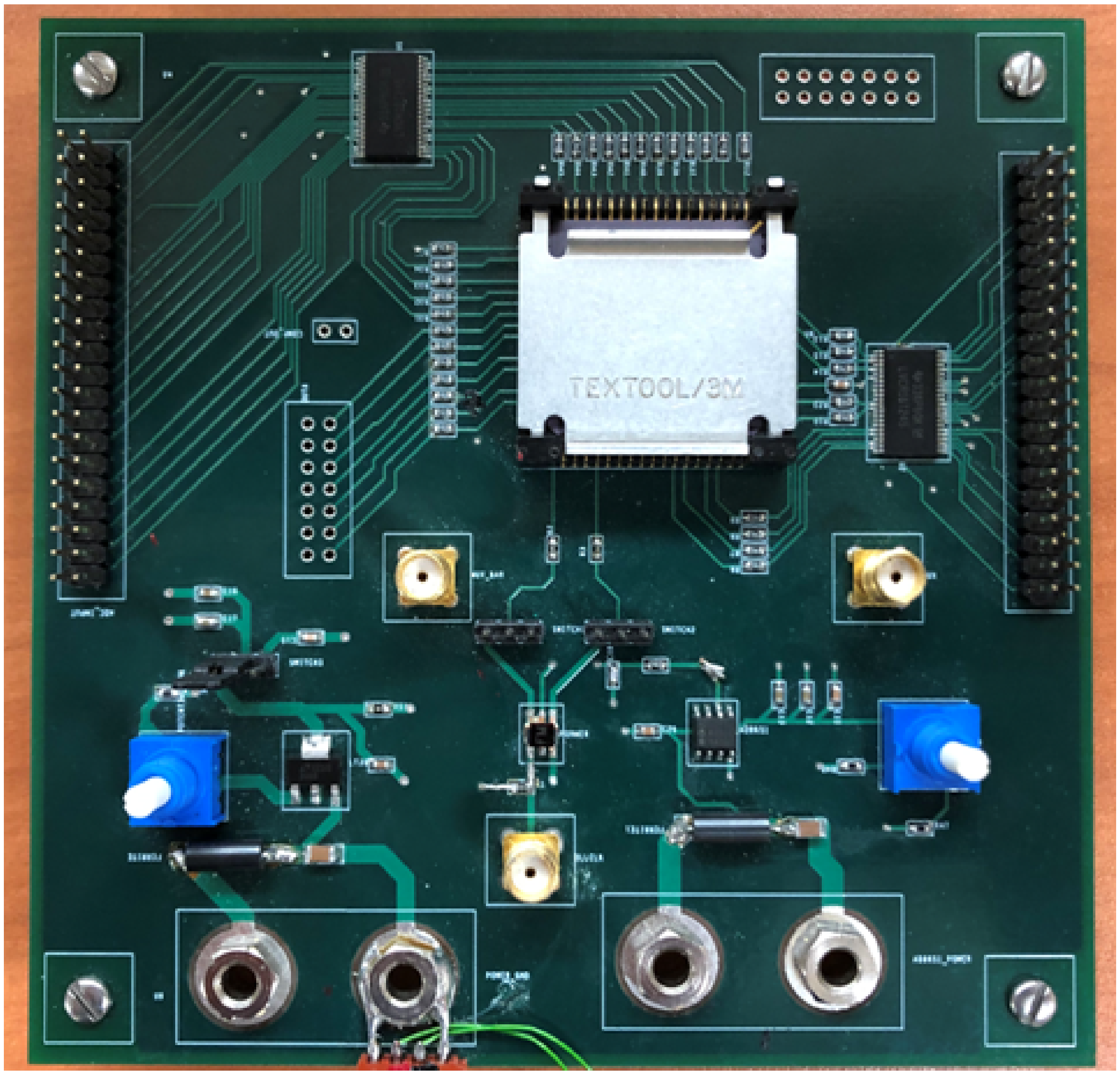


Figure 4.2: Photo of the PCB

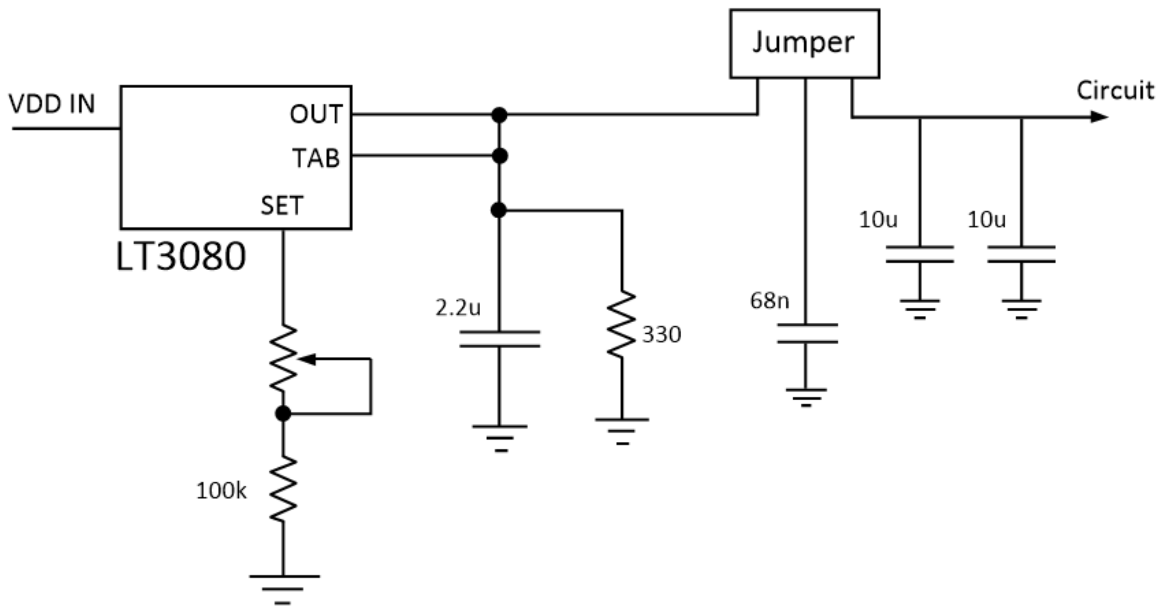


Figure 4.3: Schematic of the regulator on PCB (resistor unit is Ohm, capacitor unit is Farad)

chip connect to SMA connectors which were connected to a sampling oscilloscope during testing.

Fig 4.2 shows the photo of the PCB which is used to evaluate the test chip. Since there is another ADC project on this chip, some circuits on the PCB are used to deactivate the other ADC and eliminate unwanted interference while evaluating the low jitter delay line.

In order to avoid signal reflection, 50 Ω resistors are used for impedance matching of high speed signals. For an ultra low jitter delay line, the clock distribution circuitry should be placed as close as possible to the delay line input, to help prevent clock jitter degradation. PCB trace dimensions (length, width, and depth) affect the characteristic impedance (Z_0) of the trace; therefore the output impedance of high speed signals must be matched to the characteristic impedance of the traces.

Fig 4.3 shows the schematic of the regulator with low dropout voltage. The regulator chip is LT3080. A 3-pin jumper is placed on the board that allows selection between the regulated voltage and direct voltage from an off board power supply. A potentiometer is used to generate different value of the regulated voltage.

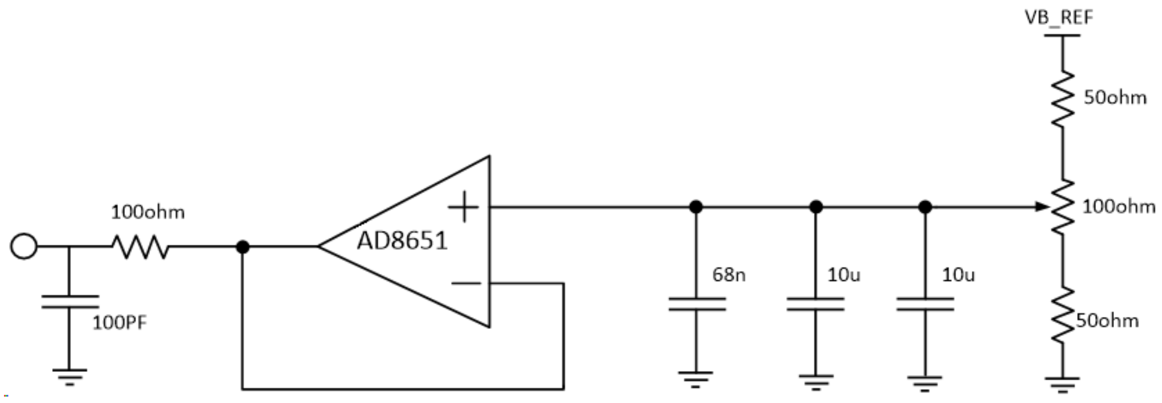


Figure 4.4: Schematic of the 0.9v reference voltage generation circuit

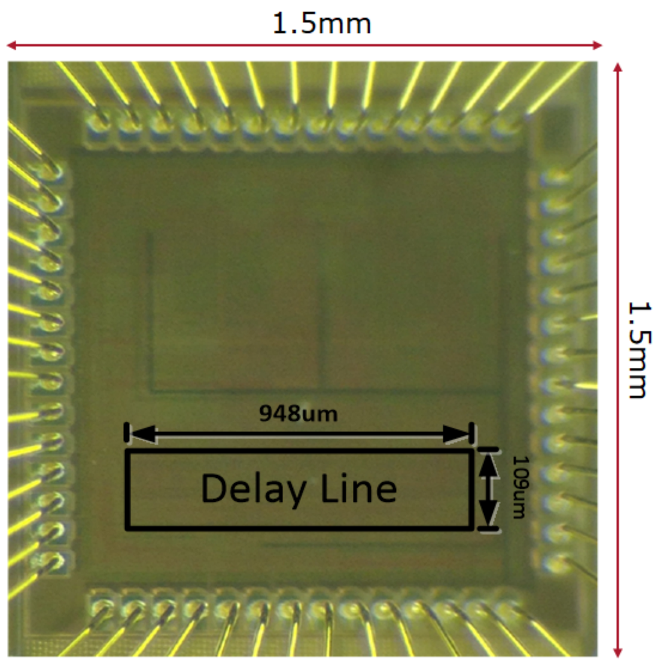


Figure 4.5: Chip Micrograph of Low Jitter Delay Line.

Fig 4.4 shows the schematic of the 0.9 v reference voltage generation circuit for the center tap of the transformer to generate a differential clock for the delay line. A potentiometer is used to generate the voltage , and a buffer is used followed a low

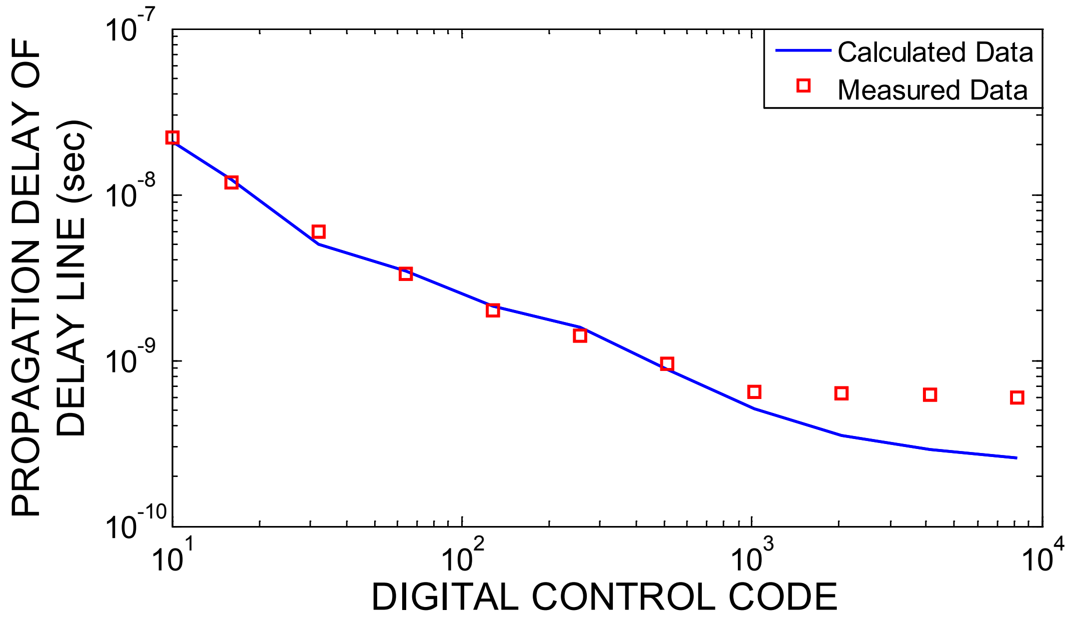


Figure 4.6: Propagation Delay of the Delay Line

pass filter to suppress the noise of the reference voltage.

The proposed low-jitter delay line was fabricated in a 0.18 μm CMOS process, The die photo is shown in Fig 4.5. The size of the low jitter delay line is 948 um x 109 um. Since the test chip did not integrate the interleaved ADCs, the quality of the stage clocks was measured using a multiplexer.

4.2 Measured Results - Propagation Delay

Fig 4.6 shows the measured propagation delay of the delay line. Over the full range of the DAC control word, t_{PD} varies from 22.2 ns to 0.6 ns. Fig 4.6 shows good agreement between the measured delay and predicted delay from equation (3.1) using an effective load capacitance of $C_L = 58.3fF$. Ideally, under this range of propagation delay, a DLL with the proposed delay line can lock the clock from 50MHz to 1.6GHz. However, the 180nm CMOS will limit the speed of switching.

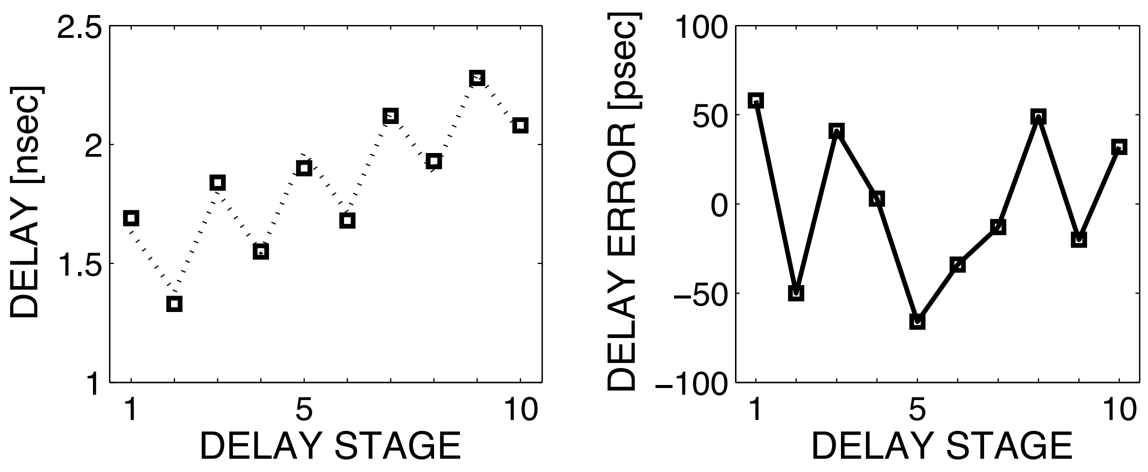


Figure 4.7: (a) Propagation delay of each stage ($n_{CTL} = 1024$) (b) Nonuniformity of stage delay in psec

Fig 4.7 shows the measured delay from the input clock edge to the output of each delay stage. The observed "odd-even" pattern is due to a nonideality in the layout of the multiplexer in Fig 3.1. The inherent delay nonuniformity of the DLL can be estimated by fitting and subtracting out an estimated odd-even pattern, shown in Fig 4.7b. The remaining systematic error is ± 60 psec pk-pk, within the range that can be calibrated out by [5, 24, 88].

4.3 Measured Results - Jitter

Fig 4.8 shows the measured clock jitter at each output. Due to the small cross coupled inverter in pseudo differential delay stage and the additional unmodeled jitter of the multiplexer, the individual stage jitter was inferred using (3.5) from the differences in the (higher jitter) measured signals. Variance σ^2 can be obtained from the measured jitter σ, In ideal conditions, the variance should be linear. In Fig 4.8, the slope of the curve is the variance of a single stage in delay line. The jitter of single stage in the delay line can be obtained from taking the square root

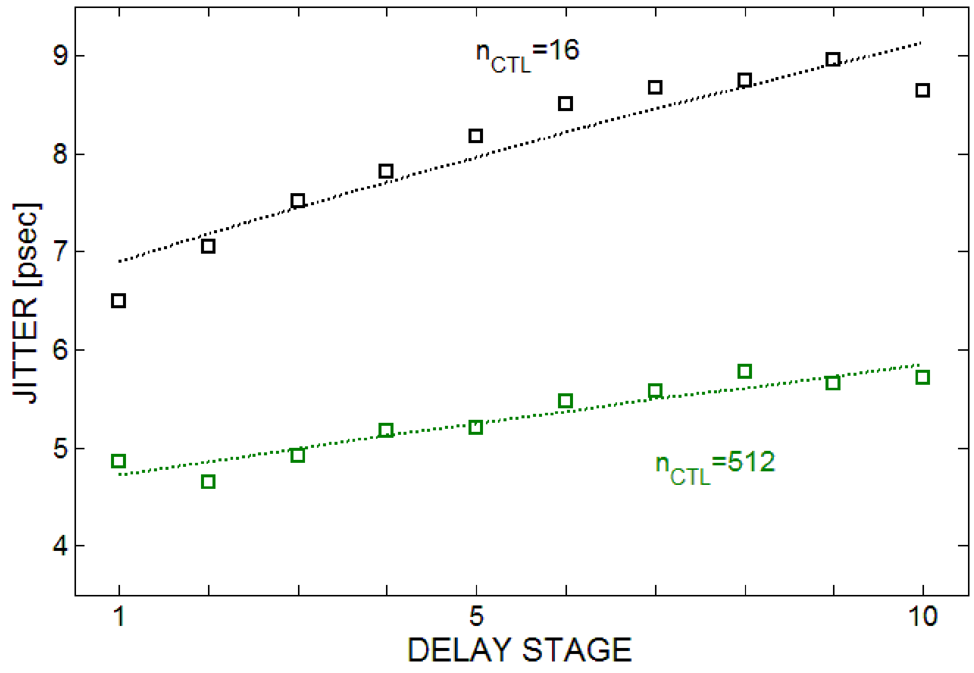


Figure 4.8: Jitter of the Clock under different Digital Control Code

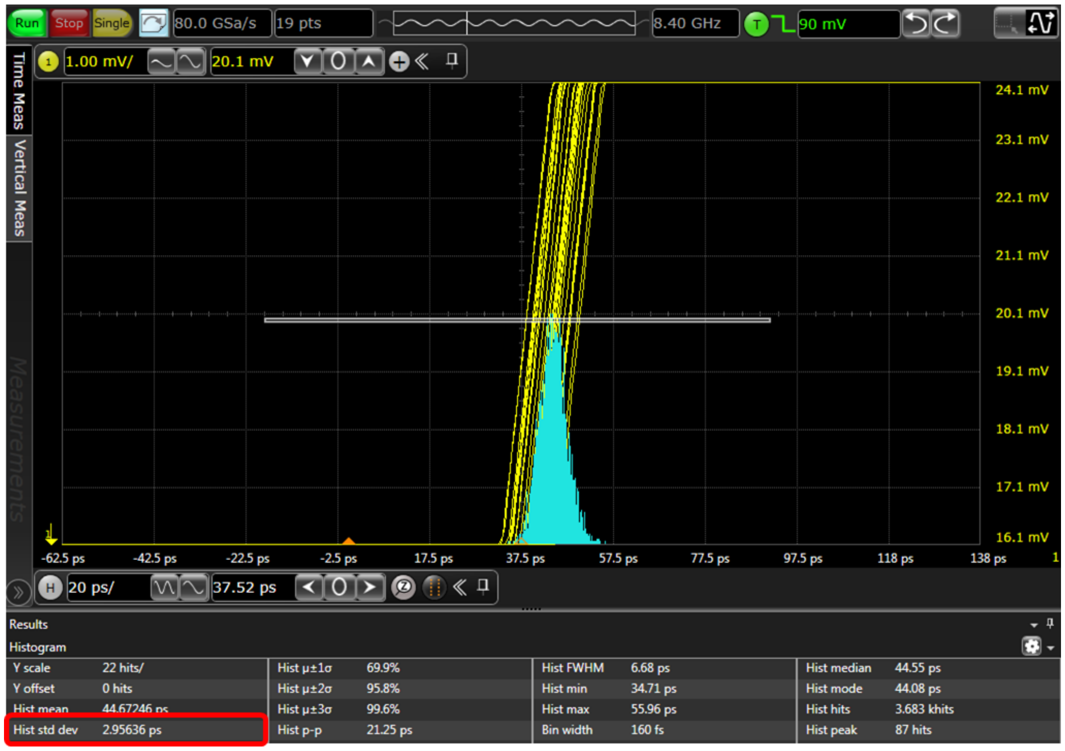
of the variance. Fig 4.9 and Fig 4.10 show the jitter of 1st stage and 10th stage respectively. From equation (3.5), the jitter of single delay stage is

$$\sigma_{stage} = \sqrt{\frac{1}{9}\left(\underbrace{2.97346ps^2}_{\sigma_{out10}} - \underbrace{2.95636ps^2}_{\sigma_{out1}}\right)} = 0.10614psrms \tag{4.1}$$

in good agreement with the design target from Chapter 3. And the total jitter of the delay line can also be obtained from equation (4.2):

$$\begin{aligned}\sigma_{total} &= \sqrt{\frac{10}{9}\left(\sigma_{out10}^2 - \sigma_{out1}^2\right)} \\ &= \sqrt{\frac{10}{9}\left(2.97346ps^2 - 2.95636ps^2\right)} = 0.33566psrms \end{aligned} \tag{4.2}$$

For each digital control code, we can calculate a "best-fit" expected jitter-vs-

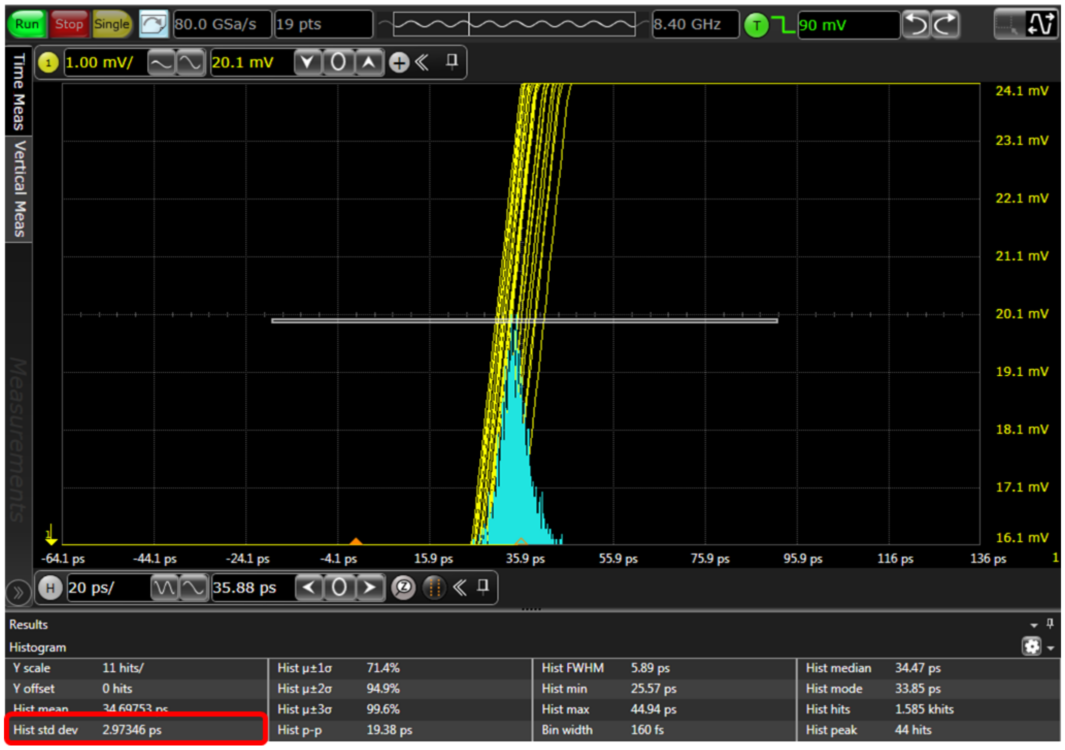


Results

Histogram

Y scale	22 hits/	Hist μ±1σ	69.9%	Hist FWHM	6.68 ps	Hist median	44.55 ps
Y offset	0 hits	Hist μ±2σ	95.8%	Hist min	34.71 ps	Hist mode	44.08 ps
Hist mean	44.67246 ps	Hist μ±3σ	99.6%	Hist max	55.96 ps	Hist hits	3.683 khits
Hist std dev	2.95636 ps	Hist p-p	21.25 ps	Bin width	160 fs	Hist peak	87 hits

Figure 4.9: Measured jitter of 1st stage, σ=2.95636ps



Results

Histogram

Y scale	11 hits/	Hist μ±1σ	71.4%	Hist FWHM	5.89 ps	Hist median	34.47 ps
Y offset	0 hits	Hist μ±2σ	94.9%	Hist min	25.57 ps	Hist mode	33.85 ps
Hist mean	34.69753 ps	Hist μ±3σ	99.6%	Hist max	44.94 ps	Hist hits	1.585 khits
Hist std dev	2.97346 ps	Hist p-p	19.38 ps	Bin width	160 fs	Hist peak	44 hits

Figure 4.10: Measured jitter of 10th stage, σ=2.97346ps

delay characteristic based on the κ model derived in Section 2.3. These best-fit characteristics are plotted as dashed lines in Fig 4.8.

4.4 Measured Results - κ

Table 4.1 shows the κ values from measured data. When the digital control code is $2^{13}-1$(biggest number), 2^5 and 2^4, and the difference of propagation delay is around 0.5 ns, we choose these three points to calculate the κ. For lower values of the DAC control code (longer t_{PD}), the measured κ fit is within approximately 50% of the predicted theoretical value. At higher values of the DAC code, corresponding to shorter t_{PD}, significant deviation is observed. The deviation due to the high jitter floor of the available test instrumentation and the additional unmodeled multiplexer jitter. In an integrated interleaved ADC system, these deviations from the model would not be issues since jitter performance would be inferred more directly from behavior of ENOB versus input signal frequency.

Table 4.1: Calculated and Measured κ ($\sqrt{sec}$).

Control Code	measured t_{PD} per stage	Measured jitter per stage	κ fit from measured jitter	I_{peak} from simulation	Calculated κ from I_{peak}	Measured κ / Calculated κ
$2^{13}-1$	0.06 ns	0.11ps	1.3E-8	8.16mA	1.81E-9	7.1
2^5	0.60 ns	1.47ps	5.7E-8	0.51mA	2.98E-8	1.9
2^4	1.18 ns	2.37ps	6.5E-8	0.39mA	5.68E-8	1.14

From Table 4.1, it can be seen that when the digital control code is small and the jitter is big, the measured κ is within 50% error comparing to the calculated value, but when the digital control code is big and the jitter is very small, there is a big deviation from measured κ to the calculated value due to the small cross coupled inverter in each pseudo differential delay stage and the additional multiplexer. From the jitter theory in [87], the smaller the κ is, the better the gate's ability to resolve time accurately, therefore the jitter is smaller. When the digital control code is "$2^{13} - 1$", the jitter is very small and the measured jitter of single delay stage is smaller than 0.2 ps. At these time resolution, the jitter of the cross coupled inverters in pseudo differential delay stage cannot be neglected and it can lead to κ to be bigger. When the digital control code is big, all PMOS devices are on in the circuit, and the capacitance is very big, thus the slope of the clock edge is smaller. Therefore jitter in vertical direction will also be measured by the oscilloscope, and will also increase the value of κ. The jitter in the multiplexer cannot be neglected either at this time.

The measured power consumption is 9.7 mW with the maximum I_{peak}, whose measured average value is 5.4 mA. From (2.12), using the calculated κ, the expected power consumption is 5.1 mW for the pseudo-differential delay line. This is within 50% of the measured data, consistent with the order-of-magnitude jitter-power theory in [87].

Chapter 5

Low Jitter Delay-Locked Loop Chip Implementation

The main challenge of time interleaved ADC is maintaining acceptable effective-number-of-bits (ENOB), which can be degraded by jitter in the sample clock. Although several ADCs [20–22] published to date can achieve acceptable performance with calibration of systematic clock phase mismatch [23], these work do not address the challenge of providing a low jitter sample clock for interleaved ADCs. In Chapter 3, an open loop delay line was explored, however an interleaved ADC needs a Delay-Locked Loop to make interleaved clocks work. In this chapter, a low jitter Delay-Locked Loop is proposed to investigate the jitter theory in [87].

The system block diagram of a self-calibrating interleaved ADC with proposed DLL output sample clocks is shown in Fig 2.14. In this work, the high speed on-chip sampling clock phases are developed using a Delay-Locked loop. This significantly eases system-level requirements, since the input clock need only be a low cost, low frequency ($< f_s/$(Number of stage)MHz), low jitter source. Jitter accumulation in the DLL delay string is minimal, unlike the integration over time of jitter in a phase-locked loop (PLL) voltage controlled oscillator (VCO). Therefore delay stages with moderate power dissipation can be used for clock generation. Since the DLL phases are used individually as the ADC sample clocks, there is no need for techniques (such as edge combiners [91]) to synthesize the higher sample clock frequency f_s. In order to achieve 3 GS/s to 3.6 GS/s speed for RF ADC application, there are 30 stages in DLL's delay line due to 180nm CMOS speed limitation, the propagation delay of each delay stage is about 200 ps to 300 ps if we want to achieve the target I_{PEAK}. Each ADC connects to one of thirty DLL output clocks to sample the input analog signal. The speed of this interleaved ADC will be 30× faster than each single ADC.

In this chapter, a Delay-Locked Loop is presented with low jitter delay line based on low jitter theory and is used to generate a high frequency, ultra low jitter interleaved sample clock phases from a low-cost, low frequency reference source. The DLL can work from 80MHz to 120MHz with design target of 0.1 psrms jitter. Because there are 30 delay stages in the delay line, the time interleaved ADC sampling frequency can be 2.4 GHz to 3.6 GHz range which is suitable for the application in Fig 1.1.

5.1 System Overview of Low Jitter Delay-Locked Loop

Figure 5.1 shows the system diagram of the proposed low jitter Delay-Locked Loop. The reference clock and 30th DLL output (last output) are connected to the Bang-Bang phase detector. The output of Bang-Bang phase detector is either '1" or "0". According to the output of Bang-Bang phase detector, the coarse control block will change the coarse control code of the low jitter delay line, changing the time

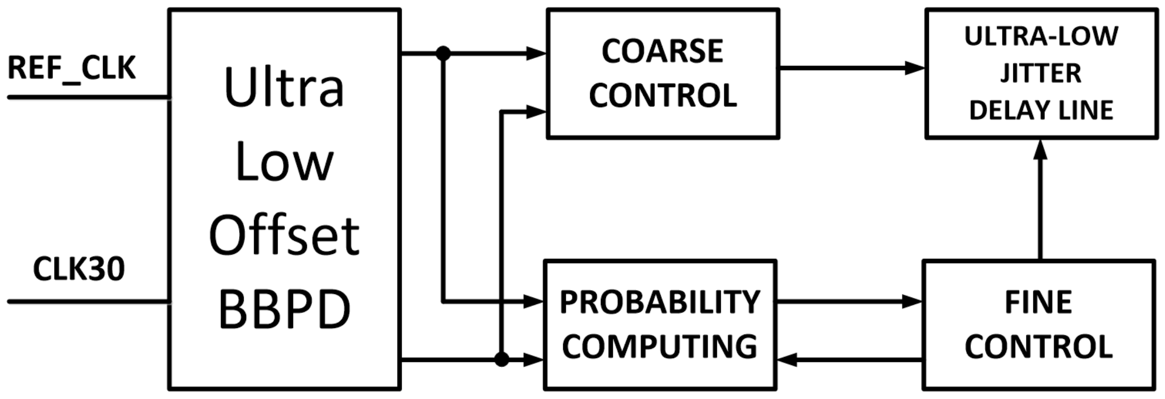


Figure 5.1: Low Jitter Delay-Locked Loop System Diagram

delay. After jumping into the fine control mode, the probability computing block will determine if the fine control code should continue to change or stop to change. After fine control control block stops changing, the Delay-Locked Loop is considered fully locked.

Fig 5.2 shows the jitter source of a traditional Delay-Locked Loop. It can be seen that there are four main sources of jitter in a traditional DLL: (1) the reference clock, (2) phase detector (PD), (3) charge pump (CP), and (4) the delay cells [92]. The first jitter source is from the reference clock. This jitter is not generated in the DLL, it is generated by the external clock source. Since the random jitter between each stage of delay cell is uncorrelated, the reference clock jitter will directly pass to the output of the DLL. The second jitter source is from the phase detector of the Delay-Locked Loop. According to [92], the bigger value of stage number, X, in delay line has bigger value of jitter in the DLL, and this jitter depends on the total number of stage, N, in delay line, but the last stage output of the delay line will not change and it does not depend on the total number of stage N. The third jitter source is due to the charge pump of the Delay-Locked Loop. According to [92], the jitter at the Xth stage of Delay Line due to the charge pump is even bigger than the jitter due to the Voltage Controlled Delay Line. The fourth jitter source is from

Table 5.1: Pin Description of Proposed Delay-Locked Loop

Pin Name	Input /Output	Analog /Digital	Pin Description
DL_VDD	Input	Analog	Power Supply for Delay Line, 1.8v
DL_VSS	Input	Analog	Power Supply for Delay Line, 0v
DB_VDD	Input	Analog	Power Supply for Delay Line Output Buffer, 1.8v
DB_VSS	Input	Analog	Power Supply for Delay Line Output Buffer, 0v
CAP_CTRL	Input	Analog	Control Voltage for Varactor in Delay Line
AIN1	Input	Analog	Sinusoidal Input waveform for differential S/H circuit
AIN2	Input	Analog	Sinusoidal Input waveform for differential S/H circuit
DUMMY_VDD	Input	Analog	Power Supply Dummy Circuits, 1.8v
DUMMY_VSS	Input	Analog	Power Supply Dummy Circuits, 0v
ANALOGOUT	Output	Analog	Output of Source Follower
ANALOGOUT_BAR	Output	Analog	Output of Source Follower
REF_CLK	Input	Digital	Reference Clock for DLL
SLOW_CLK	Input	Digital	Clock for Digital Circuit Input Shift Register
RESET	Input	Digital	Reset Signal for Digital Circuit
SHIFTSERIDATA	Input	Digital	Digital Shift Register Serial Input Data
MUXSERIDATA	Input	Digital	Shift Register Serial Input Data for Analog Multiplexer
DVDD	Input	Digital	Power Supply for Digital Circuit, 1.8v
VSS	Input	Digital	Power Supply for Digital Circuit, 0v
SHIFTWE	Input	Digital	Shift Register Write Enable
SHIFTRST	Input	Digital	Shift Register Reset
MUXWE	Input	Digital	Analog Mux Input Data Shift Register Write Enable
MUXRST	Input	Digital	Analog Mux Input Data Shift Register Reset
HANDENABLE	Input	Digital	Automatic Manual Mode Switch
DIGITAL_OUT	Output	Digital	Digital Clock of Delay Line

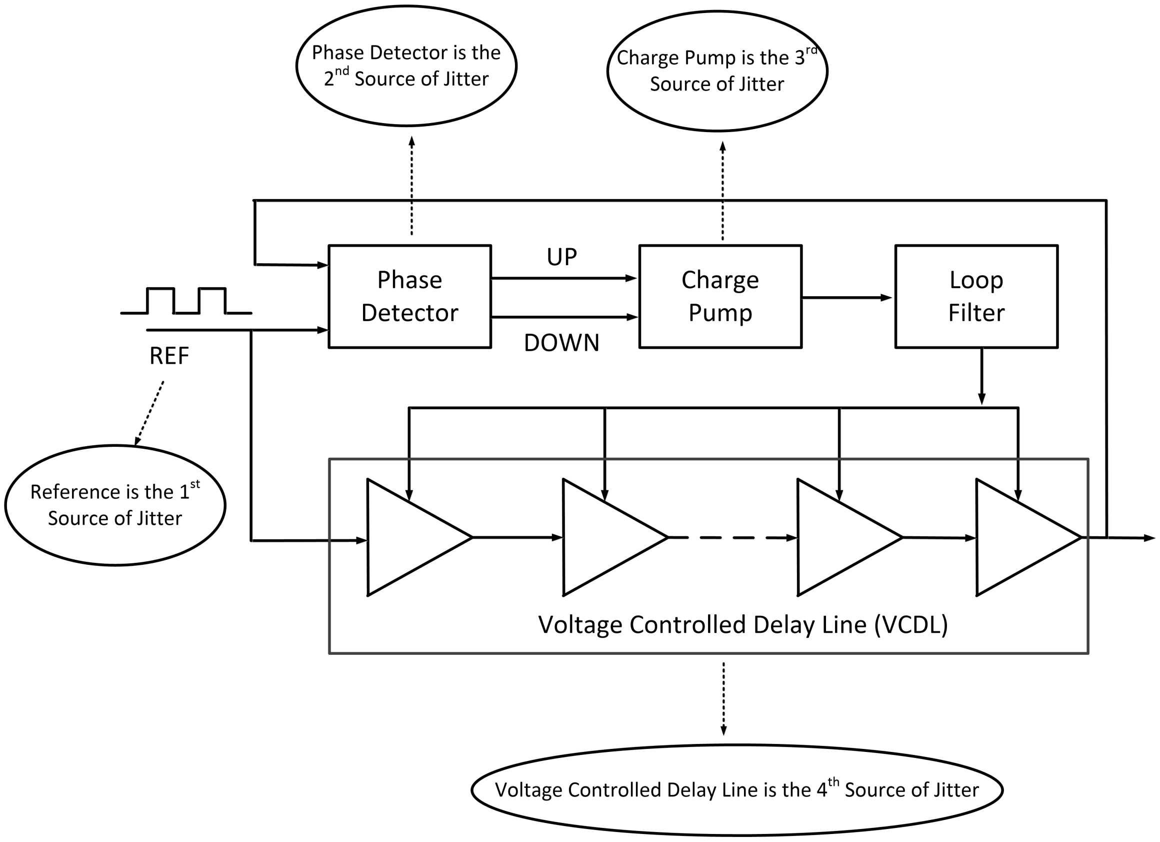

Phase Detector is the 2nd Source of Jitter
Charge Pump is the 3rd Source of Jitter
UP
DOWN
Phase Detector
Charge Pump
Loop Filter
REF
Reference is the 1st Source of Jitter
Voltage Controlled Delay Line (VCDL)
Voltage Controlled Delay Line is the 4th Source of Jitter

Figure 5.2: Jitter Source of Traditional Delay-Locked Loop [92]

delay cell in the voltage controlled delay line (VCDL), and from [92], the charge pump current and capacitance in the loop filter will affect the jitter at the output of VCDL.

The design of digital Delay-Locked Loop is necessary, because there is no requirement for a charge pump, the output of DLL will not be affected by this big jitter source. The phase detector and voltage controlled delay line also need to be designed carefully to fulfill the low jitter requirement.

5.2 Low Jitter Delay-Locked Loop Implementation

Although there are many analog and digital DLLs presented in previous work, [93–104], they have not addressed the random jitter problem and they cannot achieve very low random jitter, and therefore they are not suitable for the time interleaved ADC applications. All these previous works can only achieve several pico-second jitter performance. This order-of-magnitude jitter can degrade the ENOB of the ADC dramatically. Fig 5.3 shows the FFT of an ADC with a 3.6GHz ADC sampling clock with and without 0.1 psrms jitter, it can be seen that the jitter of ADC sampling clock degrades the SNR of the ADC. Fig 5.4 shows the ENOB of the ADC for different levels of jitter when the input signal change from 10MHz to 1.8GHz. The sampling clock frequency is 3.6GHz. It can be seen that if the 30th stage output of DLL has 0.1 psrms jitter, the ENOB will degrade from 12 to 11.2 when the input signal frequency increases to the Nyquist rate, if the 30th stage output of DLL has 0.5 psrms, the ENOB of ADC will degrade from 12 to 9 when the input signal frequency approaches the Nyquist rate. The several pico-second jitter performance will destroy the ENOB of ADC.

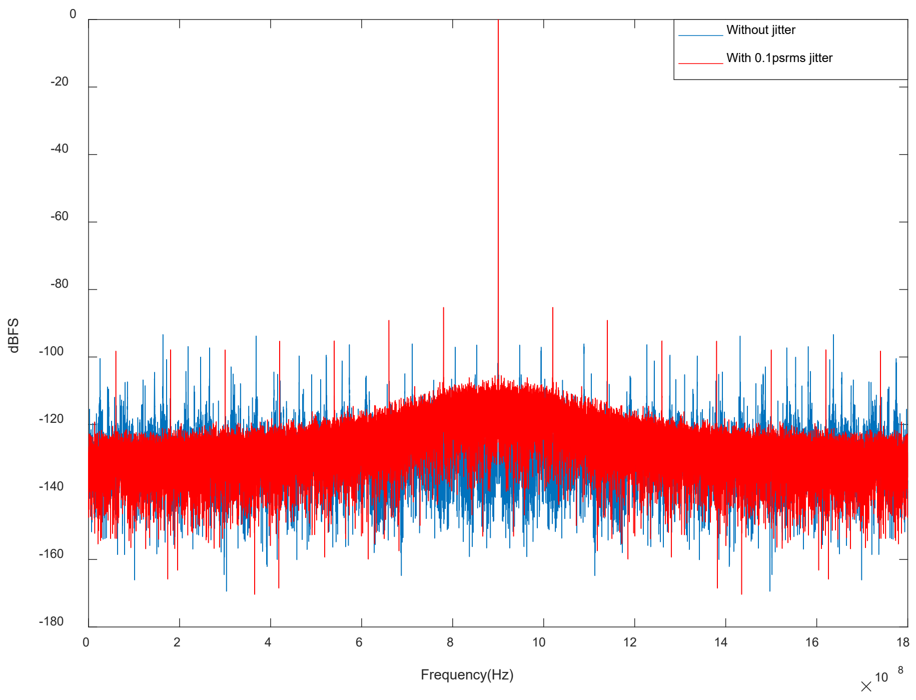


Figure 5.3: FFT of ADC output with/without 0.1 psrms jitter

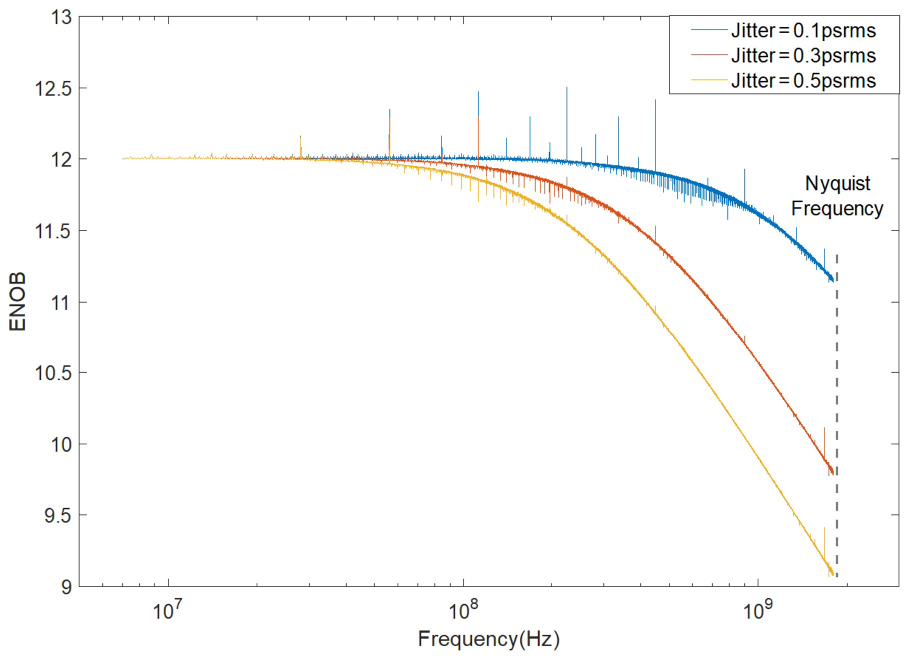


Figure 5.4: ENOB vs Input Signal Frequency, fs = 3.6GHz

5.2.1 Ultra Low System Time Offset Bang-Bang phase detector

Although the DLL is a good candidate for the low-jitter clock application, its jitter performance still cannot fulfill the requirement of current high-precision signal processing circuits. To solve the random jitter problem in the clock phases, a low jitter design technique has been derived [87] and used in the DLL [26] to realize the low-cost, low-complexity generation of a sub-picosecond-jitter sample clock suitable for time interleaved ADCs. However, few works published to date address issues related to the time offset (deterministic jitter) which is caused by the all-digital DLL's Bang-Bang phase detector.

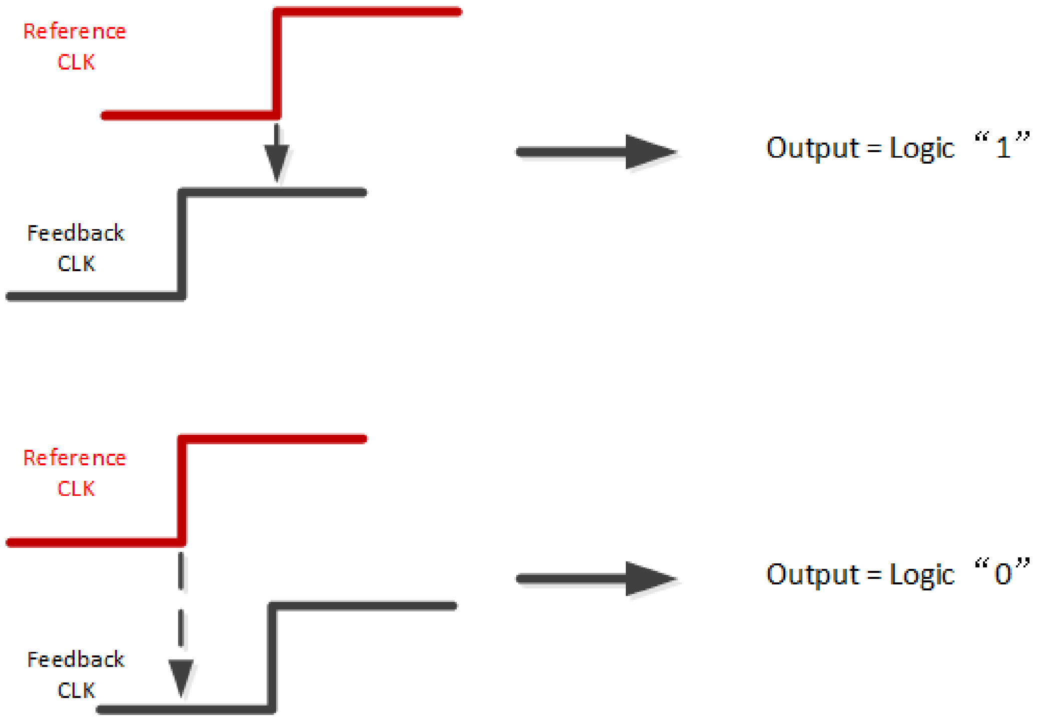


Figure 5.5: Output of Bang-Bang Phase Detector in different cases

Fig 5.5 shows how the BBPD works when the reference clock leads or lags the feedback clock. When the reference clock phase leads the feedback clock, at the rising edge of the reference clock, the value of the feedback clock is logic "1", so the output of BBPD is "1". When the reference lock phase lags the feedback clock, at the rising edge of the reference clock, the value of the feedback clock is logic "0", the output of BBPD will be "0".

The easiest way to design a BBPD for an all digital DLL is a D Flip-Flop based BBPD [25]. Because the digital DLL only needs the digital logic "1" or "0" to determine the phase difference, D Flip-Flop can provide enough information to the digital circuit when the reference clock and the delay-line feedback clock connect to the clock input pin and data input pin respectively. In this way digital circuit can determine if the delay-line feedback clock is either leading or lagging the reference clock. However, the setup time and hold time of the D Flip-Flip is the main limitation for digital DLL low-jitter clock generation circuits. Setup time, which is around 10ps in 180nm CMOS process, can be a big problem for the low jitter clock generation circuits. The reference clock and delay-line feedback clock have a 10ps time offset, which can be a 10ps deterministic jitter for the clock phases. The systematic offset in the DLL phase delays can lead to spurs in the FFT degrading SNR. According to (1.2), the SNR impact of a sample clock jitter σ_J with a full-scale sinusoidal input at frequency f_{SIG}. When these clock phases are provided to Time-Interleaved ADC, the Signal-to-Noise Ratio (SNR) performance can be degraded dramatically, which can affect ADC's Effective Number of Bits (ENOB), ENOB is determined from the SNR by equation (1.1).

In order to eliminate this systematic time offset, some phase frequency detectors are proposed [89]. Two D Flip-Flops are used to detect the clock phases. Although the output of the two DFFs are digital signal, they need to connect a charge pump to cancel the time offset in analog domain. However, this method is only suitable for the analog DLL. A Phase Detector design which is suitable for ultra low jitter

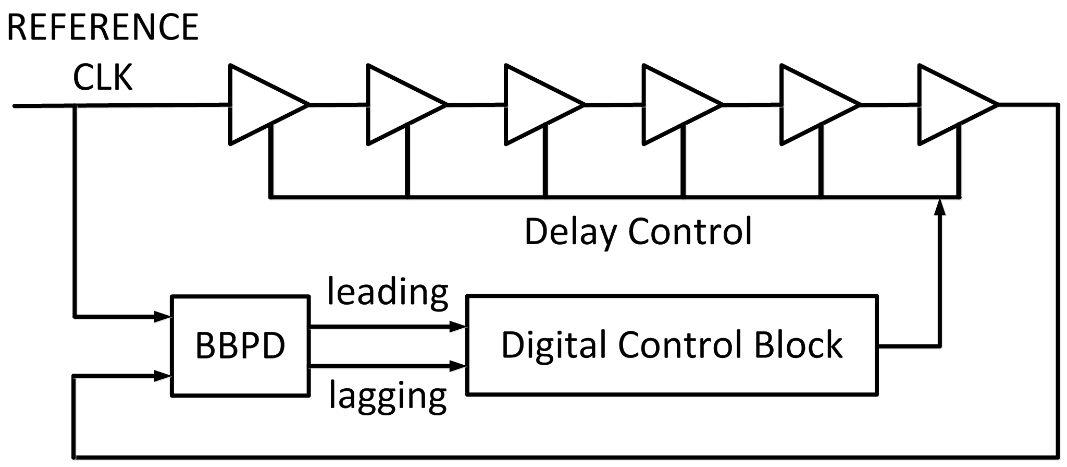


Figure 5.6: System Block Diagram for BBPD application.

all digital DLLs is more attractive and valuable.

Figure 5.6 shows the system block diagram for the BBPD application in an all digital DLL. The two inputs of BBPD connect to the reference clock and the delay line final output. The output of BBPD can provide the phase information about the reference clock and delay line feedback clock. From this phase information, the digital control block can determine that if the reference clock is leading or lagging the the delay line feedback clock. According to the digital logic from BBPD output, digital control block can tune the delay of the digital VCDL. This feedback drives the phase difference between the delay line and reference clocks closer. When the digital system detects that the two BBPD input clock phase are close enough, the digital system will stop changing the digital control code for the delay line. The detail of the method to determine if the difference between two BBPD input clocks is negligible will be discussed in Section 5.2.2.

Fig. 5.7 shows the schematic of the proposed low offset Bang-Bang phase detector. The DFF cannot be used for collecting two input clock signal in the ultra low time offset BBPD because the D Flip-Flop (DFF) has the setup time and hold time, which can be a huge offset value for the low jitter clock generation circuit design.

In the proposed BBPD, an SR latch is used for the phase detector input. The SR latch is symmetric under ideal conditions, therefore the output of SR latch Q and $\overline{Q}$ in Fig 5.7 can provide clock phase information between the two inputs precisely. Fig 5.8 shows the timing diagram of the proposed BBPD, the initial two outputs of BBPD are set to be "1". If reference clock phase leads the feedback clock, the SR latch output Q and $\overline{Q}$ will be "0" and "1" respectively. When SR Latch inputs are both "1", an AND gate output A will be "1" too. In order to collect the Q and $\overline{Q}$ information safely, a delay cell is used to make sure that voltages of Q and $\overline{Q}$ are stable at the rising edge of delayed signal B. Two DFFs are used to record the SR latch ouput information. The delayed signal B connects to both Clock pin of the DFFs. The delay cell should provide enough delay which can make sure that DFFs can record correct data at the output. The minimum value of this delay is given by equation (5.1)

$$T_{delay} > T_{SR} + T_{SETUP} - T_{AND} \tag{5.1}$$

Where T_{SR} is the propagation delay of SR Latch, T_{SETUP} is the DFF's setup time and T_{AND} is the AND gate's propagation delay. The two outputs of proposed phase detector can provide either leading or lagging information to the digital control block when the delay of delay cell is long enough.

Fig. 5.9 shows the traditional NAND gate, two PMOSs are connected in parallel and two NMOSs are connected in series. The problem of this NAND is the asymmetry on the two inputs IN1 and IN2. The NMOSs are connected in series, which means the propagation delay for IN1 and IN2 are different on the NMOS side. For example, if IN1 is "1" and IN2 changes from "0" to "1", the discharge current needs to go through N1, which can delay the changing of output. If IN2 is "1" and IN1 changes from "0" to "1", because voltage at point A is already 0 volts, the NAND

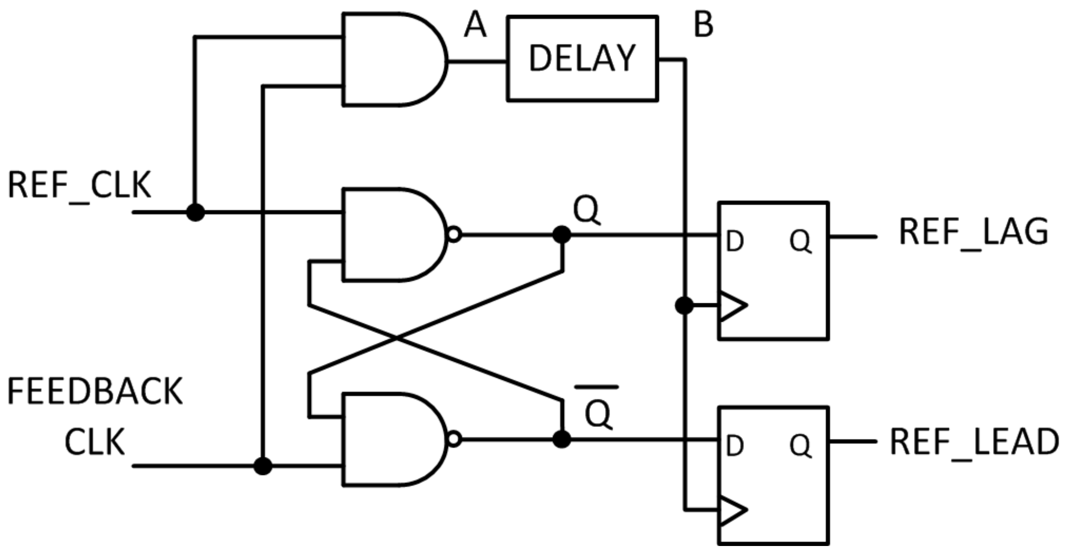


Figure 5.7: Proposed Ultra Low Offset BBPD

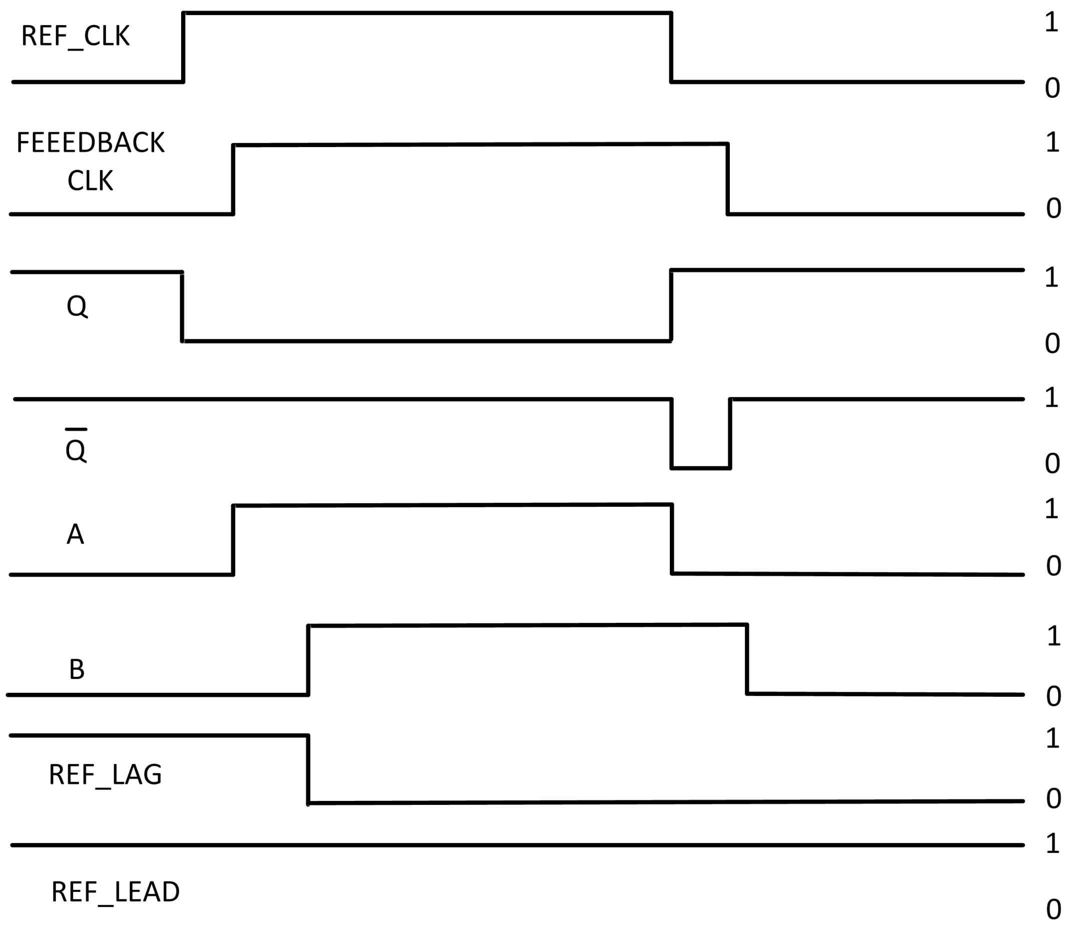


Figure 5.8: Timing Diagram of Proposed BBPD

gate discharges faster and the propagation delay is smaller. This asymmetry on the two inputs can lead to approximately 0.1 ps time offset in 180 nm CMOS process, this time offset is already comparable to the target jitter performance in ultra low jitter clock generation circuits, so this traditional NAND gate is not suitable for the low jitter application in proposed BBPD.

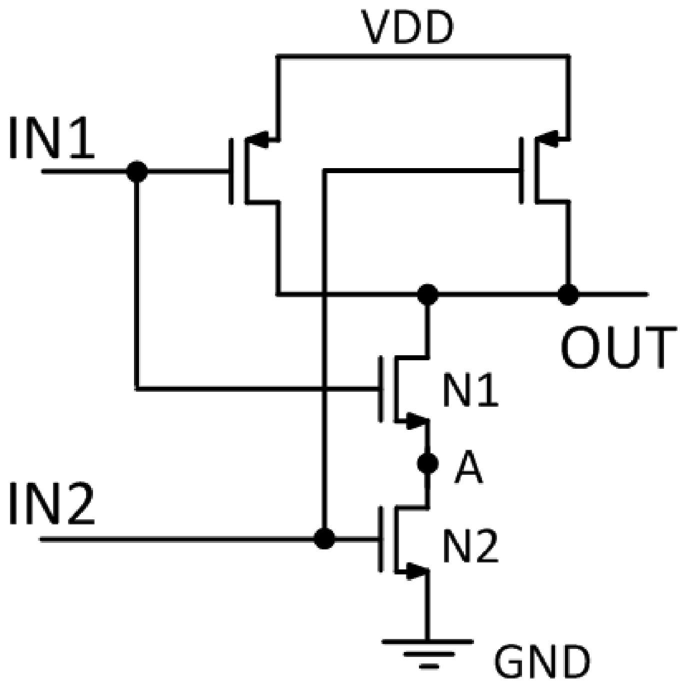


Figure 5.9: Schematic of traditional NAND Gate

Fig. 5.10 shows the symmetric NAND gate which can provide two inputs IN1 and IN2 the same input to output propagation delay. In this symmetric NAND gate, the number of transistors is doubled. Input signal IN1 controls N1 and N4, IN2 controls N2 and N3, which are fully symmetric. Since there is no asymmetric problem in the new NAND gate, the discharging time of IN1 going high or IN2 going high are the same. This method can decrease the time offset between two inputs to zero in ideal conditions.

Another consideration for this SR latch based BBPD is the layout parasitic

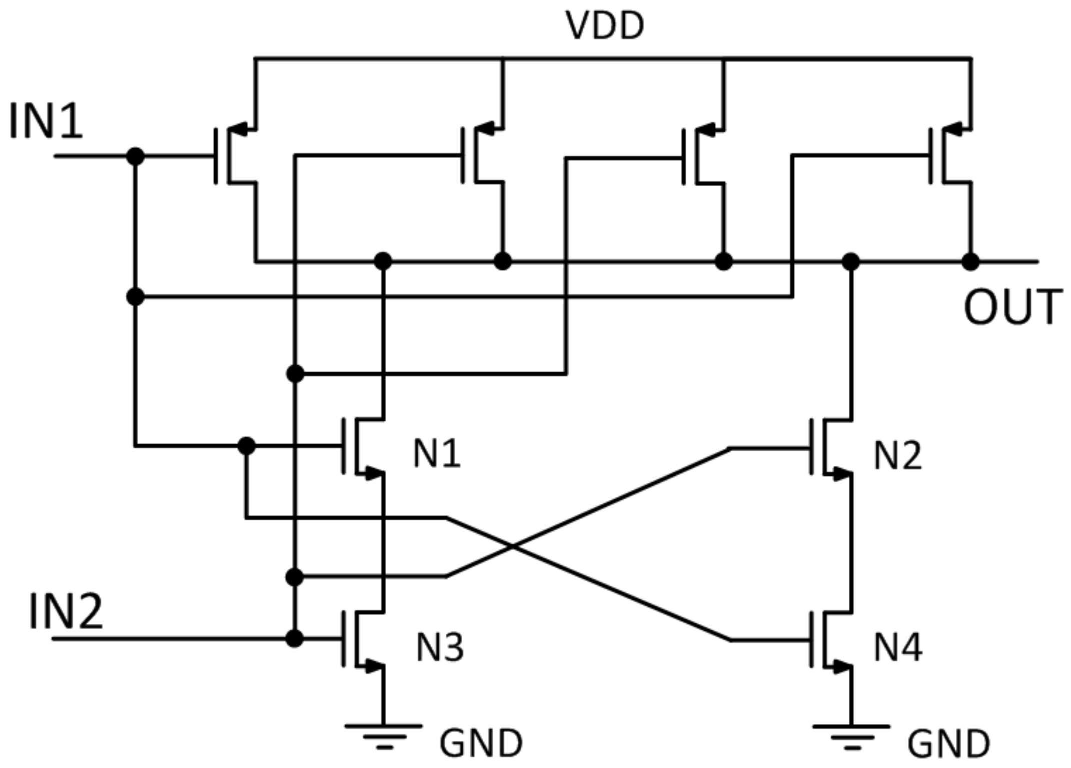


Figure 5.10: Schematic of Porposed Symmetric NAND Gate

mismatch between point Q and $\overline{Q}$ in Fig 5.7. The typical capacitive load at Q in 180nm CMOS is about 0.1fF. If there is capacitive load mismatch in the layout, that can cause mismatch in the propagation delay, which can be a problem for the low jitter application. Table 5.2 shows the simulation results when point Q keeps the typical capacitive value and point $\overline{Q}$ has a capacitive mismatch within 50% . From Table 5.2, capacitive mismatch must be smaller than 20% to achieve the 0.1 psrms jitter performance. This layout mismatch problem can be avoided by proper analog layout techniques.

Fig. 5.11 shows the Monte Carlo simulation of proposed BBPD time offset, the standard deviation of the time offset is 0.3 ps. Comparing to traditional DFF based BBPD whose time offset is dominated by DFF setup/hold time, the proposed BBPD time offset does not have this problem. Its time offset mean value is zero.

Table 5.2: Time Offset Due To Capacitive Load Mismatch at SR Latch Output

Cap Value at Q (fF)	Cap Value at $\overline{Q}$ (fF)	Time Offset (ps)
0.1	0.05	0.271
0.1	0.06	0.217
0.1	0.07	0.163
0.1	0.08	0.108
0.1	0.09	0.054
0.1	0.1	0
0.1	0.11	-0.054
0.1	0.12	-0.108
0.1	0.13	-0.162
0.1	0.14	-0.215
0.1	0.15	-0.269

This ultra low time offset can help interleaved ADC decrease lookup table size in the digital correction block, and save ADC chip area, power consumption and design complexity.

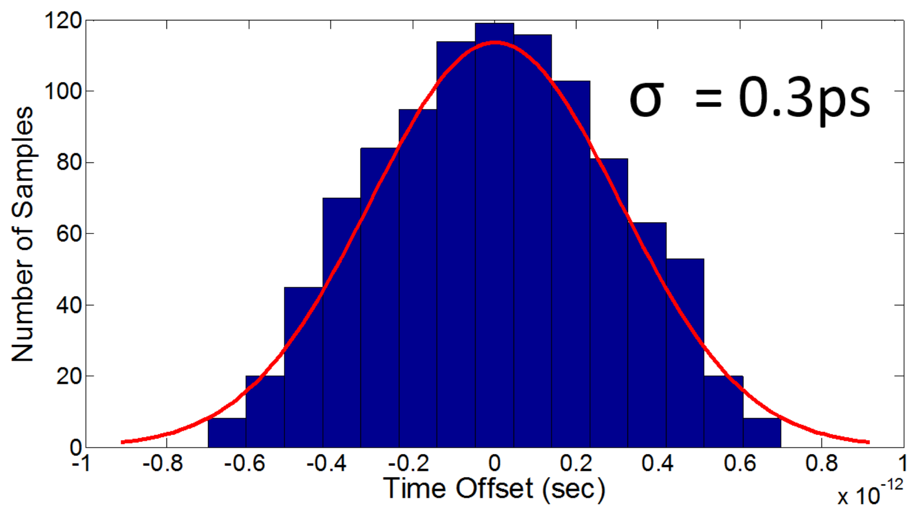


Figure 5.11: Monte Carlo Simulation for Proposed Bang-Bang Phase Detector

Fig. 5.12 shows simulated results for the proposed BBPD when the reference clock leads the feedback clock 0.1 ps. In this case the voltage at Q point is “0” at

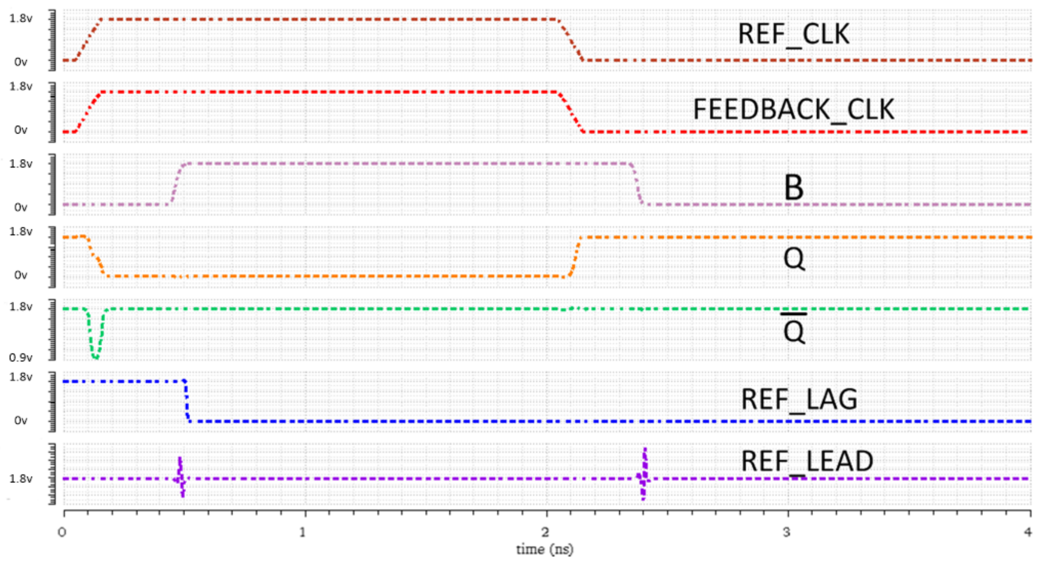


Figure 5.12: Simulation Result When Reference CLock Leads 0.1ps

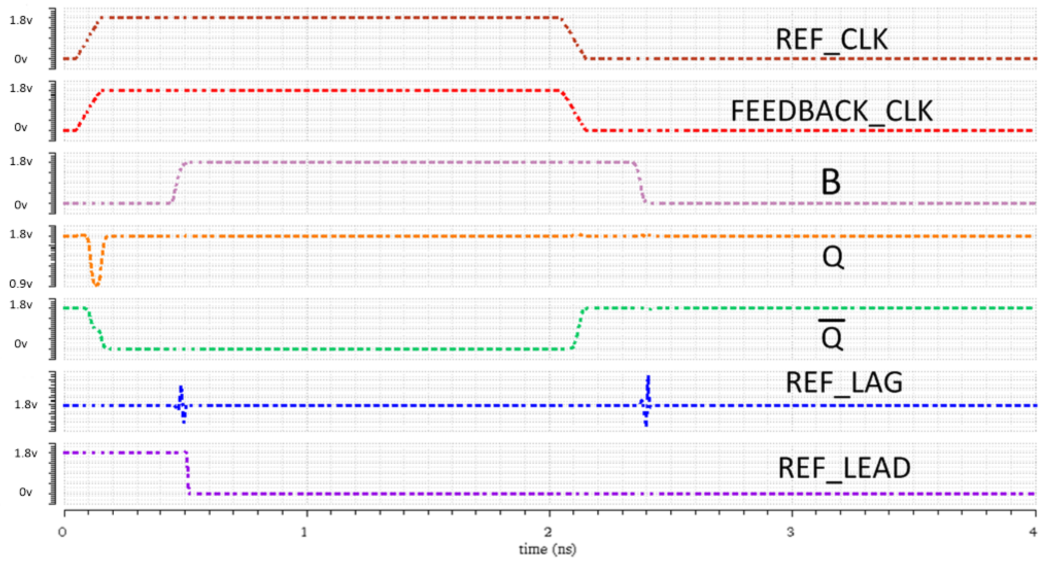


Figure 5.13: Simulation Result When Reference Clock Lags 0.1ps

the rising edge of signal B, and $\overline{Q}$ is "1" at this time. Signal B connects to the clock input pin of DFFs. The BBPD output pin *REF_LEAD* will be set to be "1" in this case, the digital control block will receive the reference leading information and decrease the delay in delay line. Fig. 5.13 shows simulated results for the proposed BBPD when reference clock lags the feedback clock 0.1 ps. In this case voltage at point Q is “1” at the rising edge of signal B, and $\overline{Q}$ is “0”. BBPD outout port REF_LAG will be set to be “1” and the digital control block can increase the delay in delay line.

Simulations in Fig. 5.12 and Fig. 5.13 shows that the proposed BBPD has better than 0.1 ps phase difference detection resolution when the two NAND gates in SR Latch are well matched. The digital control block in an all digital DLL needs digital logic as the input to determine the phase lagging or leading, the proposed BBPD has two outputs which can provide the leading or lagging information, and these two logic signal is always diffenent after the rising edge of signal “B”. The proposed BBPD has a phase difference detection resolution that is better than 0.1 ps. The setup/hold time of the DFF is no longer a limit of the phase difference detection.

5.2.2 Probability

The application of the proposed low jitter Delay-Locked Loop is to solve the sub-picosecond jitter problem, therefore the step resolution of the low jitter delay line should be able to ensure that the distance from the lock position to reference clock is smaller than the target jitter performance. If the final distance value in DLL lock condition is bigger than the random jitter, the deterministic jitter will ruin the clock jitter and the ENOB of the time interleaved ADC will degrade.

Due to the very small delay line step resolution, the rising edge of the reference clock will be very close to the rising edge of delay line feedback clock when the Delay-Locked loop is very close to the lock condition. Ideally, when the reference clock perfectly matches the feedback clock, we consider that this is the lock condition and the control circuit can stop changing the delay in the delay line. However, due to the existence of the jitter in the feedback clock, the lock condition cannot be determined from only one time detection. For example, it is possible that the feedback clock is

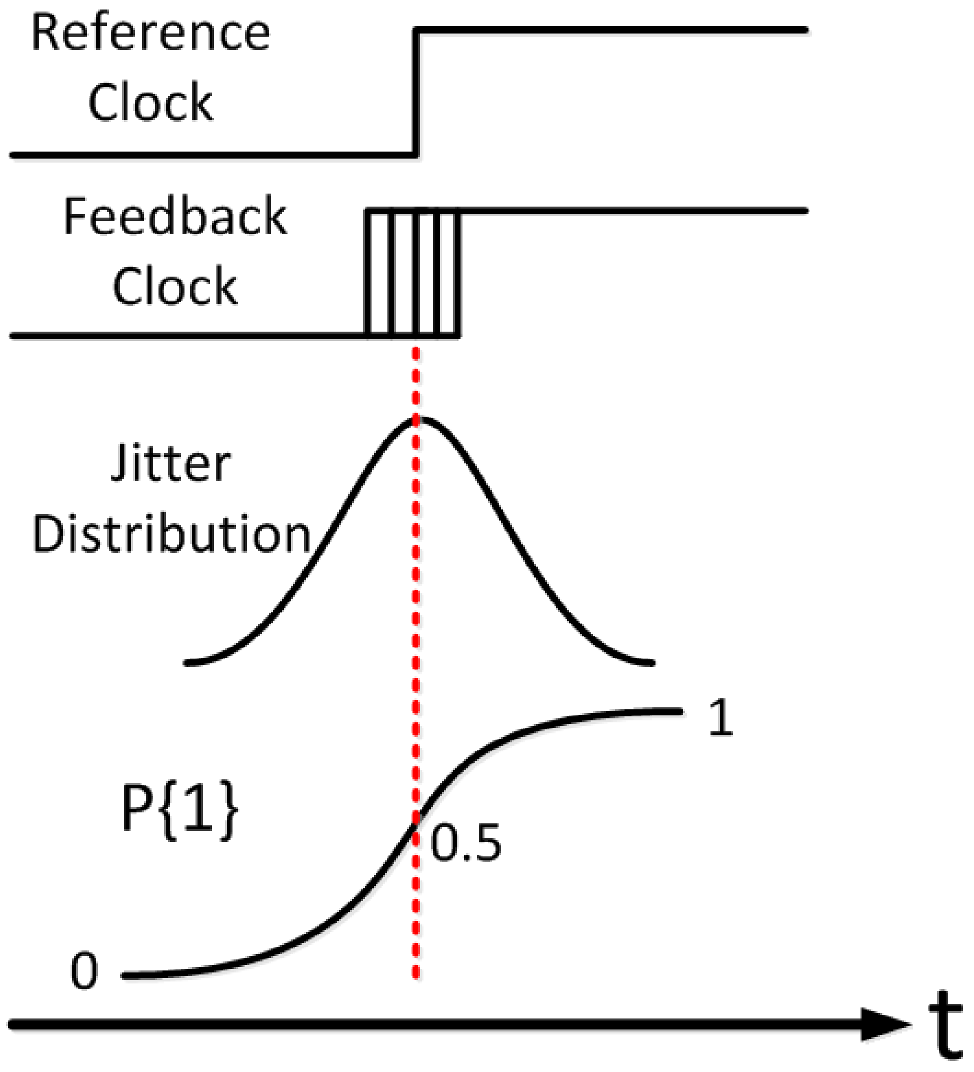


Figure 5.14: Probability Problem in Bang-Bang Phase Detection with Random Jitter Clock

already perfectly matched with the reference, however, the phase detector output shows that the feedback clock phase lags the reference clock phase due to the jitter. Although the Bang-Bang phase detector output can be either 1 or 0, and because the Gaussian Distribution with zero norm value is symmetric, the probability of detecting 1 should approach 50 percent as we detect more times when the feedback clock phase perfectly match the reference clock phase.

Rather than enforce exact equality between 0 and 1 at the Bang-Bang phase detector output, a probabilistic lock condition for the probability of 1 at the DLL output P{1} is defined to prevent "hunting" of the DLL loop and the associated increased jitter. Assuming a Gaussian distribution, a range of "reasonable" probability can be determined each time the Fine Control block changes the control code. The Fine Controller will collect a large number of samples of the BBPD output to estimate $\hat{P}\{1\}$. If $\hat{P}\{1\}$ falls in the acceptable region, then the fine control block decides that the reference clock and output clock of last delay stage are effectively locked. After each control code change, the fine control block will collect the data from BBPD again and analyze the data to for next time the fine control code changes.

A delay step resolution of proposed delay line of 0.2 ps is good enough, because the target random jitter is 0.1 psrms. Fig. 5.15 shows results of a MATLAB simulation of $\hat{P}\{1\}$ for a sample size of 64. The reference clock is chosen randomly, the horizontal axis shows the steps of changing the control number, and the vertical axis shows $\hat{P}\{1\}$ at the edge of last delay stage output. From Fig. 5.16 it shows that, with 0.2ps step resolution, a range of 15% to 85% range can be defined so DLL will be locked with 0.1 psrms jitter.

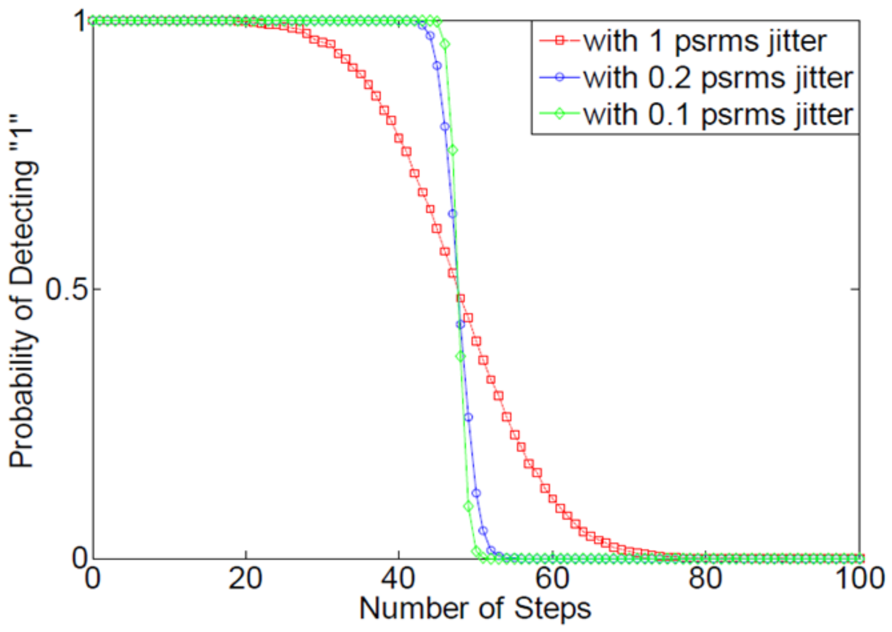


Figure 5.15: Probability of detecting “1” with different value of jitter

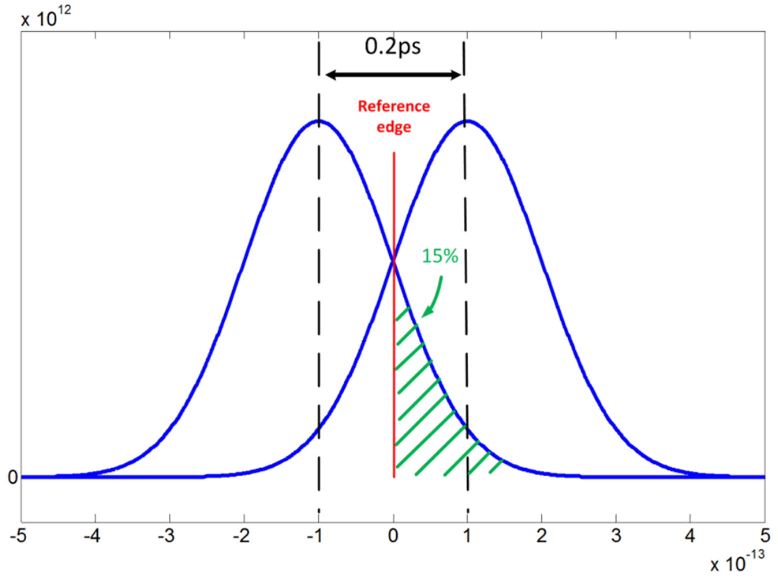


Figure 5.16: With 0.2ps step resolution, $15\% < p\{1\} < 85\%$ can be considered as the DLL lock condition

5.2.3 Digital System

Figure 5.17 shows the digital circuit system for the low jitter Delay-Locked loop. In the digital system, the low offset Bang-Bang Phase Detector is designed using

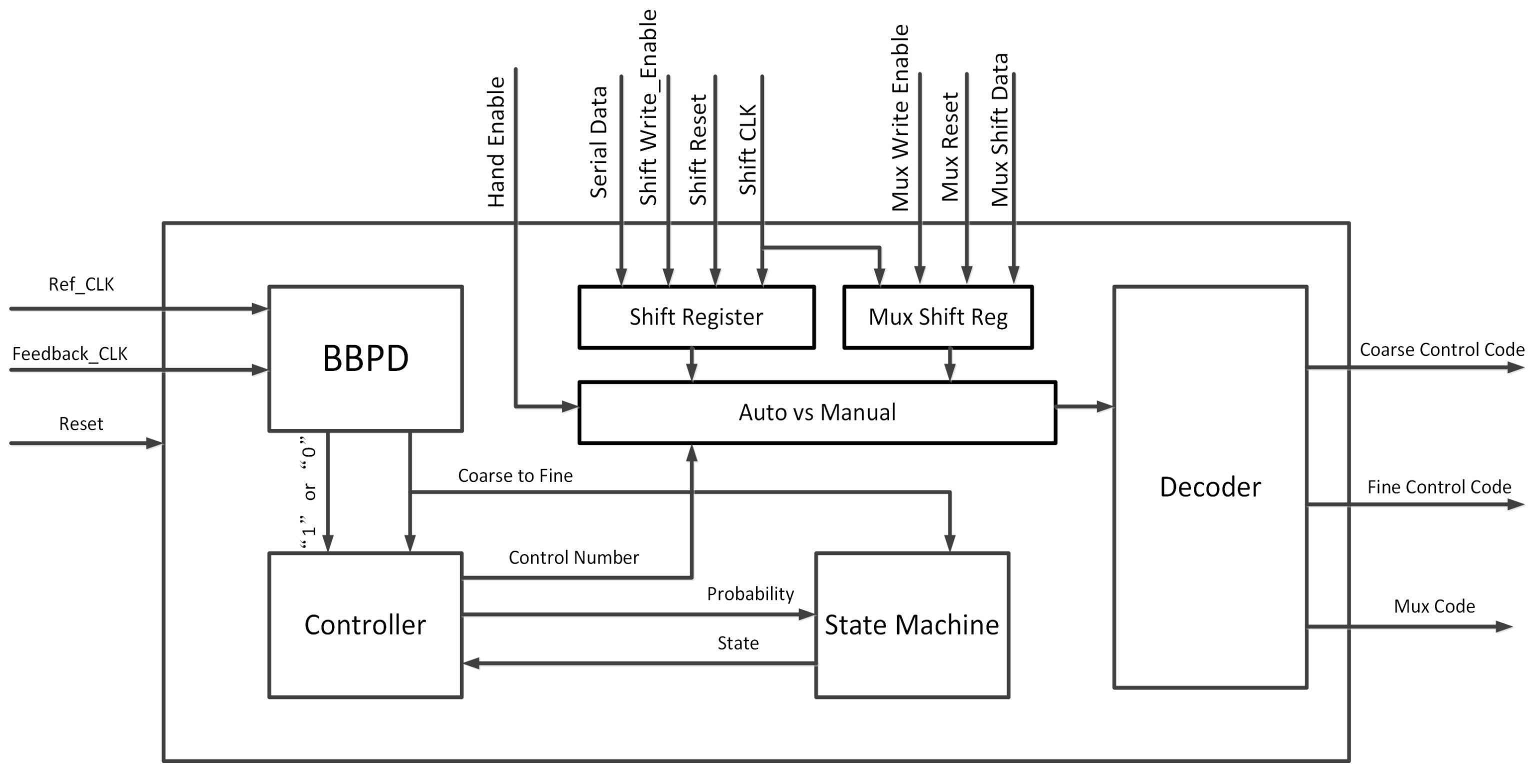


Figure 5.17: The Implementation of Digital System

analog technique to alleviate the Bang-Bang Phase Detector two inputs' offset, the detailed design is discussed in Section 5.2.1. Table 5.3 shows the pin description of the digital system. In Figure 5.17, the input reference clock and DLL feedback clock connect to the Bang-Bang phase detector, which determines if the reference clock is leading or lagging to the DLL feedback clock. Then the Bang-Bang phase detector will provide "1" or "0" to the controller. The controller will receive the data from Bang-Bang phase detector and decide when the digital system should jump from the coarse control mode to the fine control mode. In the controller, there is also a probability calculation block. This block will calculate the probability of "1" in the total data that Bang-Bang phase detector sent to controller. This probability value will be sent to state machine, which will decide if the digital system should continue changing the delay line control code or stop. According to Section 5.2.2, when the probability of "1" is between 15 percent and 85 percent, the digital circuit will stop changing the delay and the system will assume that the reference clock is matching the DLL feedback clock. Although the digital control code for the delay line will stop changing, the probability block in the controller continues calculating the probability and sending this data to the state machine. When the probability of "1" is lower than 5 percent or higher than 95 percent, the digital system will assume that the reference clock is not matching the DLL output clock anymore and the digital system will go back to work changing the delay line control code.

There is an auto/manual block which is designed for the test purposes. The auto mode is used for the DLL output to lock the reference clock automatically, and this mode is the default mode. The manual mode is used for testing. This chip is designed to demonstrate the jitter theory which is presented is Section 2.3, therefore we need a way to manually control the delay line's digital control code and achieve the I_{PEAK} on purpose. There is a shift register to receive the manual

Table 5.3: Input/Output of Low Jitter Delay-Locked Loop Digital System

Pin Name	Input /Output	Pin Description
Ref_CLK	Input	Reference Clock for DLL
Feedback_CLK	Input	DLL feedback Clock
Shift CLK	Input	Clock for Digital Circuit Shift Register
Reset	Input	Reset Signal for Digital Circuit
Serial Data	Input	Digital Shift Register Serial Input Data
Mux Shift Data	Input	Shift Register Serial Input Data for Analog Multiplexer
Shift Write Enable	Input	Shift Register Write Enable
Shift Reset	Input	Shift Register Reset
Mux Write Enable	Input	Analog Mux Input Data Shift Register Write Enable
Mux Reset	Input	Analog Mux Input Data Shift Register Reset
Hand Enable	Input	Automatic/Manual Mode Switch
Coarse Control Code	Output	Control Code for Delay Line Corase Control
Fine Control Code	Output	Control Code for Delay Line Fine Contro
Mux Code	Output	Control Code for Analog Multiplexer

mode control code. The digital data is shifted into the chip in series because there is limited number of bond pad. After the shift write enable signal is high, the data will be shifted in.

There is also a mux shift register, which is used to received the mux control code. This block is also designed for the test purposes. Due to the limited number of bond pads and the need to measure the jitter of each delay line stage's output, an analog mux was designed, and this mux shift register is used to control the analog mux.

A decoder is necessary because the all digital code is shifted into the chip in series and this data is binary number. For example, because the coarse control code in delay line is a thermometer number, there is a binary number to thermometer number decoder. The analog mux control code is one-hot number, there is a binary number to one-hot number decoder.

5.2.4 Low Jitter Voltage Controlled Delay Line

Figure 5.18 shows the schematic of the low jitter delay line in proposed Delay-Locked Loop. The proposed delay line has two modes: coarse tune mode and fine tune mode. In the coarse control mode switches, the size (W/L) of the NMOS transistor is 2um/180nm. According to the jitter theory in Chapter 2.4, in order to achieve the 0.1 psrms jitter target with 8.333 ns propagation delay (120MHz), the peak current should be around 3 mA. However, this low jitter Delay-Locked Loop can work from 80MHz to 120MHz, this Delay-Locked Loop can still achieve very low jitter performance during the lower working frequencies. Therefore the I_{PEAK} should be able to be bigger than 3 mA. To explore the space of this test chip, more transistors are used to exceed the design requirement. The designed circuit in Figure 5.18 can achieve a maximum current of 5.2 mA. The coarse control code uses thermometer code to ensure the monotonic change of the propagation delay. There are 13 bits in the fine control switches, the minimum size (W/L) of the NMOS transistor in fine control switch is 220nm/19.2um. The reason of designing a such long transistor is to ensure the step resolution of the delay line is smaller than 0.1ps, then the deterministic jitter can be alleviate. The digital code for fine control switches is binary number.

Figure 5.19 shows the simulation result of the proposed low jitter Delay-Locked Loop. The clock cycle of reference clock is 8.333 ns, which is 120MHz clock. The 30th stage output of the delay line is locked to the reference clock.

5.2.5 Test Circuit

Figure 5.20 shows the test circuit for low jitter delay line. To ensure that we can measure the jitter of the delay line output successfully, two methods are realized on

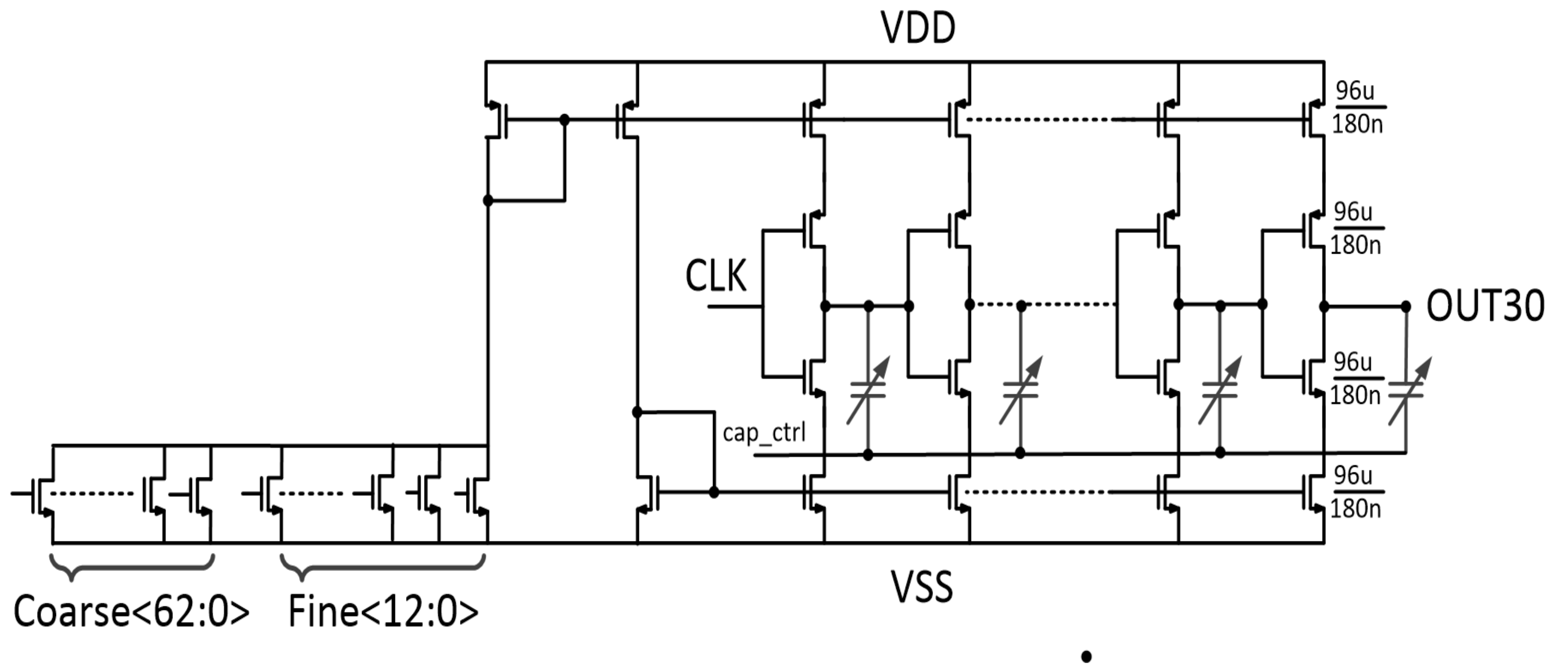


Figure 5.18: Schematic of Low Jitter Delay Line

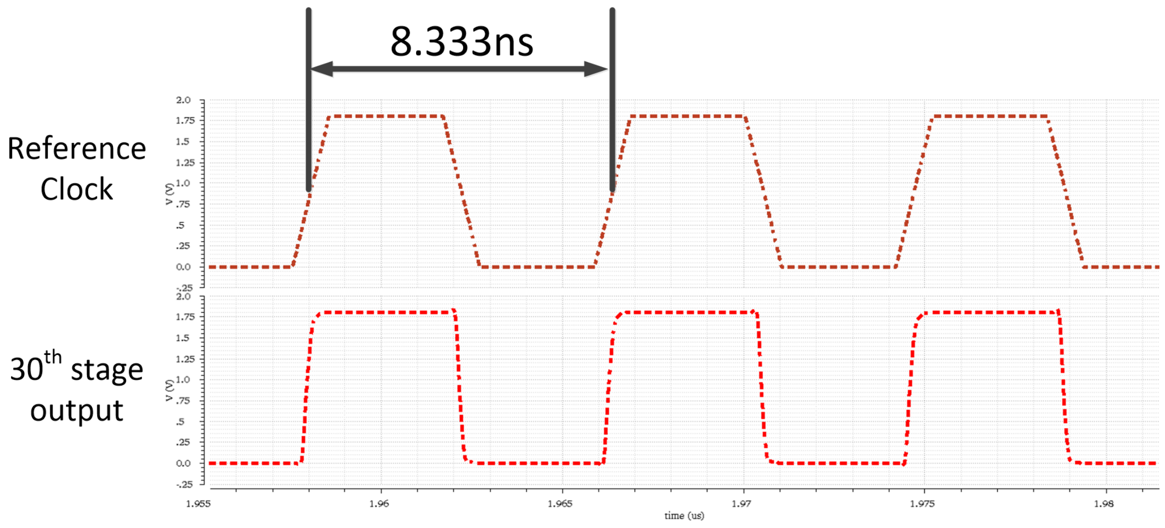


Figure 5.19: Simulation of Low Jitter Delay-Locked Loop

this test chip. After each stage of the delay line there are two output buffers, one buffer is used to drive the analog test circuit and the other is use to drive the digital signal test circuit. In each delay stage, there are two current starved inverters, however, in Figure 5.20 it can be seen that there are two buffers even this inverter output is not going to be tested. The main reason of designing in this way, even though only one output will be measured, is the matching consideration. At the output of each inverter in the delay line, the output load should be the same, so there is a dummy stage in the delay line as well as testing buffers on every stage.

On the upper half in Figure 5.20 is the analog signal test circuit. In order to model the effect of clock jitter to ADC performance, the output buffer's output will connect to a differential sample and hold circuit. The buffer's output serves as the sample clock of the sample and hold circuit. This sample and hold circuit also has two differential sinusoidal signal inputs. When the sample clock is high , the sample and hold circuit will generate two hold voltages. These voltages will be chosen by the analog multiplexer, and then connect to the input of the difference amplifier, where we get the difference of these two voltages. The purpose of using a differential sample and hold circuit is to alleviate the noise from the power supply or circuit.

On the lower half in Figure 5.20 is the digital signal test circuit. Since an oscilloscope is used to measure the clock directly in this method, we only need to use a analog mux to choose the output of the buffers.

In the low jitter Delay-Locked Loop design, because the delay line output needs to connect to the output buffer, analog multiplexer and ESD circuit, we cannot measure just the jitter from the delay line. However, the input reference clock can also be connected to an output buffer. In Figure 5.20 it can be seen that the input clock can also be measure through the mux selection, thus the jitter from output buffer, mux and ESD circuit can be measured directly, allowing the jitter of the delay line to be calculated.

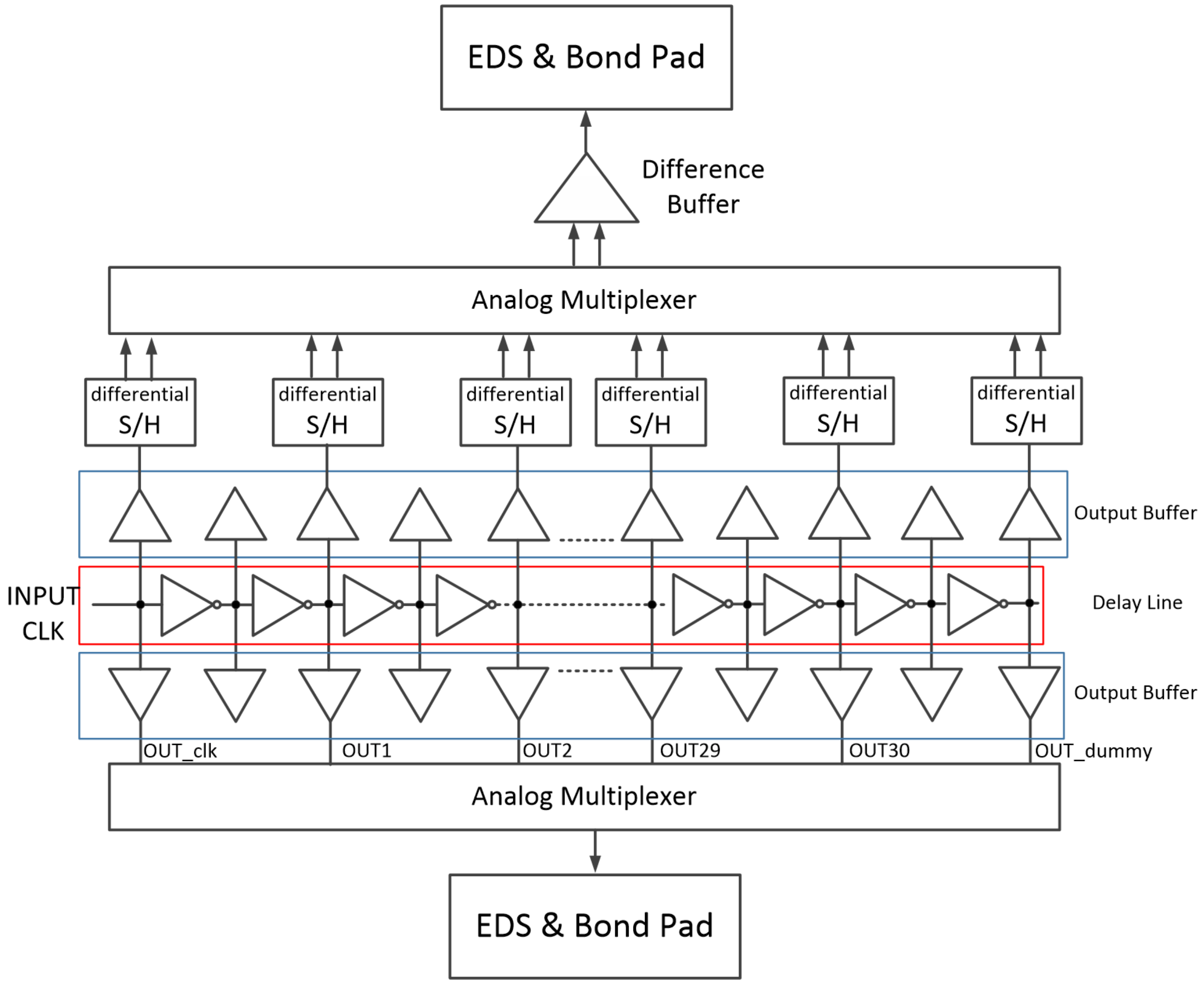


Figure 5.20: Block Diagram of Delay Line Test Circuit

Chapter 6

Low Jitter Delay-Locked Loop Testing and Analysis

6.1 Evaluation Board design and Layout

Fig 6.1 shows the PCB layout of the evaluation board. On the board, there are several different low dropout voltage regulators (LDOs) for different power supplies. For example, because we want to measure the power of the delay line, there is a 1.8v regulator for the low jitter delay line. There are also regulators for the crystal oscillator and voltage controlled crystal oscillator for testing the chip. On the left bottom corner in Figure 6.1, there is a GPIO connector which is used to receive the FPGA control signal. For the corresponding FPGA code, please refer to Appendix A.3. Fig 6.2 shows the photo of PCB which is used to evaluate proposed low jitter Delay-Lock Loop test chip.

Fig 6.3 shows the schematic of the LDO. The sub-picosecond jitter performance is very sensitive and could be destroyed by the noise from the power supply or the noise from other circuits. The power supply of the proposed Delay-Locked Loop

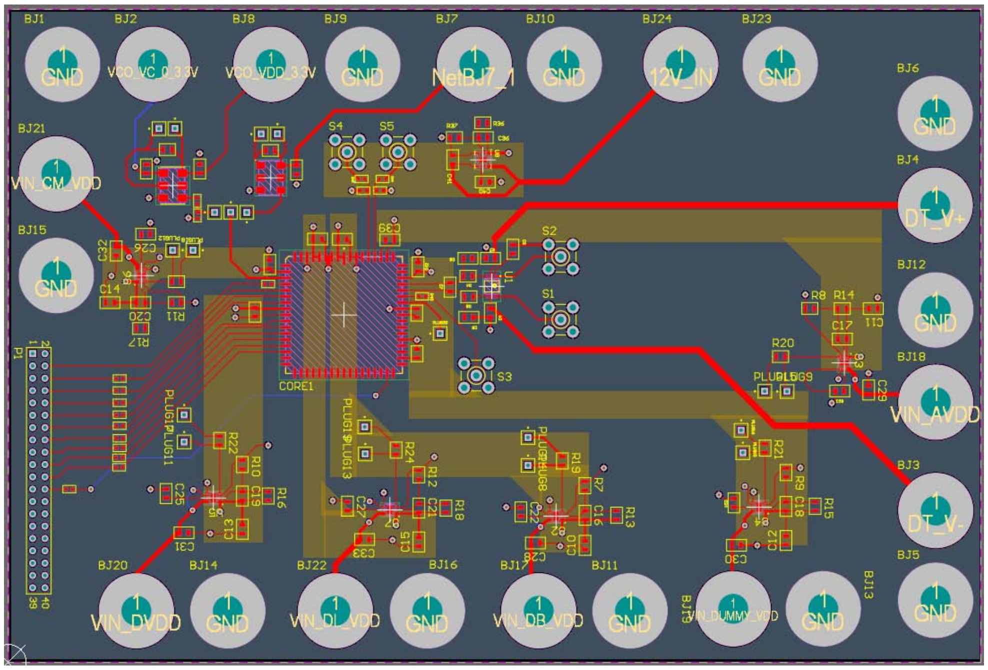


Figure 6.1: Layout of the Printed Circuit Board

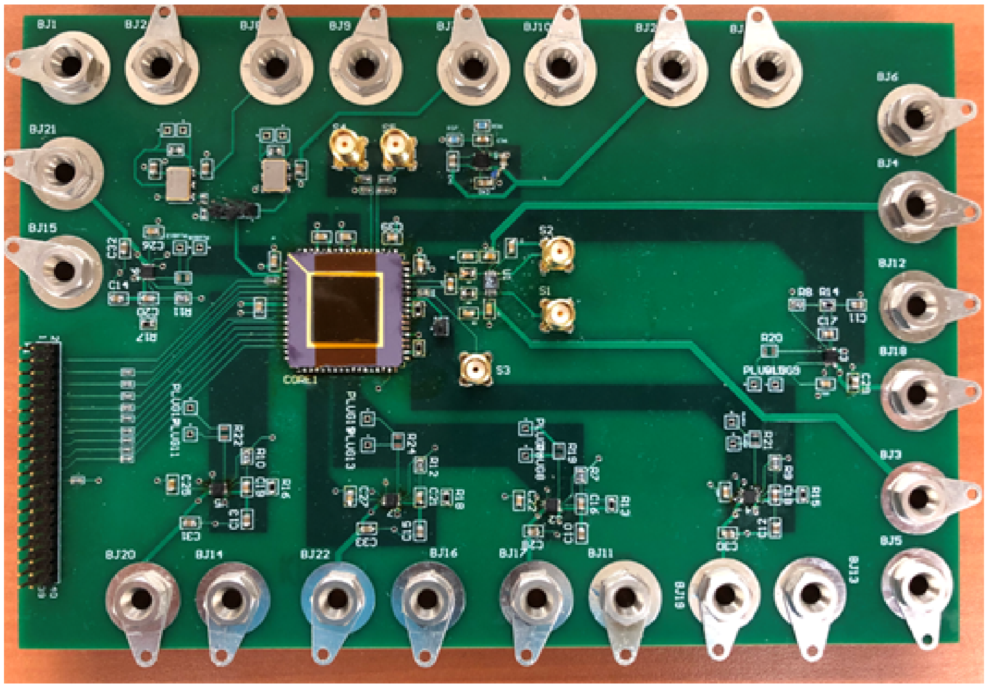


Figure 6.2: Photo of the Printed Circuit Board

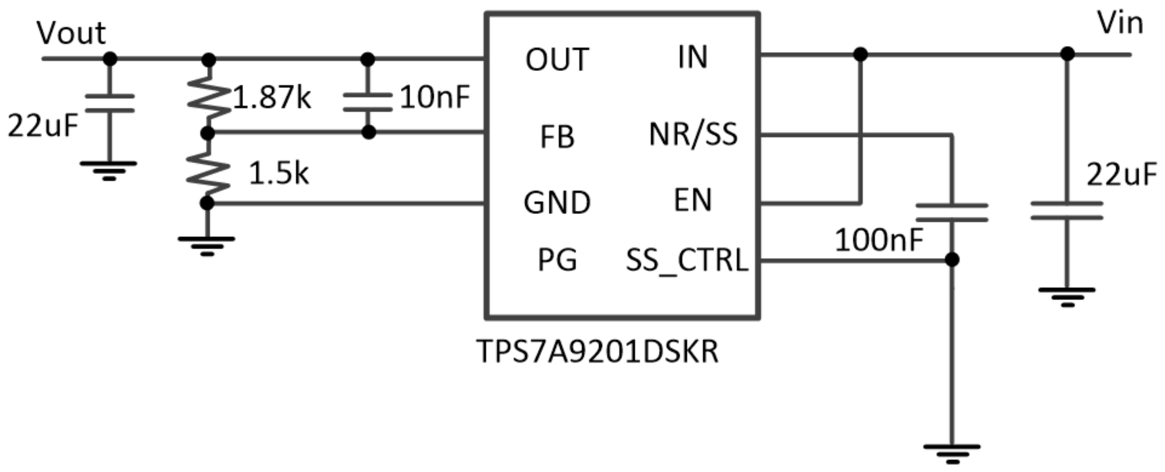


Figure 6.3: Schematic of the Low Dropout Voltage Regulator

must be a very stable and quiet power supply on the printed circuit board. A Texas Instrument's LDO is used to provide the supply voltage for the chip. This schematic is the recommended application circuit in the data sheet.

Because there is another project on this test chip, the FPGA should also provide some signals to turn other circuits off and alleviate the crosstalk between the signals.

6.2 Measurement Result - Jitter

The proposed low jitter Delay-Locked Loop was fabricated in 0.18um CMOS process, Fig 6.4 shows the layout view of the low jitter Delay-Locked Loop in Cadence Virtuoso layout editor. The size of low jitter delay line and test circuit is 1250um x 200um, and the size of digital logic is 400um x 200um. The die photo is shown in Fig 6.5, size of the whole chip is 2mm x 2mm.

Figure 6.6 shows the measured jitter of low jitter Delay-Locked Loop input reference clock. The reference clock jitter is 4.36321 psrms. From Section 5.2.5, we can know that this jitter is not just the input clock jitter, but the sum of jitter from output buffer, analog multiplexer and ESD circuits. Figure 6.7 to Figure 6.9

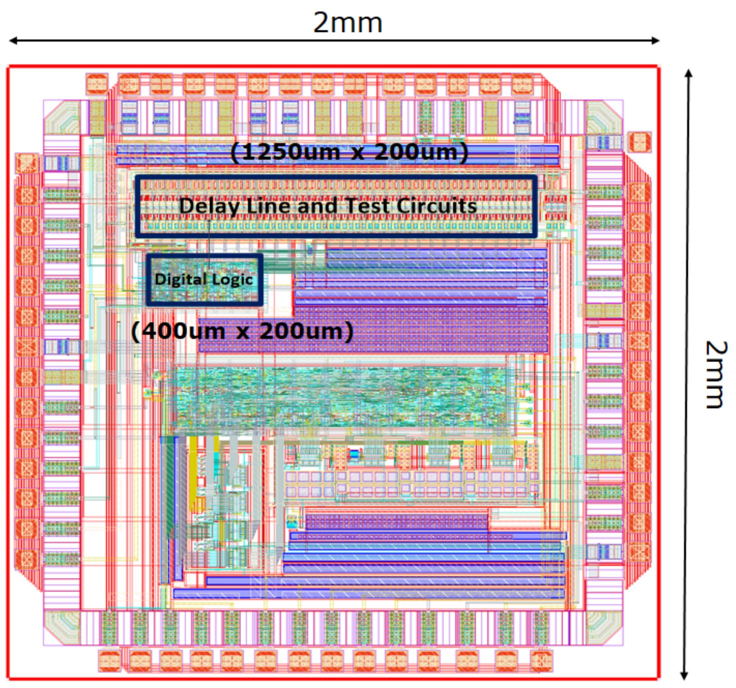


Figure 6.4: Chip Layout of Proposed DLL

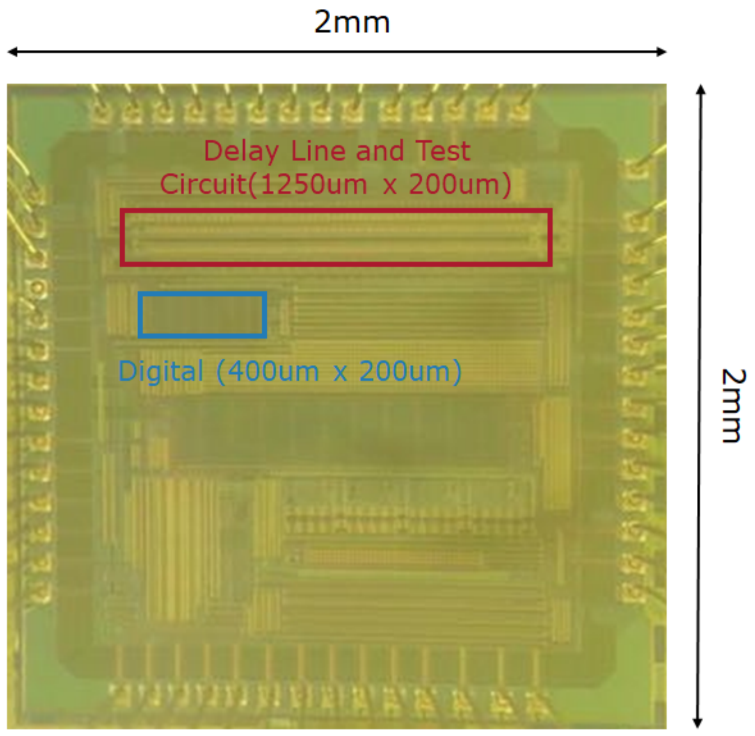
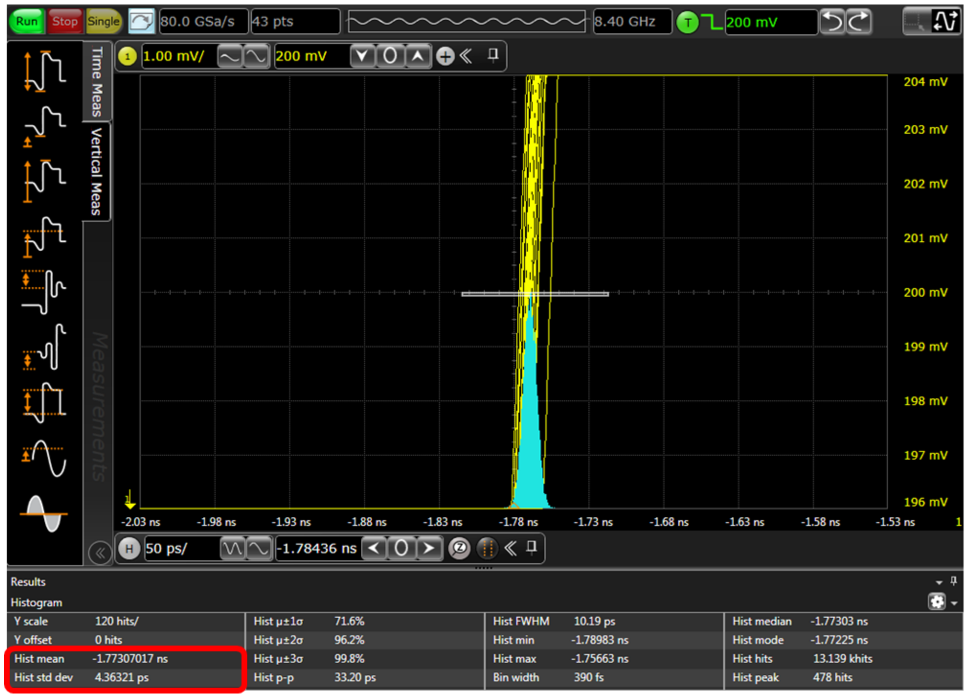


Figure 6.5: Chip Micrograph of Proposed DLL



Results
Histogram

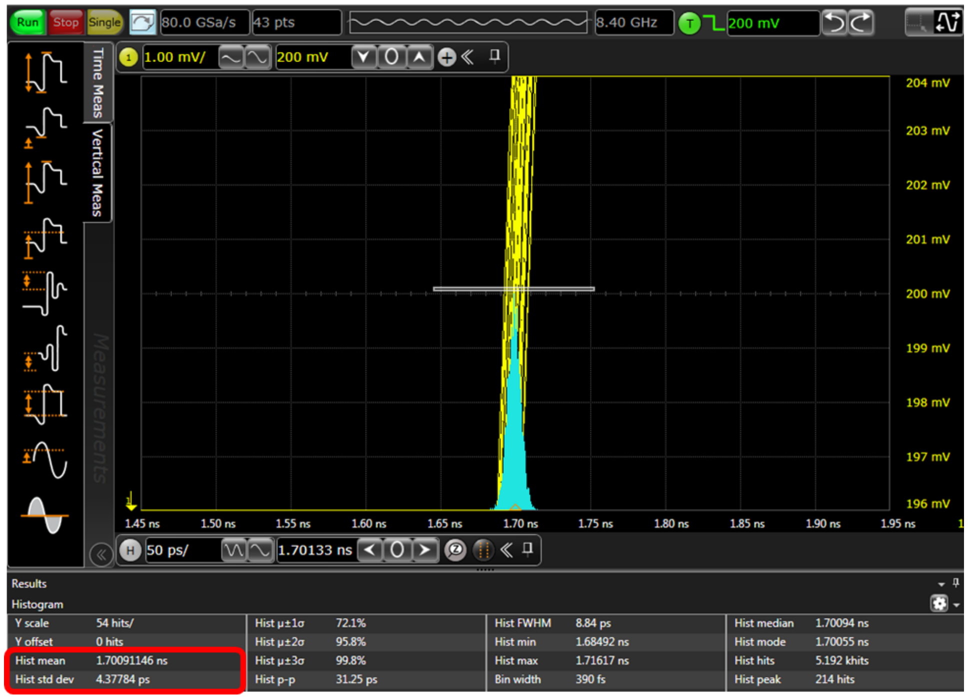
Y scale	120 hits/	Hist μ±1σ	71.6%	Hist FWHM	10.19 ps	Hist median	-1.77303 ns
Y offset	0 hits	Hist μ±2σ	96.2%	Hist min	-1.78983 ns	Hist mode	-1.77225 ns
Hist mean	-1.77307017 ns	Hist μ±3σ	99.8%	Hist max	-1.75663 ns	Hist hits	13.139 khits
Hist std dev	4.36321 ps	Hist p-p	33.20 ps	Bin width	390 fs	Hist peak	478 hits

Figure 6.6: Jitter of Input Reference Clock Source: 4.36321 psrms

show the low jitter Delay-Locked Loop 10th to 30th stage output jitter respectively. Moreover, from the "Hist mean" value from Figure 6.6 to Figure 6.9, it can be seen that the propagation delay from input reference clock to 30th stage output clock is 10ns, which proves that the DLL is working due to the 100MHz input clock in during test. From equation (3.5), the jitter of single delay stage is :

$$\sigma_{stage} = \sqrt{\frac{1}{30}\left(\underbrace{4.40350ps^2}_{\sigma_{out30}} - \underbrace{4.36321ps^2}_{\sigma_{source}}\right)} = 0.1085psrms \qquad (6.1)$$

Comparing to Section 4.3, the jitter performance of each delay stage is almost the same, however, the propagation delay of each delay stage in the Delay-Locked Loop is 5.5 times longer (333ps vs 60ps), which means the jitter performance has improved 2.4 times. The main difference between the delay line in Chapter 3 and Chapter 5 is delay line architecture. In Chapter 3, the delay line is designed in

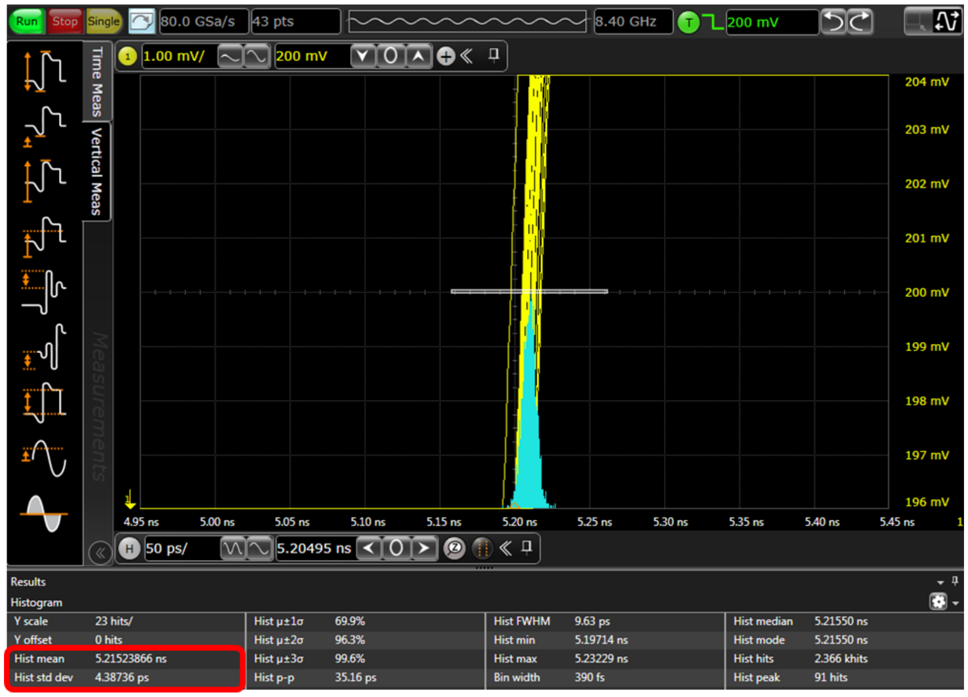


Results

Histogram

Y scale	54 hits/	Hist μ±1σ	72.1%	Hist FWHM	8.84 ps	Hist median	1.70094 ns
Y offset	0 hits	Hist μ±2σ	95.8%	Hist min	1.68492 ns	Hist mode	1.70055 ns
Hist mean	1.70091146 ns	Hist μ±3σ	99.8%	Hist max	1.71617 ns	Hist hits	5.192 khits
Hist std dev	4.37784 ps	Hist p-p	31.25 ps	Bin width	390 fs	Hist peak	214 hits

Figure 6.7: Jitter of Low Jitter Delay-Locked Loop 10th Output: 4.37784 psrms

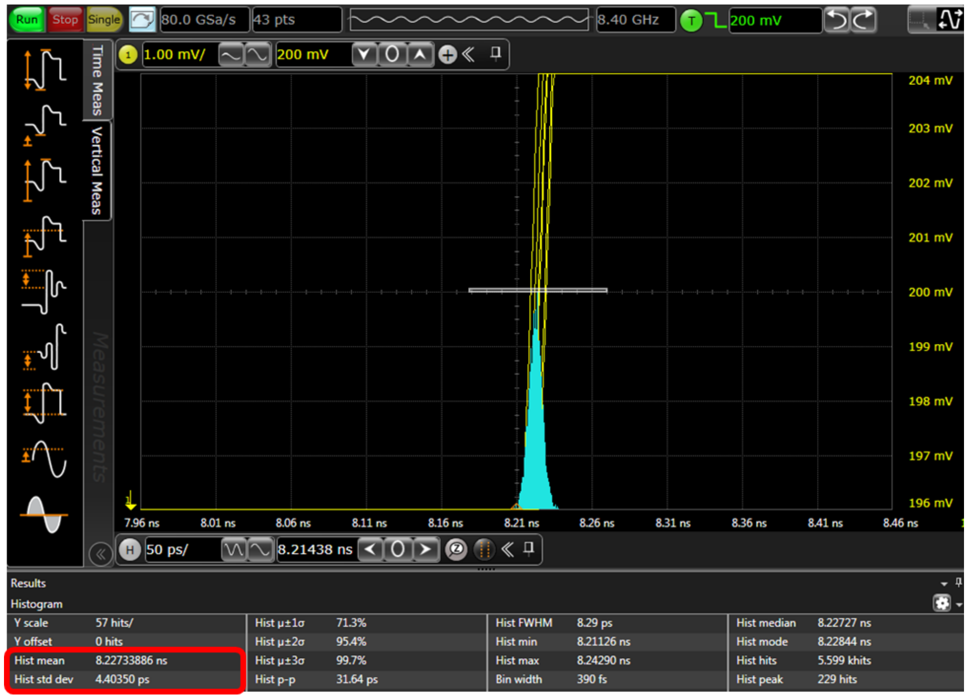


Results

Histogram

Y scale	23 hits/	Hist μ±1σ	69.9%	Hist FWHM	9.63 ps	Hist median	5.21550 ns
Y offset	0 hits	Hist μ±2σ	96.3%	Hist min	5.19714 ns	Hist mode	5.21550 ns
Hist mean	5.21523866 ns	Hist μ±3σ	99.6%	Hist max	5.23229 ns	Hist hits	2.366 khits
Hist std dev	4.38736 ps	Hist p-p	35.16 ps	Bin width	390 fs	Hist peak	91 hits

Figure 6.8: Jitter of Low Jitter Delay-Locked Loop 20th Output: 4.38736 psrms



Results

Histogram

Y scale	57 hits/	Hist μ±1σ	71.3%	Hist FWHM	8.29 ps	Hist median	8.22727 ns
Y offset	0 hits	Hist μ±2σ	95.4%	Hist min	8.21126 ns	Hist mode	8.22844 ns
Hist mean	8.22733886 ns	Hist μ±3σ	99.7%	Hist max	8.24290 ns	Hist hits	5.599 khits
Hist std dev	4.40350 ps	Hist p-p	31.64 ps	Bin width	390 fs	Hist peak	229 hits

Figure 6.9: Jitter of Low Jitter Delay-Locked Loop 30th Output: 4.40350 psrms

pseudo-differential way, although this kind of architecture can decrease the delay between each delay stage, there is a very small cross-coupled inverter between two current starved inverters in each delay stage. According to the jitter theory in Section 2.3 and 2.4, it is known that the peak current in this small inverter is very low and this leads to jitter injection in the delay line.

6.3 Measured Results - Jitter vs Propagation Delay

According to the jitter theory in Sections 2.3 and 2.4, the jitter of a inverter is:

$$\sigma_t = \kappa\sqrt{T_d} \tag{6.2}$$

where the κ is defined as:

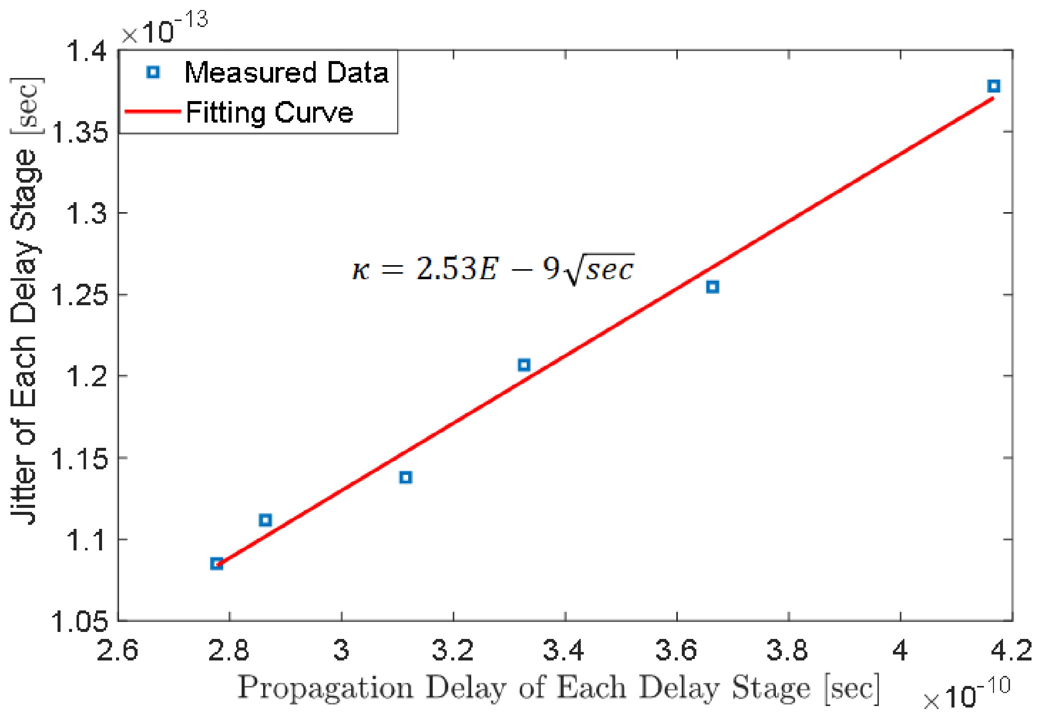


Figure 6.10: Measured Jitter When Propagation Delay Change, I_{PEAK}=4.2mA

$$\kappa = 2\sqrt{\frac{kT}{I_{PEAK}V_{DD}}} \tag{6.3}$$

From equation (6.2) and equation (6.3), if the I_{PEAK} is constant, when the propagation delay increases, the jitter should increase. In the delay line design, there is a varactor at the output of each delay stage. During the manual mode, the output of the Delay-Locked Loop will not lock to the reference clock. We can change the control voltage of the varactor to change the propagation delay of the delay line and keep the I_{PEAK} constant to observe this behavior.

Figure 6.10 shows the measured data when I_{PEAK} is 4.2 mA. From the figure it can be seen that the jitter increases when propagation delay increases. The red line in the figure is the fitting curve of the measured data. From the slope of the red fitting curve, we can get the κ, which is 2.53E-9 $\sqrt{sec}$. We can also calculate κ

using equation 6.3, and we get 2.12E-9 $\sqrt{sec}$. Comparing the measured data to the calculated data, the predicted theoretical value is 84% of measured data, which still follows the jitter theory in Section 2.3 and 2.4.

6.4 Measured Results - Jitter vs I_{PEAK}

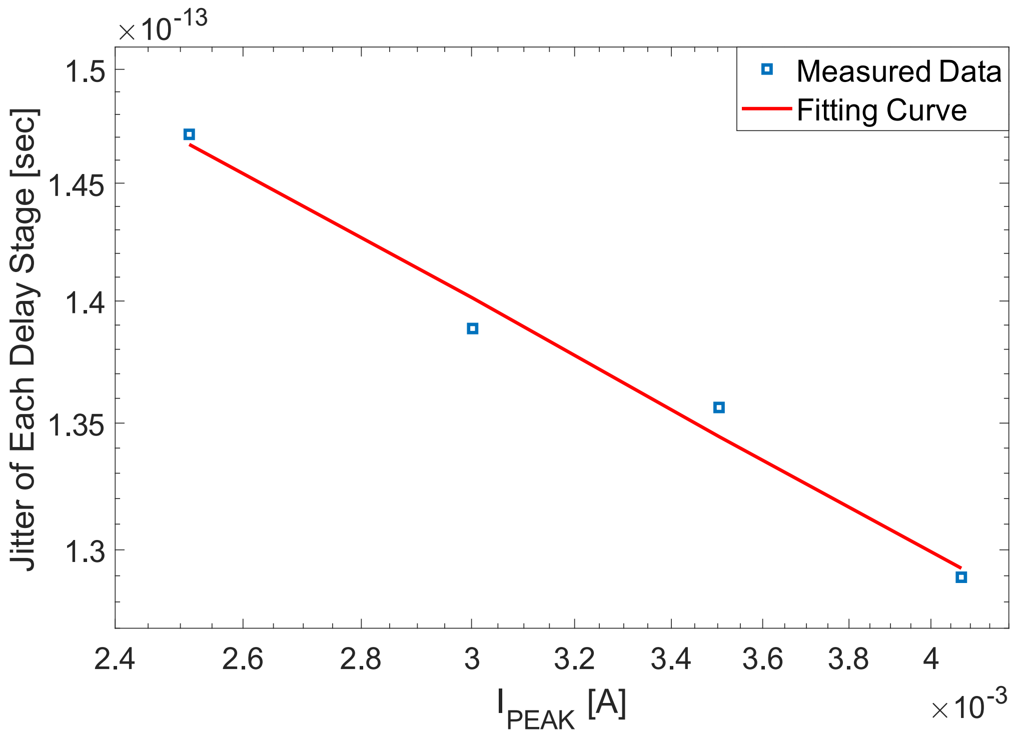


Figure 6.11: Measured Jitter when I_{PEAK} change. t_{PD} = 10ns

From the jitter theory in Sections 2.3 and 2.4, we know that jitter will decrease when I_{PEAK} increases, if the propagation delay is constant. During the manual control mode, if the control code of the delay changed, the I_{PEAK} and propagation delay will change. However, there is a varactor at the output of each delay stage, we can still control the varactor control voltage to change the delay and keep the delay as 10 ns.

Figure 6.11 shows the measured data when propagation delay is 10ns. From the figure it can be seen that the jitter decreases when I_{PEAK} increases. There are only 4 four points measured because the varactor can only change the propagation delay a limited amount. The red line in the figure is the fitting curve of the measured data. This curve also proves the jitter theory in [87] about the relationship between I_{PEAK} and jitter.

Chapter 7

Conclusion

A sub-picosecond jitter clock generation circuit which is suitable for time interleaved ADC application was presented. There are two test chips designed and taped out to demonstrate the low jitter theory of this low jitter clock generation circuit.

Chapter 1 described the motivation and application of the low jitter Delay-Locked Loop, the Delay-Locked Loop can generate sample clocks for time interleaved ADC for RF ADC application. Chapter 2 described the definition of jitter, how is jitter generated in the circuit and the jitter theory which is the basis of the designed circuit. According to the Chapter 1, time interleaved ADC requires low jitter sample clock, in order to investigate the low jitter theory, Chapter 3 presented the design details of the first test chip, where a low jitter delay line was designed in 180nm CMOS. Chapter 4 showed the measured results of the first test chip; the measured data followed the jitter theory and the measured result is within 50% of the calculated data. According to the measured results of first chip, a ultra low jitter DLL was designed in Chapter 5 to generate sample clocks for time interleaved ADC. This chapter explains the design detail including the digital system, ultra low offset Bang-Bang phase detector, the probability theory in DLL locking problem and the ultra low jitter delay line. This low jitter DLL was also taped out and the measurement result is presented in Chapter 6. Chapter 6 demonstrated the jitter theory not just through the low jitter value in delay line, but also the jitter vs propagation delay and jitter vs I_{PEAK} relationship.

7.1 Future Work

A low jitter Delay-Locked Loop with time interleaved ADC can be designed together in the future. Although the measured jitter is small, the output clock of the delay line still needs to deal with the power/ground bounce problem which is due to the big inductance of bonding wire. If the low jitter Delay-Locked Loop can be integrated with the time interleaved ADC, the output is digital code and the jitter can be determined through the ENOB of the ADC.

Appendix A

APPENDIX

A.1 Low Jitter Delay Line chip evaluation FPGA code

The following code is realized on Altera DE1 FPGA board.

```
module board1(clk,reset,start,d1,d2,d3,d4,d5,d6,d7,d8,d9,
         d10,d11,d12,d13,d14,d15,d16,d0,available,finish,
       serial_in,shift_clk,shift_enable,shift_reset,shiftA,
         VINP,OEBAR1,VINN,shiftABAR,Latch,RVDD,RVSS,
         IBIAS,IBIAS2,IBIAS3,IBIAS4,BIAS1,RVSSB,RVDDB,
         Latch2,shiftBBAR,DIR2,OEBAR2,VINNB,VINPB,shiftB,
         OEBAR3,DIR3,reset_DIGI,clk_DIGI,DIR1);

input wire       clk;
input wire       reset;
input wire       start;
input wire        d1;
input wire        d2;
input wire        d3;
input wire        d4;
input wire        d5;
input wire        d6;
input wire        d7;
input wire        d8;
input wire        d9;
input wire        d10;
input wire        d11;
input wire        d12;
input wire        d13;
input wire        d14;
input wire        d15;
input wire        d16;
input wire        d0;
output wire       available;
output wire       finish;
output wire       serial_in;
output wire       shift_clk;
output wire       shift_enable;
output wire       shift_reset;
output wire       shiftA;
output wire       VINP;
output wire       OEBAR1;
output wire     VINN;
output wire     shiftABAR;
output wire     Latch;
output wire     RVDD;
output wire     RVSS;
output wire     IBIAS;
output wire     IBIAS2;
output wire     IBIAS3;
output wire     IBIAS4;
output wire     BIAS1;
output wire     RVSSB;
output wire     RVDDB;
output wire     Latch2;
output wire     shiftBBAR;
output wire     DIR2;
output wire     OEBAR2;
output wire     VINNB;
output wire     VINPB;
output wire     shiftB;
output wire     OEBAR3;
output wire     DIR3;
output wire     reset_DIGI;
output wire     clk_DIGI;
output wire     DIR1;


wire    SYNTHESIZED_WIRE_36;
wire    SYNTHESIZED_WIRE_1;
wire    SYNTHESIZED_WIRE_2;
wire    SYNTHESIZED_WIRE_37;
wire    SYNTHESIZED_WIRE_38;
wire    SYNTHESIZED_WIRE_8;
wire    SYNTHESIZED_WIRE_9;
wire    [4:0]  SYNTHESIZED_WIRE_10;
wire    SYNTHESIZED_WIRE_11;
wire    SYNTHESIZED_WIRE_12;
wire    SYNTHESIZED_WIRE_13;
wire    SYNTHESIZED_WIRE_14;
wire    SYNTHESIZED_WIRE_15;
wire    SYNTHESIZED_WIRE_16;
wire    SYNTHESIZED_WIRE_17;
wire    SYNTHESIZED_WIRE_18;
wire    SYNTHESIZED_WIRE_19;
wire    SYNTHESIZED_WIRE_20;
wire    SYNTHESIZED_WIRE_21;
wire    SYNTHESIZED_WIRE_22;
wire    SYNTHESIZED_WIRE_23;
wire    SYNTHESIZED_WIRE_24;
wire    SYNTHESIZED_WIRE_25;
wire    SYNTHESIZED_WIRE_26;
wire    SYNTHESIZED_WIRE_27;
wire    SYNTHESIZED_WIRE_28;
wire    SYNTHESIZED_WIRE_29;
wire    SYNTHESIZED_WIRE_30;
wire    SYNTHESIZED_WIRE_31;
wire    SYNTHESIZED_WIRE_32;
wire    SYNTHESIZED_WIRE_33;
wire    SYNTHESIZED_WIRE_34;
wire    SYNTHESIZED_WIRE_35;


assign  shift_reset = SYNTHESIZED_WIRE_36;
assign  SYNTHESIZED_WIRE_8 = 1;
assign  SYNTHESIZED_WIRE_9 = 1;
assign  SYNTHESIZED_WIRE_11 = 0;
assign  SYNTHESIZED_WIRE_12 = 0;
assign  SYNTHESIZED_WIRE_13 = 0;
assign  SYNTHESIZED_WIRE_14 = 1;
assign  SYNTHESIZED_WIRE_15 = 1;
assign  SYNTHESIZED_WIRE_16 = 0;
assign  SYNTHESIZED_WIRE_17 = 1;
assign  SYNTHESIZED_WIRE_18 = 1;
assign  SYNTHESIZED_WIRE_19 = 1;
assign  SYNTHESIZED_WIRE_20 = 0;
assign  SYNTHESIZED_WIRE_21 = 1;
assign  SYNTHESIZED_WIRE_22 = 0;
assign  SYNTHESIZED_WIRE_23 = 0;
assign  SYNTHESIZED_WIRE_24 = 1;
assign  SYNTHESIZED_WIRE_25 = 0;
assign  SYNTHESIZED_WIRE_26 = 1;
assign  SYNTHESIZED_WIRE_27 = 1;
assign  SYNTHESIZED_WIRE_28 = 1;
assign  SYNTHESIZED_WIRE_29 = 1;
assign  SYNTHESIZED_WIRE_30 = 1;
assign  SYNTHESIZED_WIRE_31 = 1;
assign  SYNTHESIZED_WIRE_32 = 1;
assign  SYNTHESIZED_WIRE_33 = 1;
assign  SYNTHESIZED_WIRE_34 = 1;
assign  SYNTHESIZED_WIRE_35 = 1;
wire    [16:0]  GDFX_TEMP_SIGNAL_0;

assign  GDFX_TEMP_SIGNAL_0 = {d16,d15,d14,d13,d12,d11,
        d10,d9,d8,d7,d6,d5,d4,d3,d2,d1,d0};

controller      b2v_inst(
        .clk(clk),
        .reset(SYNTHESIZED_WIRE_36),
        .start(SYNTHESIZED_WIRE_1),
        .count17(SYNTHESIZED_WIRE_2),
        .available(available),
        .shift_enable(SYNTHESIZED_WIRE_38),
        .finish(finish));

lpm_counter0    b2v_inst1(
        .clock(SYNTHESIZED_WIRE_37),
        .cnt_en(SYNTHESIZED_WIRE_38),
        .aset(SYNTHESIZED_WIRE_36),
        .cout(SYNTHESIZED_WIRE_2),
        .q(SYNTHESIZED_WIRE_10));

alt_outbuf      b2v_inst10(
        .i(clk),
        .o(shift_clk));

lpm_dff_0       b2v_inst102(
        .clock(SYNTHESIZED_WIRE_37),
        .data(SYNTHESIZED_WIRE_38),
        .q(shift_enable));

alt_outbuf      b2v_inst104(
        .i(SYNTHESIZED_WIRE_8),
        .o(BIAS1));

alt_outbuf      b2v_inst107(
        .i(SYNTHESIZED_WIRE_9),
        .o(DIR3));

assign  SYNTHESIZED_WIRE_37 =  ~clk;

mux17   b2v_inst2(
        .d(GDFX_TEMP_SIGNAL_0),
        .select(SYNTHESIZED_WIRE_10),
        .q(serial_in));


alt_outbuf        b2v_inst28(
        .i(SYNTHESIZED_WIRE_11),
        .o(OEBAR2));


alt_outbuf        b2v_inst29(
        .i(SYNTHESIZED_WIRE_12),
        .o(VINNB));


assign   SYNTHESIZED_WIRE_36 =   ~reset;


alt_outbuf        b2v_inst30(
        .i(SYNTHESIZED_WIRE_13),
        .o(VINPB));




alt_outbuf        b2v_inst31(
        .i(SYNTHESIZED_WIRE_14),
        .o(shiftB));


alt_outbuf        b2v_inst32(
        .i(SYNTHESIZED_WIRE_15),
        .o(reset_DIGI));


alt_outbuf        b2v_inst33(
        .i(SYNTHESIZED_WIRE_16),
        .o(clk_DIGI));


alt_outbuf        b2v_inst35(
        .i(SYNTHESIZED_WIRE_17),
        .o(RVSSB));


alt_outbuf        b2v_inst36(
        .i(SYNTHESIZED_WIRE_18),
        .o(RVDDB));


alt_outbuf        b2v_inst37(
        .i(SYNTHESIZED_WIRE_19),
        .o(Latch2));


alt_outbuf        b2v_inst38(
        .i(SYNTHESIZED_WIRE_20),
        .o(OEBAR3));


alt_outbuf        b2v_inst390(
        .i(SYNTHESIZED_WIRE_21),
        .o(DIR2));


assign  SYNTHESIZED_WIRE_1 =  ~start;


alt_outbuf        b2v_inst40(
        .i(SYNTHESIZED_WIRE_22),
        .o(OEBAR1));

alt_outbuf        b2v_inst41(
        .i(SYNTHESIZED_WIRE_23),
        .o(VINP));

alt_outbuf        b2v_inst42(
        .i(SYNTHESIZED_WIRE_24),
        .o(shiftA));

alt_outbuf        b2v_inst43(
        .i(SYNTHESIZED_WIRE_25),
        .o(VINN));

alt_outbuf        b2v_inst44(
        .i(SYNTHESIZED_WIRE_26),
        .o(shiftABAR));

alt_outbuf        b2v_inst45(
        .i(SYNTHESIZED_WIRE_27),
        .o(Latch));

alt_outbuf        b2v_inst46(
        .i(SYNTHESIZED_WIRE_28),
        .o(RVDD));
```

```
alt_outbuf          b2v_inst47(
            .i(SYNTHESIZED_WIRE_29),
            .o(RVSS));

alt_outbuf          b2v_inst48(
            .i(SYNTHESIZED_WIRE_30),
            .o(IBIAS));



alt_outbuf          b2v_inst49(
            .i(SYNTHESIZED_WIRE_31),
            .o(IBIAS2));

alt_outbuf          b2v_inst50(
            .i(SYNTHESIZED_WIRE_32),
            .o(IBIAS3));

alt_outbuf          b2v_inst51(
            .i(SYNTHESIZED_WIRE_33),
            .o(IBIAS4));

alt_outbuf          b2v_inst56(
            .i(SYNTHESIZED_WIRE_34),
            .o(shiftBBAR));
```

```
alt_outbuf      b2v_inst57(
        .i(SYNTHESIZED_WIRE_35),
        .o(DIR1));

endmodule


module lpm_dff_0(clock,data,q);
/* synthesis black_box */

input clock;
input [0:0] data;
output [0:0] q;

endmodule
```

```

module controller (input clk, reset, start, count17,
output reg available, shift_enable, finish);
localparam [1:0] S0 = 2'b00, S1 = 2'b01, S2 = 2'b11;
reg [1:0] p_state, n_state;



always @( p_state, start, count17, reset)
begin: transition
n_state = S0;
case ( p_state )
```

```
S0: if(~start) n_state = S0;
else n_state = S1;
S1: if(~count17) n_state = S1;
else n_state = S2;
S2: if(~reset) n_state = S2;
else n_state = S0;
default: n_state = S0;
endcase
end

always @( p_state, start, count17, reset)
begin: Outputing
{available, shift_enable, finish} = 3'b000;
case ( p_state )
S0: {available, shift_enable, finish} = 3'b100;
S1: {available, shift_enable, finish} = 3'b010;
S2: {available, shift_enable, finish} = 3'b001;
default:
{available, shift_enable, finish} = 3'b000;
endcase
end

always @( posedge clk )
begin: sequential
if(reset) p_state <= S0;
else p_state <= n_state;
```

```
end
endmodule
```

```

module lpm_counter0 (
        aset,
        clock,
        cnt_en,
        cout,
        q);

        input       aset;
        input       clock;
        input       cnt_en;
        output      cout;
        output   [4:0]   q;

        wire   sub_wire0;
        wire  [4:0]  sub_wire1;
        wire   cout = sub_wire0;
        wire  [4:0]  q = sub_wire1[4:0];

        lpm_counter        LPM_COUNTER_component (
                                   .clock (clock),
                                   .aset (aset),
                                   .cnt_en (cnt_en),
                                   .cout (sub_wire0),
                                   .q (sub_wire1),
                                   .aclr (1'b0),
                                   .aload (1'b0),
                                   .cin (1'b1),
                                   .clk_en (1'b1),
                                   .data ({5{1'b0}}),
                                   .eq (),
                                   .sclr (1'b0),
                                   .sload (1'b0),
                                   .sset (1'b0),
                                   .updown (1'b1));
        defparam
LPM_COUNTER_component.lpm_avalue = "17",
LPM_COUNTER_component.lpm_direction = "DOWN",
LPM_COUNTER_component.lpm_port_updown = "PORT_UNUSED",
LPM_COUNTER_component.lpm_type = "LPM_COUNTER",
LPM_COUNTER_component.lpm_width = 5;

endmodule
```

```
module lpm_dff_0(clock,data,q);
input clock;
input [0:0] data;
output [0:0] q;

```

```
lpm_dff lpm_instance(.clock(clock),.data(data),.q(q));
          defparam          lpm_instance.LPM_WIDTH = 1;


endmodule
```

```
module mux17( select, d, q );


input[4:0] select;
input[16:0] d;
output q;


reg q;
wire[4:0] select;
wire[16:0] d;


always @( select or d )
begin
   case( select )
        17 : q = 1'b0;
        16 : q = d[0];
        15 : q = d[1];
        14 : q = d[2];
        13 : q = d[3];
        12 : q = d[4];
        11 : q = d[5];
        10 : q = d[6];
```

```
            9 : q = d[7];
          8 : q = d[8];
          7 : q = d[9];
          6 : q = d[10];
            5 : q = d[11];
          4 : q = d[12];
          3 : q = d[13];
          2 : q = d[14];
            1 : q = d[15];
          0 : q = d[16];
      endcase
end

endmodule
```

A.2 Low Jitter Delay-Locked Loop Digital System Verilog Code

```
// Digital System Verilog Code

module digital_system(
        REF_CLK,
        RESET,
        Vlow,
            dummyclk,
            dummyclkB,
            locksw,
            HandEnable,
            SLOWCLK,
            MUXRST,
            MUXWE,
            MUXSERIDATA,
            SHIFTRST,
            SHIFTWE,
            SHIFTSERIDATA,
            REF_LAG,
            digitalout,
            handcoarse,
            handfine,
            invmuxinputfinal,
            muxinputfinal
);


input  wire         REF_CLK;
input  wire         RESET;
input  wire         Vlow;
input  wire         dummyclk;
input  wire         dummyclkB;
input  wire         locksw;
input wire      HandEnable;
input wire      SLOWCLK;
input wire      MUXRST;
input wire      MUXWE;
input wire      MUXSERIDATA;
input wire      SHIFTRST;
input wire      SHIFTWE;
input wire      SHIFTSERIDATA;
input wire      REF_LAG;
output wire     [75:0] digitalout;
output wire     [5:0] handcoarse;
output wire     [12:0] handfine;
output wire     [31:0] invmuxinputfinal;
output wire     [31:0] muxinputfinal;


wire    [62:0] coarseout;
wire    [1:0] current;
wire    edgeout;
wire    [12:0] fineout;
wire    [5:0] handcoarse_ALTERA_SYNTHESIZED;
wire    [12:0] handfine_ALTERA_SYNTHESIZED;
wire    [4:0] muxinput_buffer;
wire    [62:0] thermometer;
wire    SYNTHESIZED_WIRE_0;
wire    SYNTHESIZED_WIRE_1;
wire    SYNTHESIZED_WIRE_2;
wire    SYNTHESIZED_WIRE_3;
wire    SYNTHESIZED_WIRE_4;
wire    SYNTHESIZED_WIRE_5;






rstAND  b2v_inst(
        .ina(SYNTHESIZED_WIRE_0),
        .inb(SYNTHESIZED_WIRE_1),
        .rst(RESET),
        .outc(edgeout));


risingdetector  b2v_inst1(
        .rclk(REF_LAG),
        .rst(RESET),
        .rd(Vlow),
        .rout(SYNTHESIZED_WIRE_0));


bintothem       b2v_inst10(
        .bin_coarsein(handcoarse_ALTERA_SYNTHESIZED),
        .thermometer(thermometer));

```

```
decoder5to32        b2v_inst11(
          .muxinput(muxinput_buffer),
          .invmuxinputfinal(invmuxinputfinal),
          .muxinputfinal(muxinputfinal));



fallingdetector  b2v_inst2(
          .fclk(REF_LAG),
          .rst(RESET),
          .fd(Vlow),
          .fout(SYNTHESIZED_WIRE_1));



statemachine        b2v_inst3(
          .clk(REF_CLK),
          .rst(RESET),
          .edgeout(edgeout),
          .MSBhigh85(SYNTHESIZED_WIRE_2),
          .MSBlow15(SYNTHESIZED_WIRE_3),
          .MSBhigh95(SYNTHESIZED_WIRE_4),
          .MSBlow5(SYNTHESIZED_WIRE_5),
          .locksw(locksw),
          .current(current));
```

```
controller          b2v_inst4(
            .clk(REF_CLK),
            .dummyclk(dummyclk),
            .dummyclkB(dummyclkB),
            .rst(RESET),
            .edgeout(edgeout),
            .CLK11(REF_LAG),
            .current(current),
            .MSBhigh85(SYNTHESIZED_WIRE_2),
            .MSBlow15(SYNTHESIZED_WIRE_3),
            .MSBhigh95(SYNTHESIZED_WIRE_4),
            .MSBlow5(SYNTHESIZED_WIRE_5),
            .coarseout(coarseout),
            .fineout(fineout));



outmux      b2v_inst6(
            .HandEnable(HandEnable),
            .coarseout(coarseout),
            .fineout(fineout),
            .handfine(handfine_ALTERA_SYNTHESIZED),
            .thermometer(thermometer),
            .digitalout(digitalout));



handshift           b2v_inst7(
            .clk(SLOWCLK),
            .rst(SHIFTRST),
            .we(SHIFTWE),
            .seridata(SHIFTSERIDATA),
            .handcoarse(handcoarse_ALTERA_SYNTHESIZED),
            .handfine(handfine_ALTERA_SYNTHESIZED));



muxinputshift    b2v_inst8(
            .clk(SLOWCLK),
            .rst(MUXRST),
            .we(MUXWE),
            .seridata(MUXSERIDATA),
            .muxinput_buffer(muxinput_buffer));

assign   handcoarse = handcoarse_ALTERA_SYNTHESIZED;
assign   handfine = handfine_ALTERA_SYNTHESIZED;

endmodule
```

```


// Controller Verilog Code
module controller(

input clk,
input dummyclk,
input dummyclkB,
input rst,
input edgeout,
input CLK11,
input [1:0] current,

output reg [62:0] coarseout,
output reg [12:0] fineout,
output reg MSBhigh85,
output reg MSBlow15,
output reg MSBhigh95,
output reg MSBlow5

);
reg [8:0] cnt1; // total number          9-bit 0 - 511
reg [8:0] cnt2;           // 1's number

reg flag;
reg finereset;




always@(posedge dummyclkB) begin
if(flag) finereset <= 1'b1;
```

```
else finereset <= 1'b0;
end

always@(posedge dummyclk or posedge rst) begin

if(rst) begin
flag <= 1'b0;
MSBhigh85 <= 1'b1;
MSBlow15 <= 1'b1;
MSBhigh95 <= 1'b1;
MSBlow5 <= 1'b1;
end

else begin
if(~finereset) begin              //when finereset = 0
if(cnt1[8:0] == 9'd511) begin
flag <= 1'b1;
MSBhigh85 <= cnt2[8] & cnt2[7] & cnt2[6];
MSBlow15 <= ~(cnt2[8] | cnt2[7] | cnt2[6]);
MSBhigh95 <= cnt2[8] & cnt2[7] & cnt2[6] & cnt2[5] & cnt2[4];
MSBlow5 <= ~(cnt2[8]|cnt2[7]|cnt2[6]|cnt2[5]|cnt2[4]);
end
end

else if(finereset) begin
flag <= 1'b0;
```

```
end
end
end

// Probability block
always@(posedge clk or posedge rst) begin
if(rst) begin
cnt1[8:0] <= 9'd0;
cnt2[8:0] <= 9'd0;
end

else if((~edgeout) & (~flag)) begin
if(cnt1[8:0] < 9'd511) begin
cnt1[8:0] <= cnt1[8:0] + 1'b1;
cnt2[8:0] <= cnt2[8:0] + CLK11; //thus, must use clk
end
end

else if ((cnt1[8:0]==9'd511)&(flag))begin
cnt1[8:0] <= 9'd0;
cnt2[8:0] <= 9'd0;
end

end
```

```
// coarse & fine control
always@(posedge clk or posedge rst) begin

//initial value in mode
if(rst) begin
coarseout[62:0] <= 63'b0_0000000000_0000000000_0000000000_
                        11_1111111111_1111111111_1111111111;
fineout[12:0] <= 13'b1000_0000_0000_0;
end

else begin
case(current)
2'b00:  // m0, coarse
if(CLK11) begin                     //CLK30, faster , NMOS
coarseout[62:0] <= {1'b0, coarseout[62:1]};
end

else begin                          //CLK30. slower
coarseout[62:0] <= {coarseout[61:0], 1'b1};
end



2'b01:  // m1, fine
if(flag) begin
if(MSBhigh85) fineout[12:0] <= fineout[12:0] -
                    13'b0_0000_0000_0001;// larger than 85%
```

```
else if
(MSBlow15) fineout[12:0] <= fineout[12:0] +
                 13'b0_0000_0000_0001; //less than 15%
end

2'b10: begin     // m2, locked
coarseout[62:0] <= coarseout[62:0];
fineout[12:0] <= fineout[12:0];
end

default:        begin   // just for avoiding    latches
coarseout[62:0] <= coarseout[62:0];
fineout[12:0] <= fineout[12:0];
end
endcase
end
end
endmodule
```

```
module statemachine(

input clk,
input rst,
input edgeout,
input MSBhigh85, MSBlow15,
input MSBhigh95, MSBlow5,
input locksw,
output reg [1:0] current

);

`define m0 2'b00                    // coarse mode
`define m1 2'b01                    //       fine mode
`define m2 2'b10                    // locked mode

always@(posedge clk or posedge rst) begin

if(rst) begin
current <= `m0;
end

else begin
case(current)
`m0:     // coarse
if(edgeout) current <= `m0;
else current <= `m1;
// edgeout = 0, transfer to fine mode

`m1:     // fine
if(locksw | (~(MSBhigh85 | MSBlow15)))  current <= `m2;
// in 15% - 85%, transfer to m2
```

```
else if(MSBhigh81 | MSBlow19) current <= ‘m1;
// larger than 85% or less than 15%, remain in m1

‘m2:      // locked
if((MSBhigh95 | MSBlow5)&(~locksw)) current <= ‘m1;
// larger than 95% or less than 5%, back to m1  //
else current <= ‘m2;
// in 5% - 95%, remain in m2
endcase
end


end


endmodule
```

```
module fallingdetector(


input fclk, rst, fd,
output reg fout


);


always@(negedge fclk or posedge rst)
begin
```

```
        if(rst) fout <= 1'b1;    //rst, high-active
        else fout <= fd;
end

endmodule
```

```
module risingdetector(

input rclk, rst, rd,
output reg rout

);

always@(posedge rclk or posedge rst)
begin
        if(rst) rout <= 1'b1;    //rst, high-active
        else rout <= rd;
end
endmodule
```

```
module rstAND(
input ina, inb, rst,
output outc
);

assign outc = (rst)? (1'b1):(ina & inb);

```

```
endmodule
```

```
module decoder5to32(

input [4:0]muxinput,

output reg [31:0]muxinputfinal,
output [31:0] invmuxinputfinal
);



assign invmuxinputfinal[31:0] = ~muxinputfinal[31:0];

always@(*)
begin
case(muxinput[4:0])

5'b00000: muxinputfinal[31:0] =
          32'b00_0000000000_0000000000_0000000001;
5'b00001: muxinputfinal[31:0] =
          32'b00_0000000000_0000000000_0000000010;
5'b00010: muxinputfinal[31:0] =
          32'b00_0000000000_0000000000_0000000100;
5'b00011: muxinputfinal[31:0] =
          32'b00_0000000000_0000000000_0000001000;
5'b00100: muxinputfinal[31:0] =
                32'b00_0000000000_0000000000_0000010000;
5'b00101:  muxinputfinal[31:0] =
                32'b00_0000000000_0000000000_0000100000;
5'b00110:  muxinputfinal[31:0] =
                32'b00_0000000000_0000000000_0001000000;
5'b00111:  muxinputfinal[31:0] =
                32'b00_0000000000_0000000000_0010000000;
5'b01000:  muxinputfinal[31:0] =
                32'b00_0000000000_0000000000_0100000000;
5'b01001:  muxinputfinal[31:0] =
                32'b00_0000000000_0000000000_1000000000;
5'b01010:  muxinputfinal[31:0] =
                32'b00_0000000000_0000000001_0000000000;
5'b01011:  muxinputfinal[31:0] =
                32'b00_0000000000_0000000010_0000000000;
5'b01100:  muxinputfinal[31:0] =
                32'b00_0000000000_0000000100_0000000000;
5'b01101:  muxinputfinal[31:0] =
                32'b00_0000000000_0000001000_0000000000;
5'b01110:  muxinputfinal[31:0] =
                32'b00_0000000000_0000010000_0000000000;
5'b01111:  muxinputfinal[31:0] =
                32'b00_0000000000_0000100000_0000000000;
5'b10000:  muxinputfinal[31:0] =
                32'b00_0000000000_0001000000_0000000000;
5'b10001:  muxinputfinal[31:0] =
              32'b00_0000000000_0010000000_0000000000;
5'b10010:  muxinputfinal[31:0] =
              32'b00_0000000000_0100000000_0000000000;
5'b10011:  muxinputfinal[31:0] =
              32'b00_0000000000_1000000000_0000000000;
5'b10100:  muxinputfinal[31:0] =
              32'b00_0000000001_0000000000_0000000000;
5'b10101:  muxinputfinal[31:0] =
              32'b00_0000000010_0000000000_0000000000;
5'b10110:  muxinputfinal[31:0] =
              32'b00_0000000100_0000000000_0000000000;
5'b10111:  muxinputfinal[31:0] =
              32'b00_0000001000_0000000000_0000000000;
5'b11000:  muxinputfinal[31:0] =
              32'b00_0000010000_0000000000_0000000000;
5'b11001:  muxinputfinal[31:0] =
              32'b00_0000100000_0000000000_0000000000;
5'b11010:  muxinputfinal[31:0] =
              32'b00_0001000000_0000000000_0000000000;
5'b11011:  muxinputfinal[31:0] =
              32'b00_0010000000_0000000000_0000000000;
5'b11100:  muxinputfinal[31:0] =
              32'b00_0100000000_0000000000_0000000000;
5'b11101:  muxinputfinal[31:0] =
              32'b00_1000000000_0000000000_0000000000;
5'b11110:  muxinputfinal[31:0] =
            32'b01_0000000000_0000000000_0000000000;
5'b11111: muxinputfinal[31:0] =
            32'b10_0000000000_0000000000_0000000000;
default  : muxinputfinal[31:0] =
            32'b00_0000000000_0000000000_0000000001;

endcase
end
endmodule
```

```
module bintothem (

bin_coarsein ,
thermometer

);

input [5:0] bin_coarsein;
output reg [62:0] thermometer;

always@(*)begin

case(bin_coarsein[5:0])

6'b000001: thermometer[62:0] <= 63'b0000000_00000000_
     00000000_00000000_00000000_00000000_00000000_00000001;
```

```
6'b000010: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000000_00000011;
6'b000011: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000000_00000111;
6'b000100: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000000_00001111;
6'b000101: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000000_00011111;
6'b000110: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000000_00111111;
6'b000111: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000000_01111111;
6'b001000: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000000_11111111;

6'b001001: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000001_11111111;
6'b001010: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000011_11111111;
6'b001011: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000111_11111111;
6'b001100: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00001111_11111111;
6'b001101: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00011111_11111111;
6'b001110: thermometer[62:0] <= 63'b0000000_00000000_
```

```
      00000000_00000000_00000000_00000000_00111111_11111111;
6'b001111: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_00000000_01111111_11111111;
6'b010000: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_00000000_11111111_11111111;

6'b010001: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_00000001_11111111_11111111;
6'b010010: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_00000011_11111111_11111111;
6'b010011: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_00000111_11111111_11111111;
6'b010100: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_00001111_11111111_11111111;
6'b010101: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_00011111_11111111_11111111;
6'b010110: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_00111111_11111111_11111111;
6'b010111: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_01111111_11111111_11111111;
6'b011000: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000000_11111111_11111111_11111111;

6'b011001: thermometer[62:0] <= 63'b0000000_00000000_
      00000000_00000000_00000001_11111111_11111111_11111111;
6'b011010: thermometer[62:0] <= 63'b0000000_00000000_
```

```
    00000000_00000000_00000011_11111111_11111111_11111111;
6'b011011: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000111_11111111_11111111_11111111;
6'b011100: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00001111_11111111_11111111_11111111;
6'b011101: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00011111_11111111_11111111_11111111;
6'b011110: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00111111_11111111_11111111_11111111;
6'b011111: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_01111111_11111111_11111111_11111111;
6'b100000: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_11111111_11111111_11111111_11111111;

// 33 - 40
6'b100001: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000001_11111111_11111111_11111111_11111111;
6'b100010: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000011_11111111_11111111_11111111_11111111;
6'b100011: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000111_11111111_11111111_11111111_11111111;
6'b100100: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00001111_11111111_11111111_11111111_11111111;
6'b100101: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00011111_11111111_11111111_11111111_11111111;
6'b100110: thermometer[62:0] <= 63'b0000000_00000000_
```

```
    00000000_00111111_11111111_11111111_11111111_11111111;
6'b100111: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_01111111_11111111_11111111_11111111_11111111;
6'b101000: thermometer[62:0] <= 63'b0000000_00000000_
    00000000_11111111_11111111_11111111_11111111_11111111;
//41 - 48
6'b101001: thermometer[62:0] <= 63'b0000000_00000000_
    00000001_11111111_11111111_11111111_11111111_11111111;
6'b101010: thermometer[62:0] <= 63'b0000000_00000000_
    00000011_11111111_11111111_11111111_11111111_11111111;
6'b101011: thermometer[62:0] <= 63'b0000000_00000000_
    00000111_11111111_11111111_11111111_11111111_11111111;
6'b101100: thermometer[62:0] <= 63'b0000000_00000000_
    00001111_11111111_11111111_11111111_11111111_11111111;
6'b101101: thermometer[62:0] <= 63'b0000000_00000000_
    00011111_11111111_11111111_11111111_11111111_11111111;
6'b101110: thermometer[62:0] <= 63'b0000000_00000000_
    00111111_11111111_11111111_11111111_11111111_11111111;
6'b101111: thermometer[62:0] <= 63'b0000000_00000000_
    01111111_11111111_11111111_11111111_11111111_11111111;
6'b110000: thermometer[62:0] <= 63'b0000000_00000000_
    11111111_11111111_11111111_11111111_11111111_11111111;
//49 - 56
6'b110001: thermometer[62:0] <= 63'b0000000_00000001_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b110010: thermometer[62:0] <= 63'b0000000_00000011_
```

```
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b110011: thermometer[62:0] <= 63'b0000000_00000111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b110100: thermometer[62:0] <= 63'b0000000_00001111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b110101: thermometer[62:0] <= 63'b0000000_00011111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b110110: thermometer[62:0] <= 63'b0000000_00111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b110111: thermometer[62:0] <= 63'b0000000_01111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b111000: thermometer[62:0] <= 63'b0000000_11111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
//57-63
6'b111001: thermometer[62:0] <= 63'b0000001_11111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b111010: thermometer[62:0] <= 63'b0000011_11111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b111011: thermometer[62:0] <= 63'b0000111_11111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b111100: thermometer[62:0] <= 63'b0001111_11111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b111101: thermometer[62:0] <= 63'b0011111_11111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
6'b111110: thermometer[62:0] <= 63'b0111111_11111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
```

```
6'b111111: thermometer[62:0] <= 63'b1111111_11111111_
    11111111_11111111_11111111_11111111_11111111_11111111;
default : thermometer[62:0] <= 63'b0000000_00000000_
    00000000_00000000_00000000_00000000_00000000_00000000;
endcase
end
endmodule
```

```
module handshift(

input clk,
input rst,
input we,
input seridata,

output [5:0] handcoarse,
output [12:0] handfine

);

reg [18:0] temp_buffer;
reg [18:0] temp;
reg [4:0] cnt;



assign handcoarse[5:0] = temp_buffer[18:13];
assign handfine[12:0] = temp_buffer[12:0];



always@(posedge clk) begin        // synchronous


        if(rst) begin
                temp[18:0] <= 19'd0;
        end


        else if (we) begin
                if (cnt[4:0] <= 5'd18) begin
                        temp[18:0] <= {temp[17:0],seridata};
                end
        end


end


always@(posedge clk) begin        // synchronous


        if(rst) begin
                cnt[4:0] <= 5'd0;
        end


        else if( (we) && (cnt[4:0] < 5'd20) )begin
                cnt[4:0] <= cnt[4:0] + 1'b1;
        end
```

```
end

always@(posedge clk) begin
        if(rst) begin
                temp_buffer[18:0] <= 19'd0;
        end

        else if(cnt[4:0] == 5'd19) begin
                temp_buffer[18:0] <= temp[18:0];
        end

end
endmodule
```

```
module muxinputshift(

input clk,
input rst,
input we,
input seridata,

output reg [4:0] muxinput_buffer
);

reg [2:0] cnt;
reg [4:0] temp;


always@(posedge clk) begin        // synchronous

         if(rst) begin
                  temp[4:0] <= 5'd0;
         end

         else if (we) begin
                  if (cnt[2:0] <= 3'd4) begin
                           temp[4:0] <= {temp[3:0],seridata};
                  end
         end

end

always@(posedge clk) begin        // synchronous

         if(rst) begin
                  cnt[2:0] <= 3'd0;
         end

         else if( (we) && (cnt < 3'd6) )begin
                  cnt[2:0] <= cnt[2:0] + 1'b1;
         end
```

```
end


// buffer once
always@(posedge clk) begin


        if (rst) begin
                muxinput_buffer[4:0] <= 5'd0;
        end


        else if (cnt == 3'd5) begin
                muxinput_buffer[4:0] <= temp[4:0];
        end


end
endmodule
```

```
module outmux(


input HandEnable,
input [62:0] coarseout,
input [12:0] fineout,




input [62:0] thermometer,
input [12:0] handfine,
output [75:0] digitalout

);

assign digitalout[75:0] = (HandEnable)?{thermometer[62:0],
        handfine[12:0]}:{coarseout[62:0],fineout[12:0]};

endmodule
```

A.3 Low Jitter Delay-Locked Loop chip evaluation FPGA code

The following code is realized on Altrea DE2-115 FPGA board.

```
// Build FPGA SEND & RECEIVE

// From FPGA to Chip
// SLOWCLK -> GPIO[3], clock divider reset control -> SW[4]

module JP_TEST(

input CLOCK3_50,     // used as the original clock
input [17:0] SW,

output reg SLOWCLK,
output reg SHIFTRST, SHIFTWE, SHIFTSERIDATA,
output reg MUXRST, MUXWE, MUXSERIDATA,

output Reset, // JP
output HandEnable,
output locksw,
output RESET_SL // SL

);

// (0)
// ============
// ------------------------
// Reset          -> GPIO[5], SW[0]
// HandEnable     -> GPIO[21], SW[1]
// locksw         -> GPIO[19], SW[2]
// RESET_SL       -> GPIO[23], SW[3]
assign Reset = SW[0];

assign HandEnable = SW[1];

assign locksw = SW[2];

assign RESET_SL = SW[3];

// SHIFTRST  -> GPIO[13]  , SW[5]
// SHIFTWE     -> GPIO[11]  , SW[6]
always@(SW[5]) begin
        SHIFTRST = SW[5];
end

always@(SW[6]) begin
        SHIFTWE = SW[6];
end



// MUXRST  -> GPIO[17]  , SW[7]
// MUXWE   -> GPIO[15]  , SW[8]
always@(SW[7]) begin
        MUXRST = SW[7];
end

always@(SW[8]) begin
        MUXWE = SW[8];
end



// ===============================
// From FPGA to Chip
//(1)
// Build Slow input clock
// -----------------------------------------------
```

```
reg [25:0] cnt_SLOWCLK;


// SLOWCLK -> GPIO[3], clock divider reset control -> SW[4]
always@(posedge CLOCK3_50) begin
        if (!SW[4]) begin
                cnt_SLOWCLK <= 26'd0;
                SLOWCLK <= 1'd0;
        end

        else begin
                if (cnt_SLOWCLK == 26'd25000000) begin
                        cnt_SLOWCLK <= 26'd0;
                        SLOWCLK <= ~SLOWCLK;
                end
                else begin
                        cnt_SLOWCLK <= cnt_SLOWCLK + 1'b1;
                end
        end
end



// (2)
// HAND Shift
// ------------------------------------------------------------
reg [5:0] cnt_HAND_DATA;
reg [18:0] HAND_DATA;
```

```
// SLOWCLK -> GPIO[3], clock divider reset control -> SW[4]
always@(posedge SLOWCLK) begin
          // SHIFTRST  -> GPIO[13] , SW[5]
          if (SHIFTRST) begin
                    cnt_HAND_DATA <= 6'd0;
          end
          // SHIFTWE    -> GPIO[11] , SW[6]
          else begin
                    if((SHIFTWE)&&(cnt_HAND_DATA < 6'd20)) begin
                    cnt_HAND_DATA <= cnt_HAND_DATA + 1'b1;
                    end
          end
end

always@(posedge SLOWCLK) begin
          // SHIFTRST  -> GPIO[13] , SW[5]
          if (SHIFTRST) begin
                    // SHIFTSERIDATA -> GPIO[7]
                    SHIFTSERIDATA <= 1'b0;
                    HAND_DATA <= 19'b100000_1111111111111;
                // Set Serial Data
          end
          // SHIFTWE    -> GPIO[11] , SW[6]
          else if ( (SHIFTWE) && ( cnt_HAND_DATA >= 6'd0 ) &&
                        ( cnt_HAND_DATA <= 6'd20 ) ) begin
          //SHIFTSERIDATA -> GPIO[7],
          SHIFTSERIDATA <= HAND_DATA[18];
          HAND_DATA[18:0]<={HAND_DATA[17:0],HAND_DATA[18]};
          end
end



// (3)
//MUX INPUT Shift
// ----------------------------------------

reg [5:0] cnt_MUX_DATA;
reg [4:0] MUX_DATA;
reg [4:0] MUX_DATA_TEMP;



// SLOWCLK -> GPIO[3], clock divider reset control -> SW[4]
always@(posedge SLOWCLK) begin
          // MUXRST  -> GPIO[17] , SW[7]
          if (MUXRST) begin
                    cnt_MUX_DATA <= 6'd0;
          end
          // MUXWE   -> GPIO[15] , SW[8]
          else begin
                    if ((MUXWE) && (cnt_MUX_DATA < 6'd6)) begin
                              cnt_MUX_DATA <= cnt_MUX_DATA + 1'b1;
                    end
          end
end

// Use Switches to input data
always@(posedge CLOCK3_50) begin
          // SW[12] enable
          if (SW[12]) begin
          MUX_DATA_TEMP<={SW[17],SW[16],SW[15],SW[14],SW[13]};
          end
end



// ------------------------------------------------
always@(posedge SLOWCLK) begin
          // MUXRST  -> GPIO[17]  , SW[7]
          if (MUXRST) begin
                    // MUXSERIDATA -> GPIO[9]
                    MUXSERIDATA <= 1'b0;
                    MUX_DATA <= MUX_DATA_TEMP;// Push data in

          end
          // MUXWE -> GPIO[15]  , SW[8]
          else if ( (MUXWE) && ( cnt_MUX_DATA >= 6'd0 )
                && ( cnt_MUX_DATA <= 6'd6 ) ) begin
                    //MUXSERIDATA -> GPIO[9],
                        MUXSERIDATA <= MUX_DATA[4];
                        MUX_DATA[4:0]<={MUX_DATA[3:0],MUX_DATA[4]};
            end
end
endmodule
```

Bibliography

[1] P. R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits, *IEEE J. Solid-State Circuits*, June 2005.

[2] A. Ashry, H. Aboushady, "A 4th Order 3.6GS/s RF $\Sigma\Delta$ ADC With a FoM of 1pJ/bit," *IEEE Trans. Circ. Syst. I*, vol. 60 no. 10, pp. 2606-2617, October, 2013.

[3] Dong-Shin Jo, Il-Hoon Jang, Dong-Suk Lee, Yong-Sang You, Yong-Hee Lee, Ho-Jin Park, Seung-Tak Ryu,"A 21fJ/conv-step 9 ENOB 1.6GS/S 2X time-interleaved FATI SAR ADC with background offset and timing-skew calibration in 45nm CMOS," *IEEE ISSCC 2015*, pp.1-3, Feb, 2015.

[4] Hyun-Wook Kang, Dong-Shin Jo, Dong-Suk Lee, Yong-Sang You, Yong-Hee Lee, Ho-Jin Park, Seung-Tak Ryu,"A 2.6b/cycle-architecture-based 10b 1 JGS/s 15.4mW 4X-time-interleaved SAR ADC with a multistep hardware-retirement technique,"*IEEE ISSCC 2015*, pp.1-3, Feb, 2015.

[5] J. McNeill, C. David, M. Coln, R. Croughwell, "Split ADC Calibration for All-Digital Correction of Time Interleaved ADC Errors," *IEEE Trans. Circ. Syst. II*, vol. 56 no. 5, pp. 344-348, May, 2009.

[6] N. Le Dortz, J-P. Blanc, T. Simon, S. Verhaeren, E. Rouat, P. Urard, S. Le Tual, D. Goguet, C. Lelandais-Perrault, P. Benabes,"A 1.62GS/s time-interleaved SAR ADC with digital background mismatch calibration achieving interleaving spurs below 70dBFS,"*IEEE ISSCC 2014*, pp.386-388, Feb, 2014.

[7] Benwei Xu ; Yuan Zhou ; Yun Chiu,"A 23-mW 24-GS/s 6-bit Voltage-Time Hybrid Time-Interleaved ADC in 28-nm CMOS,"*IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 1091-1100, 2017.

[8] Chun-Cheng Huang ; Chung-Yi Wang ; Jieh-Tsorng Wu,"A CMOS 6-Bit 16-GS/s Time-Interleaved ADC Using Digital Background Calibration Techniques,"*IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 848-858, 2011.

[9] Chin-Yu Lin ; Yen-Hsin Wei ; Tai-Cheng Lee,"A 10-bit 2.6-GS/s Time-Interleaved SAR ADC With a Digital-Mixing Timing-Skew Calibration Technique,"*IEEE J. Solid-State Circuits*, vol. 53, no. 5, pp. 1508-1517, 2018.

[10] Lukas Kull ; Jan Pliva ; Thomas Toifl ; Martin Schmatz ; Pier Andrea Francese ; Christian Menolfi ; Matthias Brandli ; Marcel Kossel ; Thomas Morf ; Toke Meyer Andersen ; Yusuf Leblebici,"Implementation of Low-Power 68 b 3090 GS/s Time-Interleaved ADCs With Optimized Input Bandwidth in 32 nm CMOS,"*IEEE J. Solid-State Circuits*, vol. 51, no. 3, pp. 636-648, 2016.

[11] Sandeep K. Gupta ; Michael A. Inerfield ; Jingbo Wang,"A 1-GS/s 11-bit ADC With 55-dB SNDR, 250-mW Power Realized by a High Bandwidth Scalable Time-Interleaved Architecture,"*IEEE J. Solid-State Circuits*, vol. 41, no. 12, pp. 2650-2657, 2006.

[12] Sunghyuk Lee ; Anantha P. Chandrakasan ; Hae-Seung Lee,"A 1 GS/s 10b 18.9 mW Time-Interleaved SAR ADC With Background Timing Skew Calibration,"*IEEE J. Solid-State Circuits*, vol. 49, no. 12, pp. 2846-2856, 2014.

[13] Si-Seng Wong ; U-Fat Chio ; Yan Zhu ; Sai-Weng Sin ; Seng-Pan U ; Rui Paulo Martins,"A 2.3 mW 10-bit 170 MS/s Two-Step Binary-Search Assisted Time-Interleaved SAR ADC,"*IEEE J. Solid-State Circuits*, vol. 48, no. 8, pp. 1783-1794, Aug, 2013.

[14] Kostas Doris ; Erwin Janssen ; Claudio Nani ; Athon Zanikopoulos ; Gerard van der Weide,"A 480 mW 2.6 GS/s 10b Time-Interleaved ADC With 48.5 dB SNDR up to Nyquist in 65 nm CMOS,"*IEEE J. Solid-State Circuits*, vol. 46, no. 12, pp. 2821-2833, 2011.

[15] Dusan Stepanovic ; Borivoje Nikolic,"A 2.8 GS/s 44.6 mW Time-Interleaved ADC Achieving 50.9 dB SNDR and 3 dB Effective Resolution Bandwidth of 1.5 GHz in 65 nm CMOS,"*IEEE J. Solid-State Circuits*, vol. 48, no. 4, pp. 971-982, 2013.

[16] Simon M. Louwsma ; A. J. M. van Tuijl ; Maarten Vertregt ; Bram Nauta"A 1.35 GS/s, 10 b, 175 mW Time-Interleaved AD Converter in 0.13 um CMOS,"*IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 778-786, 2008.

[17] Jae-Won Nam ; Mohsen Hassanpourghadi ; Aoyang Zhang ; Mike Shuo-Wei Chen"A 12-Bit 1.6, 3.2, and 6.4 GS/s 4-b/Cycle Time-Interleaved SAR ADC With Dual Reference Shifting and Interpolation,"*IEEE J. Solid-State Circuits*, vol. 53, no. 6, pp. 1765-1779, 2018.

[18] Vanessa H.-C. Chen ; Lawrence Pileggi"A 69.5 mW 20 GS/s 6b Time-Interleaved ADC With Embedded Time-to-Digital Calibration in 32 nm CMOS SOI,"*IEEE J. Solid-State Circuits*, vol. 49, no. 12, pp. 2891-2901, 2014.

[19] I.Ku, Z.Xu, Y.Kuan, Y.Wang, M.Chang.,"A 40-mW 7-bit 2.2-GS/s Time-Interleaved Subranging CMOS ADC for Low-Power Gigabit Wireless Communications," *IEEE J. Solid-State Circuits*, vol. 47, no. 8, pp. 225-228, Aug, 2013.

[20] K. Doris, E. Janssen, C. Nani, A. Zanikopoulos, G.van der Weide,"A 480 mW 2.6 GS/s 10b Time-Interleaved ADC With 48.5 dB SNDR up to Nyquist in 65 nm CMOS" *IEEE J. Solid-State Circuits*, vol. 46, no. 12, pp. 2821-2833, October, 2011.

[21] R.Sehgal, F.Goes, K.Bult.,"A 12 b 53 mW 195 MS/s Pipeline ADC with 82 dB SFDR Using Split-ADC Calibration," *IEEE J. Solid-State Circuits*, vol. 50, no. 7, pp. 1592-1603, July, 2015.

[22] S.Lee, Z.Xu, A.Chandrakasan, J.Lee.,"A 1 GS/s 10b 18.9 mW Time-Interleaved SAR ADC With Background Timing Skew Calibration," *IEEE J. Solid-State Circuits*, vol. 49, no. 12, pp. 2846-2856, Dec, 2014.

[23] M.Kijima, K.Ito, K.Kamei, S.Tsukamoto.,"A 6b 3GS/s Flash ADC with Background Calibration," *IEEE Custom Integrated Circuits Conf.*, pp. 283-286, Aug, 2009.

[24] J.McNeill, J.Gong, R.Majidi.,"Design of a Sub-Picosecond-Jitter Delay-Lock-Loop for Interleaved ADC Sample Clock Synthesis" *IEEE International Midwest Symposium on Circuits and Systems*, pp. 1854-1865, Aug, 2012.

[25] J. Gong and J. A. McNeill, "Sub-Picosecond-Jitter Clock Generation for Interleaved ADC," *2015 IEEE Conference on PhD Research in Microelectronics (PRIME2015)*, Glasgow, Scotland, June 2015.

[26] J. Gong, S. Li, J. A. McNeill, "Sub-Picosecond-Jitter Clock Generation for Interleaved ADC with Delay-Locked-Loop in 28nm CMOS, *IEEE 2016 Int'l Symposium on Circuits and Systems(ISCAS 2016)*, Montreal, Quebec, Canada, May 2016.

[27] J. Gong, S. Li, J. McNeill "Sub-Picosecond Jitter Clock Generation for Interleaved ADC," *Applied Sciences*, In preparation.

[28] M.-J. E. Lee, W. J. Dally, T. Greer, H.-T. Ng, R. Farjad-Rad, J. Poulton, and R. Senthinathan,"Jitter transfer characteristics of delay-locked loops-theories and design techniques" *IEEE J. Solid-State Circuits*, vol. 38, no. 4, pp. 614621, Apr. 2003

[29] H.Chang, S.Liu, "A wide-range and fast-locking all-digital cycle-controlled delay-locked loop," *IEEE J. Solid-State Circuits*, vol.40, no. 3, pp.661-670, March 2005

[30] Mike Peng Li, "Jitter, Noise, and Integrity at High-Speed." Prentice Hall, 2007.

[31] J.J. Ou ; Xiaodong Jin ; Chenming Hu ; P.R. Gray.,"Submicron CMOS thermal noise modeling from an RF perspective," *Symposium on VLSI Technology. Digest of Technical Papers*, June, 1999.

[32] C. Fiegna.,"Analysis of gate shot noise in MOSFETs with ultrathin gate oxides," *IEEE Electron Device Letters*, vol. 24, no. 2, pp. 108-110, Aug, 2003.

[33] Bede Liu.,"Clock jitter in sampling and reconstruction," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May, 2014.

[34] R. Navid ; T.H. Lee ; R.W. Dutton.,"An analytical formulation of phase noise of signals with Gaussian-distributed jitter," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, no. 3, pp. 149-153, March, 2005.

[35] J. Buckwalter ; B. Analui ; A. Hajimiri.,"Data-dependent jitter and crosstalk-induced bounded uncorrelated jitter in copper interconnects," *2004 IEEE MTT-S International Microwave Symposium Digest*, June, 2004.

[36] J.Rabaey, A.Chandrakasan, B.Nikolic.,"Digital Integrated Circuits: A Design Perspective,2nd Edition" Pearson Education, 2003.

[37] Gene L. Harding, "A jitter education A more detailed look at jitter for the sophomore, " *37th Annual Frontiers In Education Conference* , 2007.

[38] Ransom Stephens, "All About the Acronyms: RJ,DJ,DDJ,ISI,DCD,PJ,SJ..., " *Jitter 360 Series*

[39] Johnnie Hancock, "JitterUnderstanding it, Measuring It, Eliminating It; Part 3: Causes of Jitter " *High Frequency Electronics* , June 2004.

[40] Fangyi Rao ; Sammy Hindi.,"Jitter induced voltage noise in clock channels," *2014 IEEE International Symposium on Electromagnetic Compatibility (EMC)*,, Aug, 2014.

[41] F. Rao and S. Hindi.,"Frequency domain analysis of jitter amplification in clock channels," *Proc. IEEE 21th Topical Meeting on Electric Performance of Electronic Packaging*, Tempe, A,. pp. 51-54, Oct, 2012.

[42] Dennis Peterich , Bill Ham , Schelto Van , Doorn Edward , L. Grivna.,"Fibre Channel - Methodologies for Jitter and Signal Quality Specification - MJSQ," *INCITS, Tech. Rep. REV 10.0*, March, 2003.

[43] B. Analui ; J.F. Buckwalter ; A. Hajimiri.,"Data-dependent jitter in serial communications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 11, pp. 3388-3397, Nov, 2005.

[44] Mistry, D.; Joshi, S.; Agrawal, N.,"A novel jitter separation method based on Gaussian mixture model," *2015 International Conference on Pervasive Computing (ICPC)*, pp. 1-4, Aug, 2015.

[45] Klodjan Bidaj ; Jean-Baptiste Begueret ; Jerome Deroo.,"RJ/DJ jitter decomposition technique for high speed links," *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 584-587, 2016.

[46] Behzad Razavi, "Design of Analog CMOS Integrated Circuit" McGraw Hill

[47] A. A. Abidi, "High-Frequency Noise Measurements on FETs with Small Dimensions," *IEEE Trans. Electron Devices*, vol. 33, pp. 1801-1805, Nov 1986.

[48] Paul R. Gray, Paul J. Hurst, Stephen H. Lewis, Robert G. Meyer, "Analysis and Design of Analog Integrated Circuits, 5th Edition" Wiley

[49] V.R. von Kaenel, "A High-Speed, Low-Power Clock Generator for a Microprocessor Application," *IEEE Journal of Solid-State Circuits*, vol. 33 no. 11, pp. 1634-1639, 1998.

[50] Nilsson P. and Torkelson M, "A Monolithic Digital Clock-Generator for On-Chip Clocking of Custom DSPs," *IEEE Journal of Solid-State Circuits*, vol. 31 no. 5, pp. 700-706, 1996.

[51] Hsi-En Liu ; Chun-Jen Su ; Chih-Kang Cheng ; Wen-Kuen Liu, "Design and modeling of PLL-based clock and data recovery circuits with periodically embedded clock encoding for intra-panel interfaces," *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2234-2237, 2016.

[52] Wei-Hao Chiu ; Yu-Hsiang Huang ; Tsung-Hsien Lin, "A Dynamic Phase Error Compensation Technique for Fast-Locking Phase-Locked Loops," *IEEE J. Solid-State Circuits*, vol. 45, no. 6, pp. 11371149, 2010.

[53] Ting-Kuei Kuan ; Shen-Iuan Liu, "A Bang Bang Phase-Locked Loop Using Automatic Loop Gain Control and Loop Latency Reduction Techniques," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 821831, 2016.

[54] Wei-Zen Chen ; Jieh-Tsorng Wu, "A 2-V, 1.8-GHz BJT phase-locked loop," *IEEE J. Solid-State Circuits*, vol. 34, no. 6, pp. 784789, 1999.

[55] Joonsuk Lee ; Beomsup Kim, "A low-noise fast-lock phase-locked loop with adaptive bandwidth control," *IEEE J. Solid-State Circuits*, vol. 35, no. 8, pp. 11371145, 2000.

[56] B. Razavi, "A 2-GHz 1.6-mW phase-locked loop" *IEEE J. Solid-State Circuits*, vol. 32, no.5, pp. 730735, 1997.

[57] Kyoohyun Lim ; Chan-Hong Park ; Dal-Soo Kim ; Beomsup Kim, "A low-noise phase-locked loop design by loop bandwidth optimization," *IEEE J. Solid-State Circuits*, vol. 35, no. 6, pp. 807815, 2000.

[58] Taekwang Jang ; Seokhyeon Jeong ; Dongsuk Jeon ; Kyojin David Choo ; Dennis Sylvester ; David Blaauw, “A Noise Reconfigurable All-Digital Phase-Locked Loop Using a Switched Capacitor-Based Frequency-Locked Loop and a Noise Detector,” *IEEE J. Solid-State Circuits*, vol. 53, no. 1, pp. 5065, 2018.

[59] Hsiang-Hui Chang ; Shen-Iuan Liu, “A wide-range and fast-locking all-digital cycle-controlled delay-locked loop,” *IEEE J. Solid-State Circuits*, vol. 40, no. 3, pp. 661670, 2005.

[60] Chih-Ming Hung ; K.K. O, “A fully integrated 1.5-V 5.5-GHz CMOS phase-locked loop,” *IEEE J. Solid-State Circuits*, vol. 37, no. 4, pp. 521525, 2002.

[61] Terng-Yin Hsu ; Bai-Jue Shieh ; Chen-Yi Lee, “An all-digital phase-locked loop (ADPLL)-based clock recovery circuit,” *IEEE J. Solid-State Circuits*, vol. 34, no. 8, pp. 10631073, 1999.

[62] Shravan S. Nagam ; Peter R. Kinget, “A Low-Jitter Ring-Oscillator Phase-Locked Loop Using Feedforward Noise Cancellation With a Sub-Sampling Phase Detector,” *IEEE J. Solid-State Circuits*, vol. 53, no. 3, pp. 703714, 2018.

[63] M. Mansuri ; D. Liu ; C.-K.K. Yang, “Fast frequency acquisition phase-frequency detectors for Gsamples/s phase-locked loops,” *IEEE J. Solid-State Circuits*, vol. 37, no. 10, pp. 13311334, 2002.

[64] Ping-Hsuan Hsieh ; Jay Maxey ; Chih-Kong Ken Yang, “A Phase-Selecting Digital Phase-Locked Loop With Bandwidth Tracking in 65-nm CMOS Technology,” *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 781792, 2010.

[65] B.R. Veillette ; G.W. Roberts, “On-chip measurement of the jitter transfer function of charge-pump phase-locked loops,” *IEEE J. Solid-State Circuits*, vol. 33, no. 3, pp. 483491, 1998.

[66] Guanghua Shu ; Saurabh Saxena ; Woo-Seok Choi ; Mrunmay Talegaonkar ; Rajesh Inti ; Amr Elshazly ; Brian Young ; Pavan Kumar Hanumolu, “A Reference-Less Clock and Data Recovery Circuit Using Phase-Rotating Phase-Locked Loop,” *IEEE J. Solid-State Circuits*, vol. 49, no. 4, pp. 10361047, 2014.

[67] Wei-Sung Chang ; Po-Chun Huang ; Tai-Cheng Lee, “A Fractional-N Divider-Less Phase-Locked Loop With a Subsampling Phase Detector,” *IEEE J. Solid-State Circuits*, vol. 49, no. 12, pp. 29642975, 2014.

[68] Eunseok Song ; Seung-Wook Lee ; Jeong-Woo Lee ; Joonbae Park ; Soo-Ik Chae, “A reset-free anti-harmonic delay-locked loop using a cycle period detector,” *IEEE J. Solid-State Circuits*, vol. 39, no. 11, pp. 20552061, 2004.

[69] Chao-Chyun Chen ; Shen-Iuan Liu, “An Infinite Phase Shift Delay-Locked Loop With Voltage-Controlled Sawtooth Delay Line,” *IEEE J. Solid-State Circuits*, vol. 43, no. 11, pp. 24132421, 2008.

[70] Amr Elshazly ; Rajesh Inti ; Brian Young ; Pavan Kumar Hanumolu, “Clock Multiplication Techniques Using Digital Multiplying Delay-Locked Loops,” *IEEE J. Solid-State Circuits*, vol. 48, no. 6, pp. 14161428, 2013.

[71] Hsiang-Hui Chang ; Jyh-Woei Lin ; Ching-Yuan Yang ; Shen-Iuan Liu, “A wide-range delay-locked loop with a fixed latency of one clock cycle,” *IEEE J. Solid-State Circuits*, vol. 37, no. 8, pp. 10211027, 2002.

[72] Jinn-Shyan Wang ; Chun-Yuan Cheng ; Pei-Yuan Chou ; Tzu-Yi Yang, “A Wide-Range, Low-Power, All-Digital Delay-Locked Loop With Cyclic Half-Delay-Line Architecture,” *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 26352644, 2015.

[73] Yeon-Jae Jung ; Seung-Wook Lee ; Daeyun Shim ; Wonchan Kim ; Changhyun Kim ; Soo-In Cho, “A dual-loop delay-locked loop using multiple voltage-controlled delay lines,” *IEEE J. Solid-State Circuits*, vol. 36, no. 5, pp. 784791, 2001.

[74] Hsiang-Hui Chang ; Jyh-Woei Lin ; Shen-Iuan Liu, “A fast locking and low jitter delay-locked loop using DHDL,” *IEEE J. Solid-State Circuits*, vol. 38, no. 2, pp. 343346, 2003.

[75] A. Efendovich ; Y. Afek ; C. Sella ; Z. Bikowsky, “Multifrequency zero-jitter delay-locked loop,” *IEEE J. Solid-State Circuits*, vol. 29, no. 1, pp. 6770, 1994.

[76] Salvatore Levantino ; Giovanni Marucci ; Giovanni Marzin ; Andrea Fenaroli ; Carlo Samori ; Andrea L. Lacaita, “A 1.7 GHz Fractional-N Frequency Synthesizer Based on a Multiplying Delay-Locked Loop,” *IEEE J. Solid-State Circuits*, vol. 50, no. 11, pp. 26782691, 2015.

[77] F. Baronti ; D. Lunardini ; R. Roncella ; R. Saletti, “A self-calibrating delay-locked delay line with shunt-capacitor circuit scheme,” *IEEE J. Solid-State Circuits*, vol. 39, no. 2, pp. 384387, 2004.

[78] Hyunik Kim ; Yongjo Kim ; Taeik Kim ; Hyung-Jong Ko ; Seonghwan Cho, “A 2.4-GHz 1.5-mW Digital Multiplying Delay-Locked Loop Using Pulsewidth Comparator and Double Injection Technique,” *IEEE J. Solid-State Circuits*, vol. 52, no.11, pp. 29342946, 2017.

[79] Chun-Cheng Huang ; Chung-Yi Wang ; Jieh-Tsorng Wu,“A CMOS 6-Bit 16-GS/s Time-Interleaved ADC Using Digital Background Calibration Techniques” *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 848858, 2011

[80] Sunghyuk Lee ; Anantha P. Chandrakasan ; Hae-Seung Lee,“A 1 GS/s 10b 18.9 mW Time-Interleaved SAR ADC With Background Timing Skew Calibration” *IEEE J. Solid-State Circuits*, vol. 49, no. 12, pp. 28462856, 2014

[81] Kostas Doris ; Erwin Janssen ; Claudio Nani ; Athon Zanikopoulos ; Gerard van der Weide,“A 480 mW 2.6 GS/s 10b Time-Interleaved ADC With 48.5 dB SNDR up to Nyquist in 65 nm CMOS” *IEEE J. Solid-State Circuits*, vol. 46, no. 12, pp. 28212833, 2011

[82] Simon M. Louwsma ; A. J. M. van Tuijl ; Maarten Vertregt ; Bram Nauta,“A 1.35 GS/s, 10 b, 175 mW Time-Interleaved AD Converter in 0.13 μm CMOS” *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 778786, 2008

[83] Kexu Sun ; Guanhua Wang ; Qing Zhang ; Salam Elahmadi ; Ping Gui,“A 56-GS/s 8-bit Time-Interleaved ADC With ENOB and BW Enhancement Techniques in 28-nm CMOS” *IEEE J. Solid-State Circuits*, Dec, 2018

[84] Takuji Miki ; Toshiaki Ozeki ; Jun-ichi Naka,“A 2-GS/s 8-bit Time-Interleaved SAR ADC for Millimeter-Wave Pulsed Radar Baseband SoC” *IEEE J. Solid-State Circuits*, vol. 52, no. 10, pp. 27122710, 2008

[85] M. El-Chammas and B. Murmann,“ Background Calibration of Time-Interleaved Data Converters(Analog Circuits and Signal Processing)”, Springer Science+Business Media, LLC 2012

[86] IDT Understanding Jitter Units Application Note

[87] John A. McNeill and David Ricketts, “The Designer’s Guide to Jitter in Ring Oscillators.” Springer Science+Business Media, LLC, 2009.

[88] H. Wei, P. Zhang, B. Datta Sahoo, B. Razavil, “An 8-Bit 4-GS/s 120-mW CMOS ADC,” *IEEE Custom Integrated Circuits Conf.*, pp. 22-25, Sept, 2013.

[89] M. Soyuer, R. G. Meyer, “Frequency limitations of a conventional phase-frequency detector” *IEEE J. Solid-State Circuits*, vol. 25, no. 4, pp. 1019-1022, August 2002.

[90] M. Horowitz, D. Stark, E. Alon, “Digital Circuit Design Trends” *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 757-761, March 2008.

[91] G. Chien, P. Gray, “A 900-MHz local oscillator using a DLL-based frequency multiplier technique for PCS application,” *IEEE J. Solid-State Circuits*, Dec. 2000, pp. 1996-1999.

[92] Mohammad Gholami, “Total Jitter of Delay-Locked Loops Due to Four Main Jitter Sources” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2040-2049, Jun 2016.

[93] Chi-Nan Chuang ; Shen-Iuan Liu, “A 38 GHz Delay-Locked Loop With Cycle Jitter Calibration” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 11, pp. 1094-1098, Nov 2008.

[94] C. N. Chuang and S. I. Liu, “A 0.55 GHz wide-range multi-phase DLL with a calibrated charge pump” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 11, pp. 939-943, Nov 2007.

[95] R. J. Yang and S. I. Liu, “A 2.5 GHz all-digital delay-locked loop in 0.13um CMOS technology” *IEEE J. Solid-State Circui*, vol. SC-42, pp. 2338-2347, Nov 2007.

[96] A. Alvandpour, R. K. Krishnamurthy, D. Eckerbert, S. Apperson, B. Bloechel, and S. Bork, “A 3.5 GHz 32 mW 150 nm multiphase clock generator for high-performance microprocessor” *IEEE Int. SolidState Circuits Conf. Dig. Tech. Paper*, pp. 112-113, Feb 2003.

[97] T. Hamamoto, K. Furutani, T. Kubo, S. Kawasaki, H. Iga, T. Kono, Y. Konishi, and T. Yoshihara, “A 667-Mb/s operating digital DLL architecture for 512-Mb DDR SDRAM” *IEEE J. Solid-State Circuits*, vol. 39, no. 1 pp. 194-206, Jan 2004.

[98] Y.-J. Jeon, J.-H. Lee, H.-C. Lee, K.-W. Jin, K.-S. Min, J.-Y. Chung, and H.-J. Park, “A 66333MHz 12-mW register-controlled DLL with a single delay line and adaptive-duty-cycle clock dividers for production DDR SDRAMs” *IEEE J. Solid-State Circuits*, vol. 39, no. 11 pp. 2087-2092, Nov 2004.

[99] T. Matano, Y. Takai, T. Takahashi, Y. Sakito, I. Fujii, Y. Takaishi, H. Fujisawa, S. Kubouchi, S. Narui, K. Arai, M. Morino, M. Nakamura, S. Miyatake, T. Sekiguchi, and K. Koyama, “A 1-Gb/s/pin 512Mb DDRII SDRAM using a digital DLL and a slew-rate-controlled output buffer” *IEEE J. Solid-State Circuits*, vol. 38, no. 5 pp. 762-768, May 2003.

[100] A. Alvandpour, R. K. Krishnamurthy, D. Eckerbert, S. Apperson, B. Bloechel, and S. Borkar, “A 3.5 GHz 32 mW 150 nm multiphase clock generator for high-performance microprocessors” *IEEE Int. SolidState Circuits Conf. Dig. Tech. Papers*, pp. 112-113, 2003.

[101] K. Minami, M. Mizuno, H. Yamaguchi, T. Nakano, Y. Matsushima, Y. Sumi, T. Sato, H. Yamashida, and M. Yamashina, “A 1 GHz portable digital delay-locked loop with infinite phase capture ranges” *IEEE Int. SolidState Circuits Conf. Dig. Tech. Papers*, pp. 350-351, 2000.

[102] J.-S. Wang, Y.-M. Wang, C.-H. Chen, and Y.-C. Liu, “An ultra-lowpower fast-lock-in small-jitter all-digital DLL” *IEEE Int. SolidState Circuits Conf. Dig. Tech. Papers*, pp. 422-423, 2005.

[103] R.-J. Yang and S.-I. Liu, “A 40 550 MHz harmonic-free all-digital delay-locked loop using a variable SAR algorithm” *IEEE J. Solid-State Circuits*, vol. 42, no. 2 pp. 361-373, Feb 2007.

[104] D. Shin, J. Song, H. Chae, K.-W. Kim, Y.-J. Choi, and C. Kim, “A 7ps 0.053 mm2 fast-lock all-digital DLL with wide-range and high resolution all digital DCC” *IEEE Int. SolidState Circuits Conf. Dig. Tech. Papers*, pp. 184-185, 2007.