2007

High-speed Low-voltage CMOS Flash Analog-to-Digital Converter for Wideband Communication System-on-a-Chip

Mingzhen Wang
Wright State University

HIGH-SPEED LOW-POWER CMOS FLASH ANALOG-TO-DIGITAL CONVERTER FOR WIDEBAND COMMUNICATION SYSTEM-ON-A-CHIP

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

By

MINGZHEN WANG
B.S.in Computer Engineering, Huazhong University of Science and Technology, 1990
M.S.Egr. in Computer Science, Wright State University, 2002
M.S.Egr. in Electrical Engineering, Wright State University, 2005

2007
Wright State University

WRIGHT STATE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

September 11, 2007

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Mingzhen Wang ENTITLED High-Speed Low-Power CMOS Flash Analog-to-Digital Converter for Wideband Communication Systems-on-a-Chip BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

_____
Chien-In Henry Chen, Ph.D.
Dissertation Director

_____
Ramana V. Grandhi, Ph.D.
Director, Ph.D. in Electrical Engineering

_____
Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies

Committee on
Final Examination

_____
Chien-In Henry Chen, Ph.D.

_____
Raymond Siferd, Ph.D.

_____
Marian Kazimierczuk, Ph.D.

_____
Jack S.N. Jean, Ph.D.

_____
Wen-Ben Jone, Ph.D.

ABSTRACT

Wang, Mingzhen, Ph.D, Engineering Ph.D Program, Department of Electrical Engineering, Wright State University, 2007. High-Speed Low-Voltage CMOS Flash Analog-to-Digital Converter for Wideband Communication System-on-a-Chip

With higher-level integration driven by increasingly complex digital systems and downscaling CMOS processes available, system-on-a-chip (SoC) is an emerging technology of low power, high cost effectiveness and high reliability and is exceedingly attractive for applications in high-speed data conversion wireless and wideband communication systems.

This research presents a novel ADC comparator design methodology; the speed and performance of which is not restricted by the supply voltage reduction and device linearity deterioration in scaling-down CMOS processes. By developing a dynamic offset suppression technique and a circuit optimization method, the comparator can achieve a 3 dB frequency of 2 GHz in 130 nanometer (nm) CMOS process.

Combining this new comparator design and a proposed pipelined thermometer-Gray-binary encoder designed by the DCVSPG logic, a high-speed, low-voltage clocked-digital-comparator (CDC) pipelined CMOS flash ADC architecture is proposed for wideband communication SoC. This architecture has advantages of small silicon area, low power, and low cost. Three CDC-based pipelined CMOS flash ADCs were implemented in 130 nm CMOS process and their experimental results are reported:

1. 4-b, 2.5-GSPS ADC: SFDR of 21.48-dB, SNDR of 15.99-dB, ENOB of 2.4-b, ERBW of 1-GHz, power of 7.9-mW, and area of 0.022-mm^2.

2. 4-b, 4-GSPS ADC: SFDR of 25-dB, SNDR of 18.6-dB, ENOB of 2.8-b, ERBW of 2-GHz, power of 11-mW.
3. 6-b, 4-GSPS ADC: SFDR of 48-dB at a signal frequency of 11.72-MHz, SNDR of 34.43-dB, ENOB of 5.4-b, power of 28-mW.

An application of the proposed CDC-based pipelined CMOS flash ADC is 1-GHz bandwidth, 2.5-GSPS digital receiver on a chip. To verify the performance of the receiver, a mixed-signal block-level simulation and verification flow was built in Cadence AMS integrated platform. The verification results of the digital receiver using a 4-b 2.5-GSPS CDC-based pipelined CMOS ADC, a 256-point, 12-point kernel function FFT and a frequency detection logic show that two tone signals up to 1125 MHz can be detected and discriminated.

A notable contribution of this research is that the proposed ADC architecture and the comparator design with dynamic offset suppression and optimization are extremely suitable for future VDSM CMOS processes and make “all-digital” receiver SoC design practical.

Contents

List of Figures

List of Tables

Acknowledgements

I am grateful to many people who supported, encouraged and guided me during the dissertation research work. My most sincere gratitude belongs to my dissertation advisor, Prof. Chien-In Henry Chen, for his guidance and support throughout this endeavor. His good insight on my dissertation research is invaluable, and he has been providing full support on the research source. I also thank him for his constant encouragement and patience throughout the entire research period. This doctoral research experience would be a special journey in my professional life.

I would like to sincerely thank Prof. Raymond Siferd for taking me as an IC design research assistant at the first place of my doctoral study and teaching me the analog circuit design. I also thank Prof. Marian Kazimierczuk and Prof. Jack Jean for their encouragement and guidance during my graduate study. I am truly grateful to Prof. Wen-Ben Jone, Prof. Marian Kazimierczuk, Prof. Raymond Siferd and Prof. Jack Jean for willing to serve on my dissertation committee.

I thank all the colleagues in the VLSI research lab for sharing and discussion. Mike Myers deserves a special thank for helping me at the beginning of my research in the VLSI lab, and also with design layout in the end. Without his help, my ADC layout would not be completed successfully in that short time.

I extend my acknowledgements to Sheila Hollenbaugh and the Department of

Electrical Engineering for providing me all the necessary technical support required for the successful completion of this dissertation. I would also like to thank the Engineering Ph. D. Program for the financial support.

I am most grateful to my family, my husband and daughter, my mom, my brothers and sisters for their love and support during my doctoral study. The most special gratitude goes to my husband, Li, for teaching me technical writing and organizing the research, and also taking care of our daughter, Jiechen, when I was not there. Without him, I would never get to this point.

Dedicated to

My husband and our daughter, Li and Jiechen.

Chapter 1 INTRODUCTION

1.1 Background

Analog-to-digital converters (ADCs) are electronic circuits that convert continuous electric signals into discrete digital numbers for signal analysis or signal transmission. ADCs are characterized primarily by conversion speed, resolution, power and area consumption, which vary with a variety of applications. In addition, selection of an ADC for an individual application is essential for cost effectiveness.

With higher-level integration driven by increasingly complex digital systems and downscaling processes available, system-on-a-chip (SoC) is an emerging technology for low power-consumption, high cost effectiveness, and high reliability. SoC is in great demand for wireless and wideband communication systems which require high-speed data conversion, especially consumer electronic devices. SoC integrates all components of electronic systems into a single integrated circuit containing digital, analog, and mixed-signal functions on one chip. A high-speed A/D converter becomes an integral part of SoC products.

Generally three types of solid state technologies are available for A/D converter implementation. They are Gallium Arsenide (GaAs), Silicon Germanium (SiGe) and

Silicon (Si), and of the three GaAs is the most and Si is the least expense process. As to the Si process the CMOS Si, unlike the BiCMOS Si using high power consumption bipolar devices, uses low-power NFET and PFET devices which makes CMOS mixed-signal/SoC become an emerging technology in the semiconductor industry.

However, shrinking the feature size of the semiconductor technology makes high-speed high-performance ADC design quite challenge in a standard CMOS technology, due to the reduced effective voltage headroom and the deteriorated linearity of transistor device in the small feature size technology. The semiconductor technology is now approaching the 45 nanometer feature size, pushing development of SoC applications to a new milestone.

Giga-sample-per-second (GSPS) flash-based ADCs in CMOS technology were reported [1]-[3]. Although they had improved the conversion speed by using time-interleaving, the conversion precision by averaging offset, or the effective resolution bandwidth (ERBW) by interpolating and folding, etc., the sample rate is still limited by 2 GHz. Besides, these ADCs are not suitable for SoC due to noise effect on the analog components, and high power consumption. We need a robust ADC architecture to minimize substrate noise in a SoC core and has as few analog nodes and components as possible [4].

1.2 Initial Research Requirements

The initial objective of this research is to investigate and design a high-speed, low-power, and low-voltage CMOS A/D converter for use in a 2.5 GSPS, 1 GHz wideband digital receiver on a chip, as shown in Figure 1-1. The receiver is to produce a low cost (fewer devices and less PCB area), small and lightweight (130 nanometers CMOS) and low power (less than 2 W) SoC for correctly processing two simultaneous signals (in a frequency range between 125 and 1125 MHz) by detecting their frequency, pulse width (PW), and time of arrival (TOA). The design of digital receiver is divided into two areas: 1) signal sampler and formatting and 2) super resolution and frequency measurement [5].

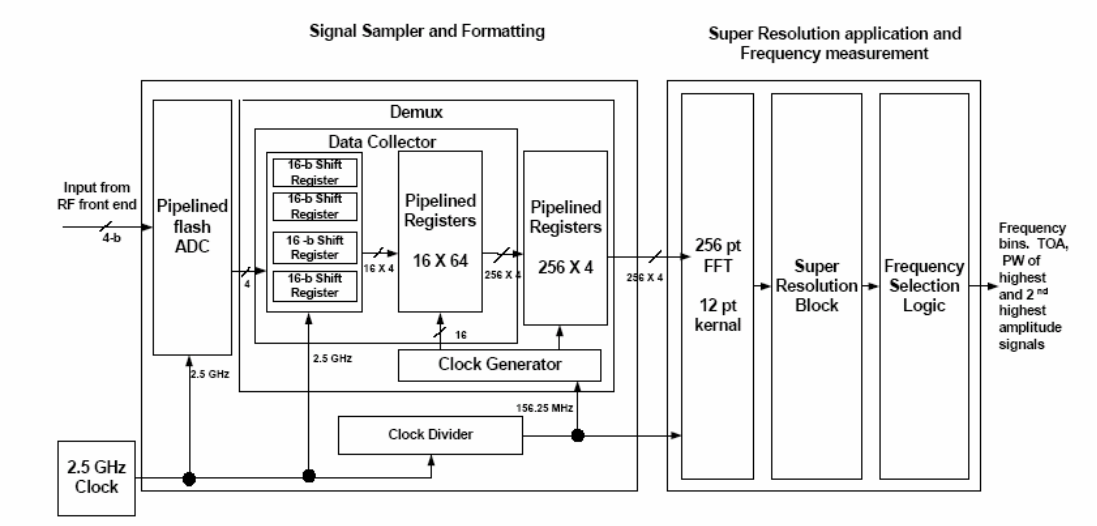


Figure 1-1 Digital Receiver SoC [5]

Then the requirements of ADC for the digital receiver are shown in Table 1-1. To process two simultaneous signals with maximum instantaneous dynamic range (IDR), the digital receiver requires the embedded ADC with high spur-free-dynamic-range (SFDR).

Table 1-1 Initial Requirements of ADC for Digital Receiver

Resolution	4 bits
Technology	130 nm
Power supply	1.2 V
Conversion rate	2.5 GS/s
Bandwidth	1 GHz (125MHz – 1125 MHz)
Power consumption	Low power (< 200mw)
SFDR (for Maximum IDR)	Higher

1.3 Dissertation Research Summary

The research goal is to investigate and design a high-speed, low-power, low-voltage flash CMOS ADC for 1 GHz bandwidth digital receiver on a chip with a high two-signal instantaneous dynamic range (IDR). The research is accomplished by proposed CDC-based pipelined CMOS flash ADC architecture, dynamic offset suppression technique, and circuit optimization. The performance of a 4-b CDC-based pipelined ADC is summarized in Table 1-2.

Table 1-2 Achieved Performance of ADC for the Digital Receiver

Resolution	4 bits	4 bits
Technology	130 nm	130 nm
Power supply	1.2 V	1.2 V
Conversion rate	2.5 GSPS	4 GSPS
Bandwidth	1.25 GHz (9.7 MHz – 1.248 GHz)	2 GHz (62.5 MHz – 1.997 GHz)
Power consumption	7.94 mW	11.08 mW
SFDR @ Nyquist freq. (for Maximum IDR)	26 dB	24 dB

Using a 2.5 GHz sampling rate, the ADC achieves a high spurious-free dynamic range (SFDR) of 26 dB at a Nyquist frequency signal of 1.248 GHz. After the sampling rate increases to 4 GHz, the ADC achieves a SFDR of 24 dB at a Nyquist frequency signal of 1.997 GHz. The proposed ADC is suitable for applications in wideband communication SoC.

1.4 Dissertation Layout

After an overview of ADC fundamentals in chapter 2, where examples and the state-of-the-art implementations are cited, chapter 3 reviews ADCs in the literature and chapter 4 discusses high speed ADCs in wideband receivers. In chapter 5 a new digital clocked comparator (CDC) is proposed and coupled with dynamic offset suppression technique and circuit optimization to improve dynamic performance of ADC. A

CDC-based pipelined flash ADC architecture is then proposed in chapter 6. Chapter 7 presents implementations and experimental results of three CDC-based pipelined flash ADCs with different design requirements in 130 nanometer (nm) digital CMOS process. The CDC-based pipelined ADCs are compared with the ADCs in literature. Chapter 8 presents an application of a 4-b CDC-based pipelined flash ADC in a wideband digital receiver with 2.5 GSPS and 1 GHz bandwidth. A block-level post-design verification flow for the receiver has been established in Cadence analog/mixed-signal (AMS) platform. Interface hardware and down-conversion between ADC and DSP is presented. Conclusions and the future work are discussed in chapter 9.

Chapter 2 ADC FUNDAMENTALS

When an ADC converts analog signals into digital signals, it converts the continuous values into the discrete values both in time and amplitude. The process converting signals from continuous time to discrete time is called sampling with a sampling frequency $f_{sampling}$. $f_{sampling}$ is one of the primary characteristics of ADC and represents the conversion speed of ADC. The other process converting signal amplitude from continuous voltages into discrete voltages is called quantization with a resolution N. N is another primary characteristic and represents the conversion precision of ADC. The sampling usually precedes the quantization in the conversion process of the conventional low-speed ADC designs, while in some high speed ADC designs two actions can be mixed or are commutative [7].

The performance characteristics of ADCs are classified into the static performance and the dynamic performance. The static performance includes offset, gain mismatch, integral non-linearity (INL) and differential non-linearity (DNL) errors, etc. The dynamic performance includes signal to noise ratio (SNR), signal to noise and distortion ratio (SINAD), effective number of bits (ENOB), spurious-free dynamic range (SFDR) and total harmonic distortion (THD), etc. Definitions and principles of primary characteristics and calculation of performance parameters are discussed in the following sections.

2.1 Primary Characteristics

2.1.1 Sampling Frequency

An analog signal is sampled at regularly-spaced time intervals *T*. The discrete samples of $x_a(t)$ are denoted by $x(n) = x_a(nT)$ with $-\infty < n < \infty$ where *x(n)* is the discrete-time sample of the continuous-time analog signal $x_a(t)$ every *T* seconds. The sampling frequency $f_{samping}$ is defined as the reciprocal of the time interval *T*, as $f_{sampling} = 1/T$ and is so called the sampling rate as well with dimensions of samples per second. If a continuous-time sinusoidal signal is $x_a(t) = A\cos(2\pi ft + \theta)$ with $-\infty < t < +\infty$ and $-\infty < f < +\infty$, where *A* is the *amplitude* of the sinusoid, *f* is the signal *frequency* in hertz, and θ is the *phase* in radian, its discrete-time sinusoidal signal can be expressed as $x(n) = A\cos(2\pi fnT + \theta) = A\cos(\omega n + \theta)$ with $\omega = 2\pi f / f_{sampling}$ in radian per sample.

How to select the sampling frequency $f_{samping}$? According to the characteristic of discrete-time sinusoids whose frequencies are separated by an integer multiple of 2π are identical, e.g., $\cos[(\omega_0 + 2\pi)n + \theta] = \cos(\omega_0 n + \theta)$. The sequences of any two discrete-time sinusoids with frequencies only in the range $-\pi \le \omega \le \pi$ are distinct. Thus the analog signals have to be equal or smaller than half of the sampling frequency as

$$-\tfrac{1}{2} f_{sampling} \le f \le \tfrac{1}{2} f_{sampling} \qquad (2\text{-}1)$$

To reconstruct signals back unambiguously without aliasing, the sampling frequency has to be equal or greater than twice of the frequency bandwidth of analog signals. This is Nyquist-Shannon sampling Theorem [8].

2.1.2 Resolution

Quantization converts signal amplitude from continuous voltages into discrete values by dividing a full-scale signal voltage into 2^N-1 sub-range where N is the *resolution* of ADC. The resolution N represents the expected conversion precision of ADC. The effective conversion precision is decreased due to various noises. A primary noise source in data conversion is quantization error.

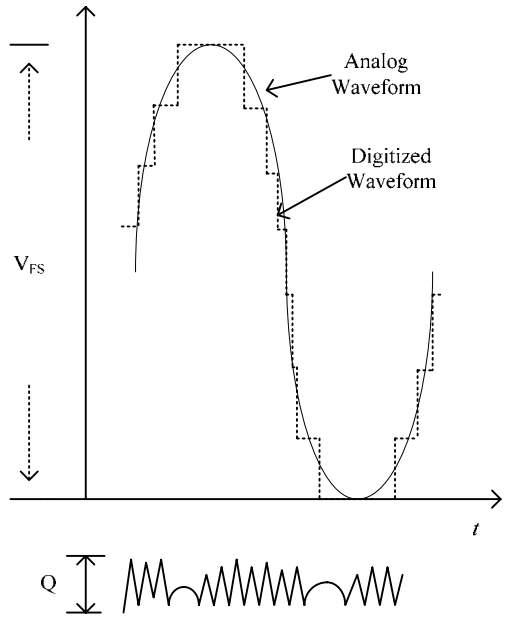


Figure 2-1 $N = 3$ Quantization Steps

Quantization inherently adds noise into digitized signals. Figure 2-1 presents quantization steps for a resolution of $N = 3$. The full-scale amplitude of analog signal is divided into 7 sub-ranges. Any voltage in a sub-range between every two steps is rounded to the closed step. Thus, quantization unavoidably results in a loss of information and is presented as the quantization noise.

Quantization error Q is one step size V_{LSB} and generally considered as a random

variable with a uniform distribution. The error density function $f_Q(x)$ is $\frac{1}{V_{LSB}}$ in the range $[-\frac{V_{LSB}}{2}, \frac{V_{LSB}}{2}]$ as shown in Figure 2-2.

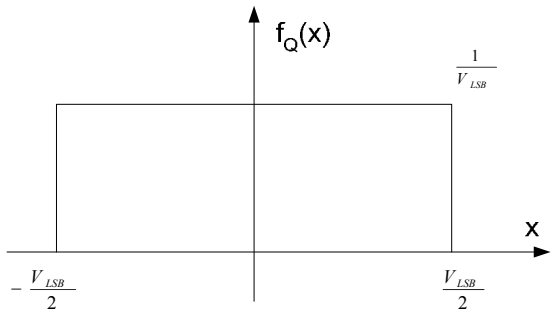


Figure 2-2 Uniform Distribution of Quantization Error

To approximate the quantization noise, the signal-to-noise ratio (SNR) is calculated by comparing the signal power over the quantization noise power. The root-mean-square (*rms*) value of the quantization noise is $V_Q rms = [\int_{-\frac{V_{LSB}}{2}}^{\frac{V_{LSB}}{2}} x^2 \frac{1}{V_{LSB}} dx]^{\frac{1}{2}} = \frac{V_{LSB}}{\sqrt{12}}$ where $V_{LSB} = \frac{V_{FS}}{2^N}$.

For a sinusoidal signal $\frac{V_{FS}}{2}\cos(\Omega t + \theta)$ with a full-scale voltage, the *rms* value is $V_{in} rms = \left[\frac{1}{2\pi} \cdot \int_0^{2\pi} (\frac{V_{FS}}{2} \cdot \cos(\Omega t + \theta)^2 d\Omega t\right]^{\frac{1}{2}} = \frac{V_{FS}}{2\sqrt{2}}$. The signal-to-noise ratio (SNR) of ADC is the ratio of the *rms* value of signal to the *rms* value of the quantization noise.

$$SNR = 20\log_{10}(\frac{V_{in} rms}{V_Q rms}) = 6.02N + 1.76(dB) \qquad (2\text{-}2)$$

Thus the ideal SNR of an *N*-b ADC is $6.02N + 1.76$ dB as shown in Equ. (2-2). For example, a 4-b ADC has an ideal SNR of 25.84 dB.

2.2 Performance Characteristics

Performance characteristics of ADCs [9, 10] presented in this section aims at giving an evaluation metric of ADCs.

2.2.1 Static Performance

2.2.1.1 Offset and Gain Error

Offset is a constant difference between the actual finite resolution characteristic and the ideal finite resolution characteristic measured at any vertical jump. Gain Error is the difference between the actual finite resolution and an infinite resolution characteristic measured at the rightmost vertical jump. It is proportional to the magnitude of ADC input voltage. In conventional comparison of ADC performance, the sampling speed is assumed for all input voltage due to the linearity of circuits, however, a high input voltage results in a faster response than does a low input voltage which eventually results in an ADC gain error.

2.2.1.2 Integral Non-Linearity and Differential Non-Linearity

Integral nonlinearity (INL) is the maximum difference between the actual finite resolution characteristic and the ideal finite resolution characteristic measured vertically. It can be expressed as a percentage of the full scale range or in terms of the least significant bit (LSB).

Differential Nonlinearity (DNL) is a measure of the separation between adjacent levels measured at each vertical jump. It measures bit-to-bit deviations from ideal output steps

and can be expressed in terms of the LSB as shown in Equ. (2-3).

$$DNL(k) = \frac{V_{k+1}^{'} - V_{k}^{'}}{V_{LSB}} - 1LSB \tag{2-3}$$

Equ. (2-3) is used to calculate the DNL for each quantization level and VLSB is one voltage step size and equal to $V_{FS}/(2^N - 2)$ in flash ADCs. V'k+1 and V'k are adjacent actual voltage levels. The maximum DNL characterizes the differential nonlinearity of the ADCs.

2.2.2 Dynamic Performance

2.2.2.1 Signal-to-Noise Ratio

The signal-to-noise ratio (SNR) is the ratio of the signal power to the total noise power at the output usually measured for a sinusoidal input. It is expressed in decibel (dB) as the following equation [11]:

$$SNR = 20\log_{10}\left(\frac{A_{rms,signal}}{A_{rms,noise}}\right) \tag{2-4}$$

where $A_{rms,signal}$ and $A_{rms,\ noise}$ are the root mean square of the amplitude for the signal and noise, respectively. The SNR can also be calculated as the signal fundamental spectrum in dB minus the sum of all the noise spectra excluding the significant harmonics.

2.2.2.2 Signal-to-Noise-and-Distortion Ratio

The signal-to-noise-and-distortion ratio (SNDR) is the ratio of the signal power to the

total noise power and harmonics power at the output when inputting a sinusoid. Mathematically the SNDR is formalized as the following [12]

$$SNDR = 20\log_{10}\left(\frac{A_{rms,signal}}{A_{rms,noise+harmonics}}\right) \tag{2-5}$$

where Arms, signal and Arms, noise+harmonics are the *rms* of the signal amplitude and the *rms* of all the noise amplitudes and all the harmonic amplitudes. According to the SNR in Equ. (2-2), the performance parameters SNR and SNDR are both up-bounded at 6.02N+1.78 dB.

2.2.2.3 Effective Number of Bits

The effective number of bits (ENOB) is defined by the following equation [15]:

$$ENOB = \frac{SNDR - 1.78dB}{6.02} \tag{2-6}$$

In the Nyquist ADC, the ENOB is smaller than the resolution *N* of ADC, while with noise-shaping technique, the over-sampling ADC may achieve an ENOB higher than the resolution *N*. The ADC architectures are reviewed and discussed in the next chapter.

2.2.2.4 Spurious-Free Dynamic Range

The spurious-free dynamic range (SFDR) is the ratio of the signal power to the largest harmonic power, or the power of the highest spur if the power of the highest spur is greater than the largest harmonic power. It can be expressed as

$$SFDR = 20\log_{10}\left(\frac{A_{rms,signal}}{A_{rms,2^{nd}\ harmonic\ /\ highestSpur}}\right) \tag{2-7}$$

The SFDR performance shows the dynamic range of an ADC. In [13] and [14], an empirical equation for the SFDR was derived as

$$SFDR \approx 9N - c \tag{2-8}$$

where N is the resolution of the ADC, c is the offset ranges from 0 for low resolution and 6 for high resolution [24].

2.2.2.5 Total Harmonic Distortion

The total harmonic distortion (THD) is the ratio of the total significant harmonic power over the signal power. The THD can be expressed as

$$THD = 20\log_{10}\left(\frac{A_{rms,harmonics}}{A_{rms,signal}}\right) \tag{2-9}$$

Chapter 3 ADC LITERATURE REVIEW

The literature review is intended to provide information to understand the context of this research. Following a brief overview of ADC architectures, high speed flash ADCs are surveyed and the performances of state-of-the-art flash ADCs are discussed. At the end of this chapter, flash ADCs using inverter comparator configuration for SoC applications are discussed.

3.1 Architecture Overview

As the primary characteristics, the precision and speed in data conversion determine the selection of ADC architectures for a specific application. ADC precision and speed, in terms of resolution and sampling frequency, is shown in Figure 3-1 [16]. Sigma-Delta ADCs have the highest resolution but the lowest sampling frequency; flash ADCs have he highest sampling frequency but the lowest resolution. The resolution and the sampling frequency for successive approximation register (SAR) ADCs and pipelined ADCs are in the between.

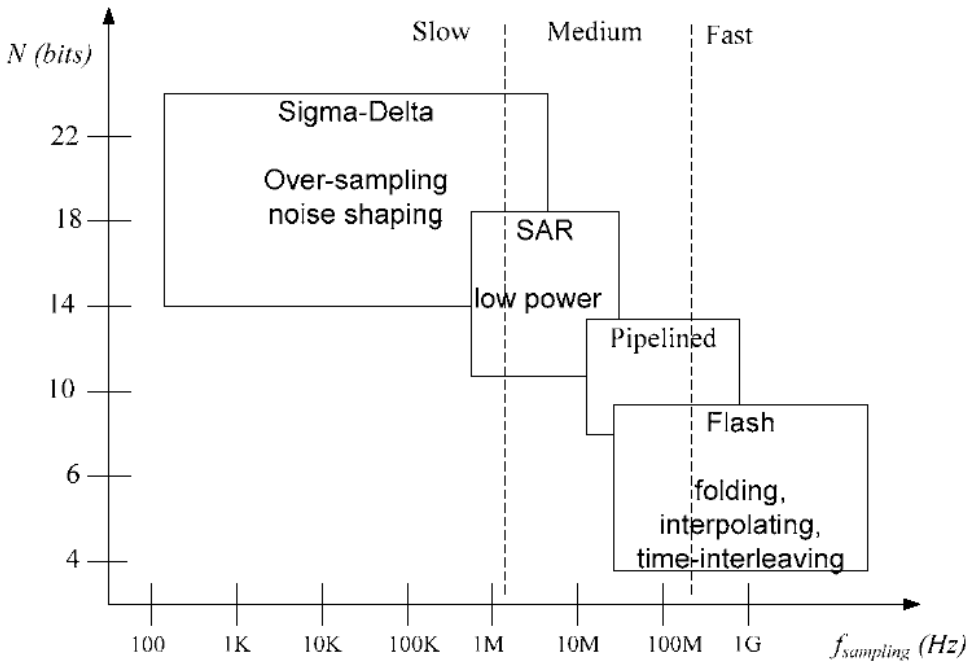


Figure 3-1 Resolution N vs. Sampling Frequency $f_{sampling}$

The primary characteristics of the ADC architectures determine their applications. The Sigma-Delta ADCs adopt over-sampling and noise shaping techniques to increase the SNR by pushing low-frequency noise to higher frequencies which is out of the interest bandwidth. The Sigma-Delta ADCs have features of low-bandwidth, high-resolution and are particularly suitable for applications of precision measurement. The SAR ADCs use one comparator over many cycles to conduct its conversion. They are often used at lower speed, high resolution applications like industrial control and battery-powered applications. The pipelined ADCs divide the conversion task into several consecutive stages to achieve high resolution than the flash ADCs. The pipelined ADCs are often applied to high speed, low resolution applications such as Video, HDTV, and Medical & CCD Imaging. The flash ADCs convert the signal in one cycle by parallelizing all comparators to achieve fast

conversion speed. They are often used in high speed applications like high-density disk drives, wireless communications, wideband satellite receivers, etc. In general, power consumption and silicon area of the flash ADCs increases with the increase of signal bandwidth.

3.2 High Speed ADC in Literature

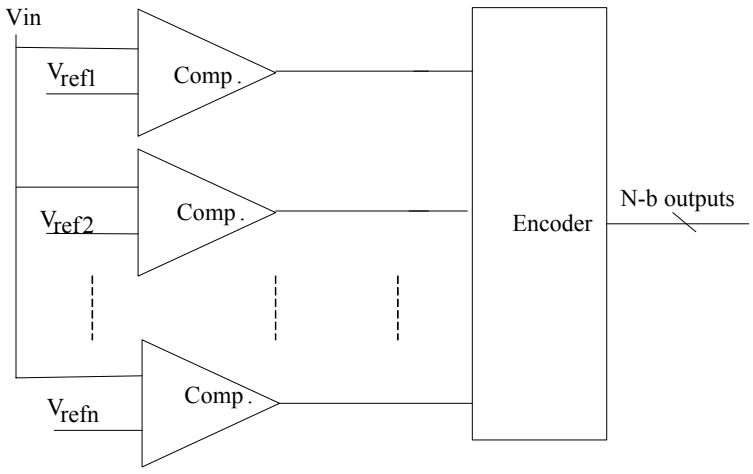


Figure 3-2 Flash ADC Architecture

A general architecture of flash ADC is presented in Figure 3-2. In an *N*-bit flash ADC, 2^N-1 reference voltages and comparators are used to convert the analog input signal into a thermometer digital output signal. Commonly, the reference voltages are provided by a 2^N-1-resistor ladder and the thermometer-code output is converted into a binary code by a thermometer-to-binary encoder.

A full flash architecture, in principle, does not need an explicit front-end sample-and-hold circuit and its performance is determined primarily by its constituent

comparators [9]. Since comparators do not require linear amplification and typically achieve a higher speed than sample-and-hold amplifiers (SHA), flash ADCs can operate faster than those that demand front-end SHA [9]. Since the number of comparators grows exponentially with the resolution, these ADCs have excessively large input capacitance, power and area consumption, and different comparator offset voltages. In addition, lack of a front-end sample-and-hold amplifier makes the converter vulnerable to sparkles and slew-dependent sampling points.

Full flash ADCs employ parallelism and "distributed" sampling to achieve a high conversion speed with a simple architecture, but the resolution is limited when its sampling frequency reaches several giga-hertz [35]. A few of circuit techniques have been proposed to improve the resolution while maintaining a one-step conversion without using sample-and-hold circuits. Folding and interpolation techniques were proposed to increase the effective input bandwidth by reducing the number of comparators but the architecture becomes complicated [17-23] [36]. Averaging technique was proposed to improve the performance of the resolution and the DNL error by suppressing the output offsets of comparators by scarifying the input signal dynamic range due to dummy comparators [1] [3] [24-26] [29-31]. Time-interleaved technique was proposed to increase the sampling frequency by parallelizing ADCs but the high nonlinearity mismatch noise requires an elaborate digital calibration system [32-34].

Table 3-1 Comparison to State-of-the-art GHz Flash ADCs

Ref. (year)	Primary Characteristics				Performance Parameters						Style
	Res. (bits)	**GSPS (GHz)**	**Tech. (μm)**	Vdd (V)	ERBW (MHz)	SNDR (dB)	SFDR (dB)	**Power (W)**	**Area (mm^2)**	Cin (pF)	
[35] '03	6	**1.3**	**0.25**	1.8	500	32	41	**0.6**	**0.12**	--	full parallel
[19] '01	6	**1.1**	**0.35**	3.3	450	29	40	**0.3**	**0.35**	--	3 avg. 2 interp.
[23] '05	6	**1.2**	**0.13**	1.5	700	32.8	42	**0.16**	**0.12**	0.4	capacitive interp.
[1] '01	6	**1.3**	**0.35**	3.3	650	31.8	37.3	**0.545**	**0.8**	1	avg. cxt. S/H
[3] '02	6	**1.6**	**0.18**	1.95	660	31.9	--	**0.34**	**--**	--	optimized avg.
[34] '05	6	**2**	**0.18**	1.8	941	30	35.5	**0.31**	**0.5**	1	avg./interp. 2-ch.interl.
[36] '04	8	**1.6**	**0.18**	1.8	1000	46	56	**0.77**	**3.6**	1.8	fol./interp. /enh.calib.
[33] '05	8	**2**	**0.18**	1.8 /3.3	1000	44	53	**1.2**	**--**	--	folding/ interleavee
[32] '03	8	**4**	**0.35**	3.3	1000	38	--	**4.6**	**28.4**	2	8-chanel time-int.

The performances of state-of-the-art GHz flash ADCs are compared in Table 3-1. The effective resolution bandwidth (ERBW) is limited by 1000 MHz (1GHz), although the sampling rate reaches 4 GHz. The input capacitance, power and area consumption increase as the ERBW increases.

3.3 Flash ADCs Using Inverter Comparator for SoC

In recent years, with technology processes continuously scaling-down and SoC technique becoming feasible, high-performance low-voltage ADCs are in demand. The

low-voltage ADCs with comparators in a simple inverter configuration particularly suitable for SoC are reported.

Segura et al. in 1998 [35] suggested that an inverter could be used as an analog comparator. Tangel in 1999 [36] first proposed using threshold inverter quantization (TIQ) technique for comparator design in CMOS flash ADCs. An 1-GSPS CMOS flash ADCs for SoC application was reported in 2001 [16][37]. Thereafter, a number of inverter-based ADCs with emphasis on optimization of comparators were reported [38]-[43]. Some of their performances are shown in Table 3-2.

Table 3-2 Comparison to Flash ADCs with Inverter Comparator

Ref. (year)	Primary Characteristics				Performance Parameters					Sizing Design
	Res. (bits)	**GSPS (GHz)**	**Tech. (μm)**	Vdd (V)	INL /DNL (LSB)	ERBW (MHz)	SNDR /SFDR (dB)	**Power (mW)**	**Area (mm^2)**	
[37] '01	6	**1**	**0.25**	2.5	0.26 /0.23	--	--	**66.7**	**0.013**	RSV/ SSV[16]
[39]	6	**2.7**	**0.07**	0.7	0.0287 /0.0502	--	--	**4.952**	**0.029**	SSV
[39]	8	**2.0**	**0.07**	0.7	0.1581 /0.2996	--	--	**19.094**	**0.109**	SSV
[40] '05	4	**2.5**	**0.13**	1.2	--	1000	-- / 9	**23.1**	**0.0464**	random
[41] '07	4	**2.5**	**0.13**	1.2	--	1248	4.976 /19.81	**7.9**	**--**	Dyn.OS. Suppr.

In [37] and [39], the ADCs were reported with emphasis on static performance of INL and DNL. Using random size variation (RSV) and systematic size variation (SSV) techniques to optimize the comparators improves the static performance of ADCs. The ADC in [40] was designed with random sizing method and achieved a SFDR of 9 dB with

an input signal frequency of 1 GHz. The ADC in [41] was designed by consideration of dynamic offset suppression and achieved a SFDR of 19.81 dB with an input signal frequency of 1.25 GHz. Their ADC performances are summarized in Table 3-2.

Flash ADCs with inverter comparator in Table 3-2, compared with flash ADCs in Table 3-1, have comparatively higher sampling rate, low power and area consumption, and using smaller feature size technologies.

Chapter 4 HIGH-SPEED ADCS IN WIDEBAND RECEIVERS

In wideband communication systems, an ADC in receivers digitizes several channels of strong and weak signals simultaneously. A digital signal processor (DSP) subsequently detects each channel. First, the input dynamic range is an important performance parameter for communication receivers. The ADC is desired to have high-speed data conversion and high input dynamic range [45]. Usually, the SFDR of ADCs limit wideband receiver sensitivity, not the SNR [44][46]. Therefore a design objective of ADC for wideband communications is to increase the SFDR by lowering the spur and noise floor therefore the weak legitimate signals can be detected. Secondly, the wide bandwidth requires a high conversion rate in the ADC design by Nyquist theorem. The flash ADC architecture is commonly used without external sample-and-hold (S/H) and has distributed sampling on high-speed comparators for giga-hertz input signals. The sparkles in thermometer code, meta-stability, and signal slew rate limitation will generate spurious code and odd harmonics in the digital output [9][51]. Engineers have to consider all these issues for a good ADC design in wideband communication systems.

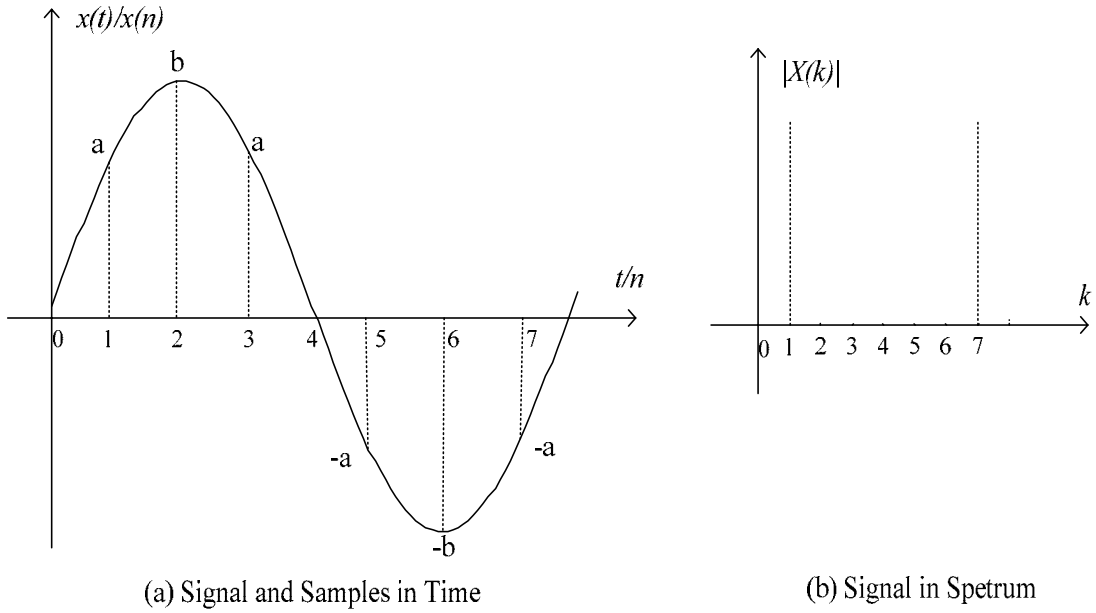
4.1 Spurs in Spectrum

The spurious-free dynamic range (SFDR) must be sufficiently high so that weak signals are not covered in the spurious-floor. The SFDR of the ADC is defined as the difference in decibels (dB) between the full-scale fundamental and the maximum spurious tone including harmonics in the output spectrum. The signal-to-noise ratio (SNR) is relatively less important here [46].

To understand the spurs in the output spectrum, we look at the discrete Fourier transform (DFT) as the following:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \tag{4.1}$$

where x(n) is the sampled output of ADC, N is the total number of samples, and $X(k)$ is the output spectrum power in real-imaginary domain with k ranging from 1 to N. Ideally, $X(k)$ is zero at the non-signal points and non-zero at the signal point. This is illustrated by an example of 1-GSPS ADC at an input signal frequency of 125 MHz signal using an 8-point DFT. Both time and frequency domain data are depicted in Figure 4-1.



(a) Signal and Samples in Time (b) Signal in Spetrum

Figure 4-1 Time and frequency domain data using an 8-point DFT

For a periodic signal, ideal samples are [0, a, b, a, 0, -a, -b, -a] as shown Figure 4-1 (a). We take those samples into Equ.(4.1) and get Equ.(4.2) with $W^{nk} = e^{-j2\pi nk/N}$ for simple expression.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ 1 & W^2 & W^4 & W^6 & W^8 & W^{10} & W^{12} & W^{14} \\ 1 & W^3 & W^6 & W^9 & W^{12} & W^{15} & W^{18} & W^{21} \\ 1 & W^4 & W^8 & W^{12} & W^{16} & W^{20} & W^{24} & W^{28} \\ 1 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} & W^{30} & W^{35} \\ 1 & W^6 & W^{12} & W^{18} & W^{24} & W^{30} & W^{36} & W^{42} \\ 1 & W^7 & W^{14} & W^{21} & W^{28} & W^{35} & W^{42} & W^{49} \end{bmatrix} \begin{bmatrix} x(0)=0 \\ x(1)=a \\ x(2)=b \\ x(3)=a \\ x(4)=0 \\ x(5)=-a \\ x(6)=-b \\ x(7)=-a \end{bmatrix} \quad (4.2)$$

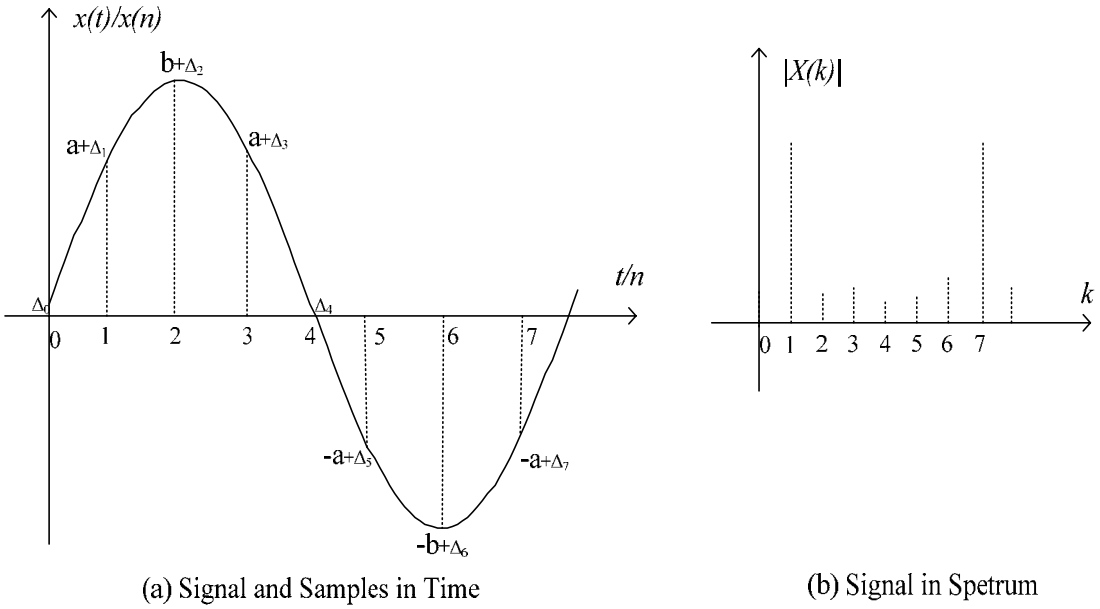
Because of symmetry and periodicity property of DFT, which are $W_N^{nk} = -W_N^{nk+N/2}$ and $W_N^{nk} = W_N^{nk+N}$, Equ.(4.2) is simplified to Equ.(4.3).

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 & -1 & -W^1 & -W^2 & -W^3 \\ 1 & W^2 & -1 & -W^2 & 1 & W^2 & -1 & -W^2 \\ 1 & W^3 & -W^2 & W^1 & -1 & -W^3 & W^2 & -W^1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -W^1 & W^2 & -W^3 & -1 & W^1 & -W^2 & W^3 \\ 1 & -W^2 & -1 & W^2 & 1 & -W^2 & -1 & W^2 \\ 1 & -W^3 & -W^2 & -W^1 & -1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ a \\ 0 \\ -a \\ -b \\ -a \end{bmatrix} \tag{4.3}$$

The values of *X(k)* are calculated in Equ.(4.4). *X(1)* and *X(7)* are signal-point and symmetry, while the rest are non-signal points and equal to zero, shown in Equ.(4.4) and in Figure 4-1 (b).

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 0 \\ 2a(W^1 + W^3) + 2bW^2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -2a(W^1 + W^3) - 2bW^2 \end{bmatrix} \tag{4.4}$$

For non-ideal case of sampling as shown in Figure 4-2 (a), the samples can be [Δ_0, a+Δ_1, b+Δ_2, a+Δ_3, Δ_4, -a+Δ_5, -b+Δ_6, -a+Δ_7], where Δ_i is an offset from its ideal value resulting from error sources, sparkles or meta-stability, for instances.



(a) Signal and Samples in Time

(b) Signal in Spetrum

Figure 4-2 8-point DFT with Spurs

Putting the offset samples into Eqn.(4.3) as shown in Equ.(4.5), we can see that *X(k)* at non-signal points are not zero and completely determined by the offsets, while *X(k)* at signal points are modified by the offset in the samples, shown in Equ.(4.6).

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 & -1 & -W^1 & -W^2 & -W^3 \\ 1 & W^2 & -1 & -W^2 & 1 & W^2 & -1 & -W^2 \\ 1 & W^3 & -W^2 & W^1 & -1 & -W^3 & W^2 & -W^1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -W^1 & W^2 & -W^3 & -1 & W^1 & -W^2 & W^3 \\ 1 & -W^2 & -1 & W^2 & 1 & -W^2 & -1 & W^2 \\ 1 & -W^3 & -W^2 & -W^1 & -1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} \Delta_0 \\ a+\Delta_1 \\ b+\Delta_2 \\ a+\Delta_3 \\ \Delta_4 \\ -a+\Delta_5 \\ -b+\Delta_6 \\ -a+\Delta_7 \end{bmatrix} \tag{4.5}$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} \Delta_0+\Delta_1+\Delta_2+\Delta_3+\Delta_4+\Delta_5+\Delta_6+\Delta_7 \\ 2a(W^1+W^3)+2bW^2+(\Delta_0-\Delta_4)+(\Delta_1-\Delta_5)W^1+(\Delta_2-\Delta_6)W^2+(\Delta_3-\Delta_7)W^3 \\ (\Delta_0-\Delta_2+\Delta_4-\Delta_6)+(\Delta_1-\Delta_3+\Delta_5-\Delta_7)W^2 \\ (\Delta_0-\Delta_4)+(\Delta_3-\Delta_7)W^1+(\Delta_6-\Delta_2)W^2+(\Delta_1-\Delta_5)W^3 \\ \Delta_0-\Delta_1+\Delta_2-\Delta_3+\Delta_4-\Delta_5+\Delta_6-\Delta_7 \\ (\Delta_0-\Delta_4)+(\Delta_5-\Delta_1)W^1+(\Delta_2-\Delta_6)W^2+(\Delta_7-\Delta_3)W^3 \\ (\Delta_0-\Delta_2+\Delta_4-\Delta_6)-(\Delta_1-\Delta_3+\Delta_5-\Delta_7)W^2 \\ -2a(W^1+W^3)-2bW^2+(\Delta_0-\Delta_4)+(\Delta_7-\Delta_3)W^1+(\Delta_6-\Delta_2)W^2+(\Delta_5-\Delta_1)W^3 \end{bmatrix} \quad (4.6)$$

Those non-zeros *X(k)* at non-signal points appear as the spurs in spectrum domain as shown in Figure 4-2 (b).

To lower the spurious-floor in ADC design, the offsets (Δ_i) need to be suppressed. The offsets Δ_i can be static offsets, dynamic offsets or the sum of them. They may be characterized as random or mutual correlated. The random offset of Δ_i is difficult to control and only can be suppressed, while the correlated offset of Δ_i may even be eliminated in certain cases.

In high-speed data conversion, a major source of errors is dynamic offsets caused by mismatches in sampling switches and sampling capacitance values [52]. The dynamic offsets can be suppressed or even eliminated in ADC design.

4.2 Signal Slew Rate Limitation

In the flash conversion without an external S/H circuit, the signal slew rate limits the ADC speed and resolution in a given technology. When the maximum slew rate of the analog input is close to the clock transition rate, the logic output can be different from the expected value because the input signal still influences the output of comparators during the time between the clock latch turned on and the input signal locked off. This phenomenon introduces odd harmonics because it occurs for both negative and positive slopes [54].

This error can be lowered when the clock transition rates are sufficiently higher than the maximum slew rate of the analog input. On a large chip, this requires careful clocking distribution with particular attention to their loading [9].

Another concept related to signal slew rate is the clock jitter, or called aperture jitter. The maximum tolerable jitter determines the ADC's resolution and speed [47]-[50]. For a full-scale analog input $V_{in} = {}^{V_{fs}}\!/_{2} \sin 2\pi ft$, whose maximum rate of change is $\pi \cdot f \cdot V_{fs}$. The maximum tolerable jitter for 1-LSB is

$$dt_{MAX} = \frac{1}{\pi \cdot f \cdot 2^{N}} \tag{4.7}$$

where dt_{MAX} represents the maximum clock jitter and N is the converter's resolution.

4.3 Sparkles and Meta-stability

Sparkles and meta-stability are general phenomena in high-speed ADCs, which result, in grossly incorrect digital output codes. The sparkle phenomenon is a ONE above ZERO caused by timing mismatch among comparators. An important effect resulting from the lack of a S/H in flash ADC is the sparkles (or bubbles) in the thermometer code [51][53]. Meta-stability is a phenomenon associated with binary digital logic system. A meta-stable state occurs when an indeterminate state between logic '1' and logic '0' is latched at the clock edge. It can happen by a comparator with slow regenerative time or using flip-flops for synchronization of signals [51].

There are digital encoding schemes developed to suppress the sparkles and meta-stability errors. But in high speed flash converters Gray encoding is used as an intermediate step between thermometer and binary codes to suppress two potential errors [9]. The error of meta-stable state can be suppressed because one meta-stable state of a comparator is fed into no more than one input in Gray coding. The error of sparkles is reduced because the accuracy of the Gray code degrades gradually as more sparkles appear in the thermometer code.

A 3-bit Gray code example is used to illustrate the above points as shown in Figure 4-3 [9]. From the correspondence, the output $G_3G_2G_1$ can be expressed in terms of the thermometer code as follows:

$$G_1 = T_1 \overline{T_3} + T_5 \overline{T_7} \tag{4.8}$$

$$G_2 = T_2 \overline{T_6} \tag{4.9}$$

$$G_3 = T_4 \tag{4.10}$$

The thermometer code T_i appears in only one expression and hence no signal is split and the effect of meta-stable states can be reduced as shown in Figure 4-4. With pipelining, the time for regeneration is increased to reduce logic meta-stabilities.

Thermometer	Gray	Binary
$T_1T_2T_3T_4T_5T_6T_7$	$G_3G_2G_1$	$B_3B_2B_1$
0 0 0 0 0 0 0	0 0 0	0 0 0
1 0 0 0 0 0 0	0 0 1	0 0 1
1 1 0 0 0 0 0	0 1 1	0 1 0
1 1 1 0 0 0 0	0 1 0	0 1 1
1 1 1 1 0 0 0	1 1 0	1 0 0
1 1 1 1 1 0 0	1 1 1	1 0 1
1 1 1 1 1 1 0	1 0 1	1 1 0
1 1 1 1 1 1 1	1 0 0	1 1 1

Figure 4-3 Correspondence among Thermometer, Gray and Binary Codes

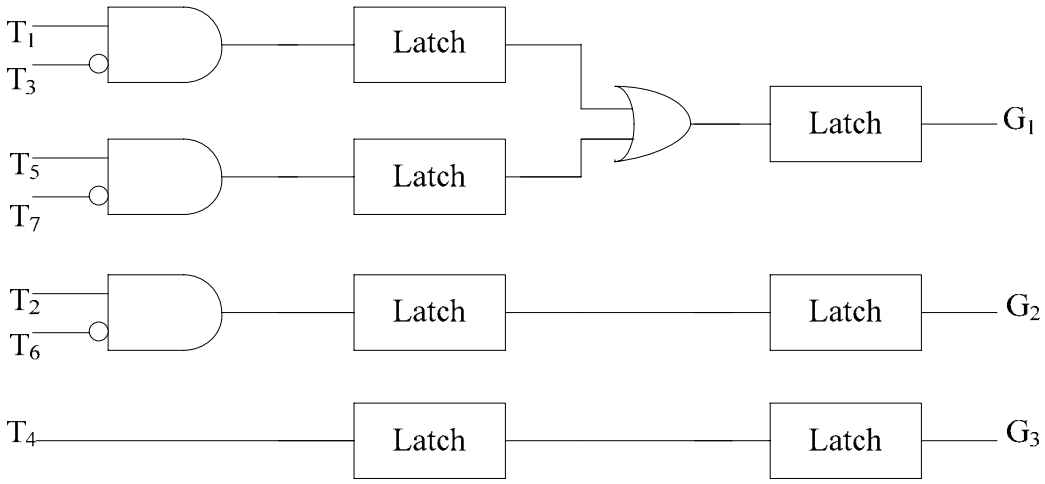


Figure 4-4 Pipelined Thermometer-to-Gray Encoder

Figure 4-5 illustrates the Gray encoding suppressing the effect of various sparkles, compared with direct binary encoding. The Gray output remains a reasonable approximation of the sampled value with the number of sparkles increasing.

	Thermometer Code	Gray Code	Equivalent Decimal Output	Direct Binary Encoding Decimal Output
No Sparkle:	1110000	010	3	3
One Sparkle:	1101000	111	5	4
Two Sparkles:	1100100	011	2	5
Three Sparkles:	1100010	001	1	6
Four Sparkles:	1100001	011	2	7

Figure 4-5 Comparison of Gray and Direct Binary Encoding at Various Sparkles

Although the Gray encoding can be used to suppress the errors by meta-stability and sparkles, the performance of flash conversion is determined primarily by the speed and accuracy of the comparators.

Chapter 5 CLOCKED DIGITAL COMPARATOR

5.1 Introduction

Comparator is the constituent part of ADCs with full-flash architecture. The overall performance of flash ADCs is limited by the performance of their comparators for the multi-bit flash ADCs are comparators parallelized, which are one-bit converters. However, CMOS flash ADCs suffer greatly from offsets in the comparators because of device mismatches and clock timing mismatches.

Generally, high-speed flash ADCs use the differential amplifier structure in pre-amplifier and comparator design. There are two types of offsets in a differential comparator. One is a static and random offset from device mismatches, which is amplified into a larger offset in the output. The other is a dynamic offset at the output of comparator arising from clock switching in the regenerative latch. The dynamic performance of ADC is degraded greatly by the offsets of differential comparators.

In the state-of-the-art flash ADCs with the speed less than 2-GHz sampling frequency reported in Table 3-1, averaging technique is used to smooth out the random mismatch across the differential comparator and improve the SNR in flash ADCs [1][3][19][34]. For

strongest averaging [44], one third dummy amplifiers are added and extra reference voltage increases the difficulty of the high-speed differential comparator design due to the stringent effective input voltage range in scaling-down CMOS processes. Besides, averaging method reduces amplifier bandwidth and consumes extra power due to dummy amplifiers [44].

In design of monolithic flash ADCs using feature size less than 130-nm CMOS processes with the sampling speed over 2-GHz and the effective signal bandwidth over 1000-MHz, the differential comparator design is challenged. More importantly, the dynamic offsets from clock timing mismatches between parallelizing comparators become more significant and apparently effect to the ADC performance at over 1-GHz frequency signal.

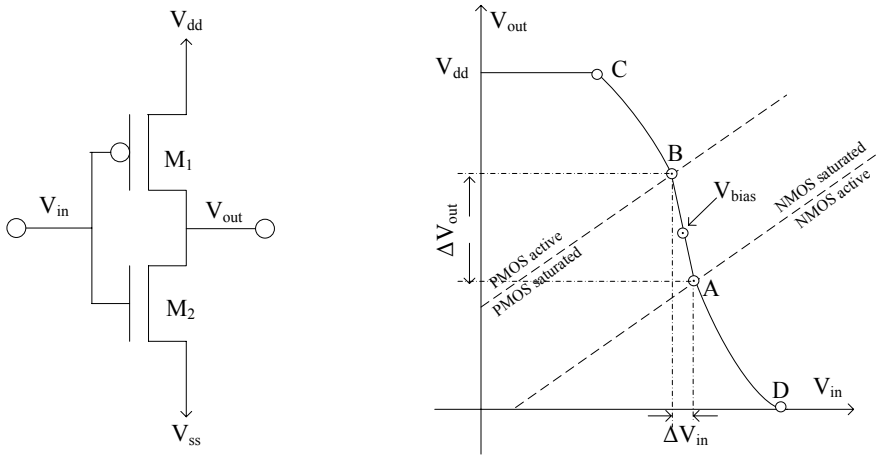
As reviewed in Chapter 3, the comparator with inverter configuration was suggested for scaling-down CMOS processes. However, the relative researches reported were focusing on the static performance, did not consider the dynamic performance of the ADCs for high frequency signals.

In this research, we propose a high-speed clocked digital comparator (CDC) with inverter configuration, coupled with a dynamic offset suppression and an optimum implementation method for high-speed wideband CMOS flash ADCs. With a simple inverter configuration, the CDC has less device mismatch issues and lower device linearity requirements. The dynamic offset arising from clock timing mismatches is more apparent in the CDC design, than the random static offset from device mismatches, which is dealt with in this research by dynamic offset suppression technique and optimization design for wideband communications without additional hardware.

A brief review of inverter amplifier is presented in Section 5.2. CDC and dynamic offset suppression is presented in Section 5.3. Section 5.4 presents procedures to implement the optimal CDC design, and discuss optimization theory and experimental results. The performance of CDC is summarized in Section 5.5.

5.2 Inverter Amplifier

Figure 5-1(a) illustrates an inverter amplifier where M_1 and M_2 are PMOS and NMOS transistors. The operating regions of the inverter amplifier for different input voltages are shown on the voltage transfer function characteristics of Figure 5-1(b). Both the transistors are in saturation region between points A and B on the output curve with a bias voltage V_{bias}. Since both transistors are being driven by v_{in} and in saturation, the inverter amplifier has the highest gain.



(a) Inverter Amplifier (b) Voltage-Transfer Characteristics

Figure 5-1 Inverter Amplifier with Voltage-Transfer Characteristics

The PMOS M1 is in the saturation region when $v_{DS1} \geq v_{SG1} - \left|V_{Tp}\right|$, then we have

$$v_{OUT} \leq v_{IN} + \left|V_{Tp}\right| \tag{5.1}$$

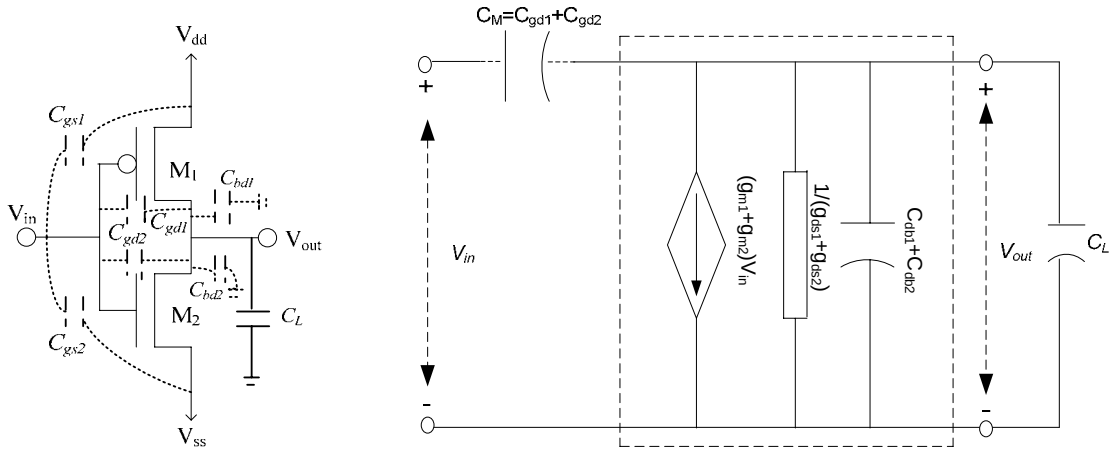
The NMOS M2 is in the saturation region when $v_{DS2} \geq v_{SG2} - V_{Tn}$, then we have

$$v_{OUT} \geq v_{IN} - V_{Tn} \tag{5.2}$$

For both transistors in saturation, the output range ΔV_{out} can be derivate by subtracting Equ.(5.2) from Equ.(5.1) as

$$\Delta V_{out} = \left|V_{Tp}\right| + V_{Tn} \tag{5.3}$$

Figure 5-2 illustrates the small-signal characteristics of inverter amplifier. The parasitic capacitances are presented in Figure 5-2(a) and Figure 5-2(b) presents the small-signal model of (a).



(a) Inverter Amplifier with Parasitic Capacitances

(b) Small-Signal Model of (a)

Figure 5-2 Small-Signal Characteristics of Inverter Amplifier

The largest small-signal voltage gain occurs when both transistors are saturated which can be expressed by

$$A_0 = \frac{-(g_{m1} + g_{m2})}{g_{ds1} + g_{ds2}} = \frac{\dfrac{2}{V_{bias} - V_{Tn}} + \dfrac{2}{V_{dd} - V_{bias} - |V_{Tp}|}}{(\lambda_n + \lambda_p)} \tag{5.4}$$

From Equ.(5.4), the value of A_0 is process-dependent and varies in a certain range with V_{bais}. For IBM 130 nm CMOS process, A_0 is around 16 and slight changes with V_{bais} from 0.4v to 0.75v.

The input range for both transistors in saturation is Δv_{in} for V_{bias} which can be expressed by

$$\Delta v_{in} = \frac{(\lambda_n + \lambda_p)(|V_{Tp}| + V_{Tn})}{\left(\dfrac{2}{V_{bias} - V_{Tn}} + \dfrac{2}{V_{dd} - V_{bias} - |V_{Tp}|}\right)} \tag{5.5}$$

For IBM 130 nm CMOS process, if $|V_{Tp}| + V_{Tn}$ approximate to 0.4V, Δv_{in} is around 25 mV.

The -3dB frequency response can be expressed as

$$\omega_0 = \frac{g_{ds1} + g_{ds2}}{C_{out}} = (\lambda_n + \lambda_p)\frac{I_{ds}}{C_{out}} \tag{5.6}$$

with $C_{out} = C_{gds1} + C_{gds2} + C_{bd1} + C_{bd2} + C_L$, illustrated in Figure 5-2(b).

5.3 Clocked Digital Comparator with Dynamic Offset Suppression

Figure 5-3 shows the structure of clocked digital comparators (CDCs) designed in the n-bit flash ADC, in which the 2^n-1 comparators have the same structure, but are sized

differently. A comparator consists of two cascaded CMOS inverters and a digital CMOS switch. The first inverter is a quantization component. The second inverter is a component which sharpens and balances the quantized output. The digital switch is a digitization component which makes the comparator perform a sampling operation.

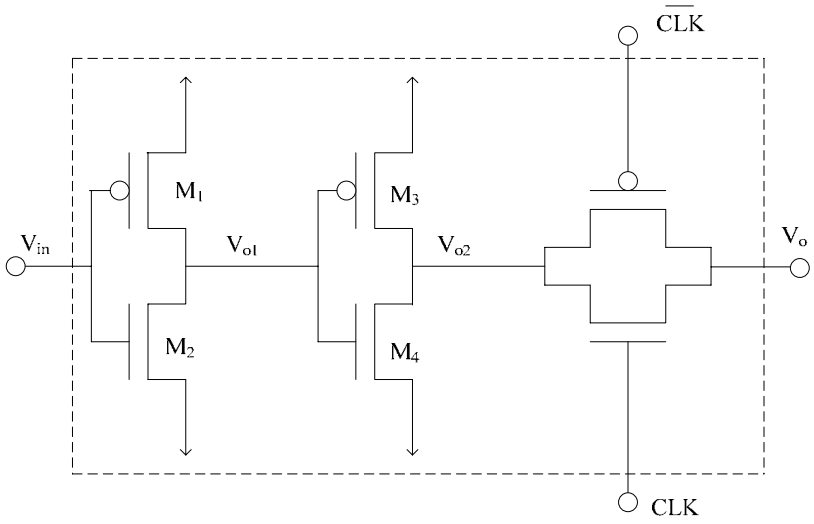


Figure 5-3 Clocked Digital Comparator

The analog quantization level of clocked digital comparator is the switching threshold voltage of the quantization inverter. It is a reference voltage and is self-determined by the size ratio of NMOS and PMOS. As shown in Figure 2(a), this internal reference voltage, V_m, is defined as the input voltage V_{in} of the quantization inverter when the output voltage V_{o1} equals to V_{in}, where both PMOS and NMOS transistors are in saturation. Figure 2(a) is the static voltage transfer characteristic (VTC) of the inverter. The voltage V_{dd} is the supply voltage of the process. The value of V_m is expressed as

$$V_m = \frac{\sqrt{\frac{\mu_p W_p}{\mu_n W_n}}(V_{dd} - |V_{Tp}|) + V_{Tn}}{1 + \sqrt{\frac{\mu_p W_p}{\mu_n W_n}}} \tag{5.7}$$

where V_{Tp} and V_{Tn} are threshold voltages of PMOS and NMOS devices; W_p and W_n are widths of PMOS and NMOS; μ_p and μ_n are hole mobility and electron mobility, respectively.

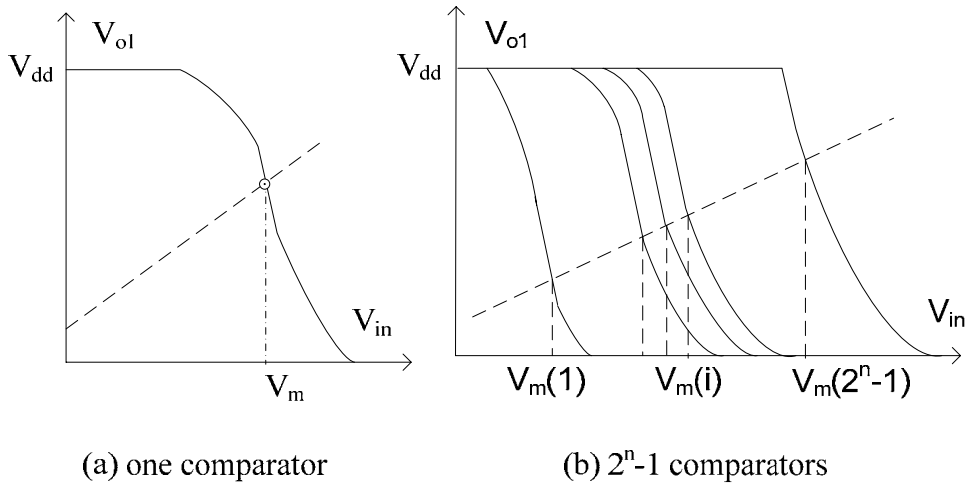
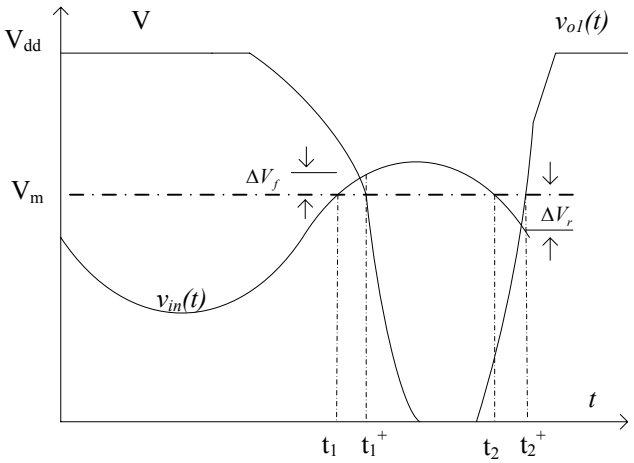


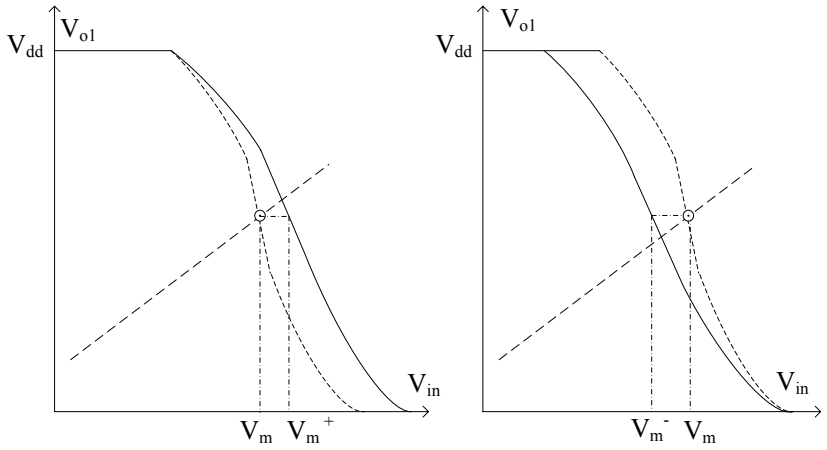
Figure 5-4 Static VTC

Figure 5-4(b) shows the static VTC for the 2^n-1 comparators. The 2^n-1 analog quantization levels $V_m(i)$, i = 1,….2^n-1, are equally-spaced. However, due to the hysteresis of the inverter switching, there will be a time delay (a time offset) Δt_{offset} for V_{o1} rise to or fall to V_m after $V_{in} = V_m$. If the Δt_{offset} of 2^n-1 comparators are all different, the outputs V_{o1} of all comparators won't equally-spaced. Then the analog quantization levels will have irregular offsets when compared with their corresponding values in static VTC. The offsets

of the comparators and the offset differences among the comparators introduce dynamic noises and degrade ADC dynamic performance.



(a) switching time



(b) falling volt. offset (c) rising volt. offset

Figure 5-5 Transient Behavior of CDC Quantization Inverters

To model the dynamic offsets, we analyze the transient behavior of a clocked digital comparator for an input sinusoidal signal, $v_{in}(t)=A\cos(\omega t)$ where A is signal amplitude and ω is signal frequency. Figure 5-5 shows the transient behaviors of the quantization inverter

in the digital comparator. The switching time offset is defined as the offset time Δt_{offset} for the inverter output V_{o1} to rise to or fall to V_m after $V_{in} = V_m$. Both fall offset time $\Delta t_{f\text{-}offset} = (t_1^+ - t_1)$ and rise offset time $\Delta t_{r\text{-}offset} = (t_2^+ - t_2)$ are shown in Figure 5-5(a). When V_{in} is approaching to $V_{m,}$, assuming the load capacitance is small and both NMOS and PMOS are in saturation, $\Delta t_{f\text{-}offset} \approx \Delta t_{r\text{-}offset}$ (=Δt_{offset}) and is expressed as

$$I_{ds} = \tfrac{1}{2} K_n (\tfrac{W}{L})_n (V_{in} - V_{Tn})^2 (1 + \lambda_n V_{o1}) \tag{5.8}$$

$$I_{ds} = \tfrac{1}{2} K_p (\tfrac{W}{L})_p (V_{dd} - V_{in} - |V_{Tp}|)^2 [1 + \lambda_p (V_{dd} - V_{o1})] \tag{5.9}$$

$$\frac{\Delta i_{dsn}}{\Delta v_{o1}} = \lambda_n I_{ds} \tag{5.10}$$

$$\frac{\Delta i_{dsp}}{\Delta v_{o1}} = -\lambda_p I_{ds} \tag{5.11}$$

$$\Delta i_{ds} = \Delta i_{dsn} - \Delta i_{dsp} = (\lambda_n + \lambda_p) I_{ds} * \Delta v_{o1} \tag{5.12}$$

$$\Delta t_{offset} = C_{o1} \frac{\Delta v_{o1}}{\Delta i_{ds}} \approx \frac{C_{o1}}{(\lambda_n + \lambda_p) I_{ds}} \tag{5.13}$$

where C_{o1} is the capacitive load at the output V_{o1}; I_{ds} is the drain-source current of NMOS and PMOS; λ_n and λ_p are active-region slope parameters for NMOS and PMOS, respectively.

On the other hand, during time Δt_{offset}, $v_{in}(t)$ will either rise or fall to V_m^+ or V_m^- depending on if it is in the rising or falling cycle. The difference between V_m and V_m^+ or V_m^- is the dynamic voltage offset. Both falling voltage offset $\Delta V_{f\text{-}offset} = (V_m^+ - V_m)$ and rising voltage offset $\Delta V_{r\text{-}offset} = (V_m - V_m^-)$ are shown in Figures 5-5(b) and (c). Since rising offset and falling offset happen when both transistors are in saturation, $\Delta V_{f\text{-}offset} \approx \Delta V_{r\text{-}offset}$ and output offset ΔV_{offset} is expressed by

$$\Delta V_{offset} = A_0 * (A\omega)\sin \omega t * \Delta t_{offset} \tag{5.14}$$

It is clear that ΔV_{offset} is frequency-dependent, so called dynamic offset. Substituting Equ. (5.13) to Equ. (5.14), we obtain

$$\Delta V_{offset} = A_0 * (A\omega)\sin \omega t \frac{C_{o1}}{(\lambda_n + \lambda_p) I_{ds}} \tag{5.15}$$

where λ_n and λ_p are process-dependent, and A and ω are signal-dependent.

To suppress the dynamic offset ΔV_{offset} of comparators and to eliminate the offset differences among comparators, as shown in Equ. (5.15) the C_{o1} is designed as small as possible and has the same value for 15 comparators, and similarly the I_{ds} is designed as large as possible and has the same value for 15 comparators.

5.4 Optimization of Sizing Design

5.4.1 Methodology

Suppressing the CDC dynamic offset demands a large value of I_{ds} as well as a small value of C_{o1}. However, with a fixed gate-source voltage, increasing I_{ds} is only by increasing sizes of the transistors, which results in large parasitic capacitance of C_{o1} in the CDC comparator. Thus optimization of I_{ds} is required for dynamic offset suppression in CDC design.

A circuit-level optimal process of CDC design is developed for the optimization of I_{ds} and the optimization of the dynamic offset suppression. In the optimal process, the CDC

feature values, including C_{o1}, I_{ds}, and the ratio of them, are firstly analyzed by calculated in Matlab with a simple model to gain an insight into the relationship between CDC dynamic offsets and I_{ds}. The simple model used here is the level-1 transistor model, a model that generally is used by most of analog circuit design analysis. Then the I_{ds} of the CDC is determined optimally by simulating 6 different sizing CDCs in Cadence Spectre with a more complex transistor model. The model used in the simulation is the level-49 transistor model, a model that is used for IBM 130 nm process. This two-step optimal process results in an optimization of dynamic offset suppression in the CDC design.

Before optimizing the CDC design for dynamic offset suppression, we analyze the relationship between CDC dynamic offsets and the switching speed of its quantization inverter. Switching speed of an amplifier is generally expressed by its output slew rate [55]. The slew rate of CDC is then expressed by

$$SlewRate = \max(\frac{\Delta v_{out}}{\Delta t}) = \frac{I_{sat}}{C_L} \tag{5.16}$$

The slew rate of the quantization inverter represents the switching speed as in Equ. (5.17):

$$SlewRate = \max(\frac{\Delta v_{o1}}{\Delta t}) = \frac{I_{ds}}{C_{o1}} \tag{5.17}$$

In Equ. (5.13) and (5.15) both the time and voltage offsets are proportional to the ratio of C_{o1} and I_{ds} and inversely proportional to the slew rate of the CDC's quantization inverter as shown in Equ.(5.18) and (5.19).

$$\Delta t_{offset} \propto \frac{C_{o1}}{I_{ds}} = \frac{1}{SlewRate} \tag{5.18}$$

$$\Delta V_{offset} \propto \frac{C_{o1}}{I_{ds}} = \frac{1}{SlewRate} \tag{5.19}$$

Therefore, Higher slew rate will generate smaller dynamic offset in CDCs. In our dynamic offset suppression slew rates of 15 CDC's quantization inverters are analyzed to determine the optimum value of I_{ds}.

To determine the slew rate of the quantization inverter, two parameters C_{o1} and I_{ds} are calculated and analyzed. In the CDC comparator, the C_{o1} is the sum of gate-to-drain capacitance, bulk-to-drain capacitance of the quantization inverter and capacitive load from the CDC second inverter. Figure 5-6 shows the small-signal model of a CDC quantization inverter for calculating the capacitance load C_{o1} in Equ.(5.20), where C_{gd1} and C_{gd2} are the gate-to-drain capacitances of NMOS and PMOS, respectively. C_{bd1} and C_{bd1} are the bulk-to-drain capacitances of NMOS and PMOS, respectively. C_L is the capacitive load from the CDC second inverter.

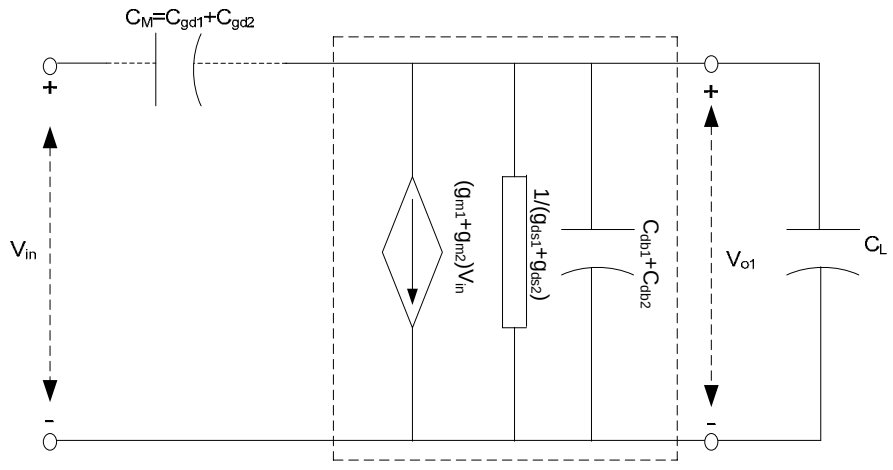


Figure 5-6 Small-Signal Model of CDC Quantization Inverters

$$C_{o1} = C_{gd1} + C_{gd2} + C_{bd1} + C_{bd2} + C_L \tag{5.20}$$

For predicting the input capacitance of CDC-based ADCs, the input capacitance of CDC is approximated in Equ. (5.21), referring to Figure 5-2(a), which is under Miller effect, where g_{m1} and g_{m2} are the trans-conductance of M1 and M2, respectively. g_{ds1} and g_{ds2} are the conductance of M1 and M2, respectively.

$$C_{in} = \left(1 + \frac{g_{m1} + g_{m2}}{g_{ds1} + g_{ds2}}\right)\left(C_{gd1} + C_{gd2}\right) + \left(C_{gs1} + C_{gs2}\right) \tag{5.21}$$

Equ. (5.22), (5.23) and (5.24) are used for calculating the gate-to-drain, gate-to-source and bulk-to-drain parasitic capacitances, C_{gd}, C_{gs} and C_{bd}. C_{ox} is the oxide capacitance. LD is the lateral diffusion. W_{eff} is the effective channel width. AD is the area of the drain. PD is the perimeter of the drain. CJ is the zero-bias junction capacitance, $CJSW$ is the zero-bias, bulk-drain sidewall capacitance. PB is the bulk junction potential. MJ is the bulk junction grading coefficient. $MJSW$ is the bulk-drain sidewall grading coefficient. The calculation of AD and PD is shown in Appendix A.

$$C_{gd} = C_{ox}\left(LD\right)\left(W_{eff}\right) \tag{5.22}$$

$$C_{gs} = C_{ox}\left(LD + 0.67L_{eff}\right)\left(W_{eff}\right) \tag{5.23}$$

$$C_{bd} = \left(CJ\right)\left(AD\right)\left[1 + \left(MJ\right)\frac{V_{bd}}{PB}\right] + \left(CJSW\right)\left(PD\right)\left[1 + \left(MJSW\right)\frac{V_{bd}}{PB}\right] \tag{5.24}$$

$$I_{ds} = K' \frac{W}{2L}\left(V_{GS} - V_T\right)^2\left(1 + \lambda V_{DS}\right) \tag{5.25}$$

The drain-source current in saturation is calculated in Equ.(5.25), where W and L are the width and length of transistor. V_{GS} and V_{DS} are the gate-source and drain-source voltages.

V_T is the transistor threshold voltage. K' is the trans-conductance parameter in saturation and λ is the channel length modulation parameter. As analyzed in the previous section, the offsets happen when both transistors are in saturation. The critical segment in the switching time of CDC quantization inverter is the time segment that both transistors are in saturation, which is used to approximate the slew-rate of the quantization inverter.

Parameter values in Equ. (5.22) to (5.24) are taken from IBM 130 nm CMOS process A diagram of Matlab analysis in Figure 5-7 is designed to illustrate the calculation of I_{ds}, C_{in}, C_{o1}, and the slew rate (SR) for optimization of I_{ds}, CDC transistor sizing, and dynamic offset suppression.

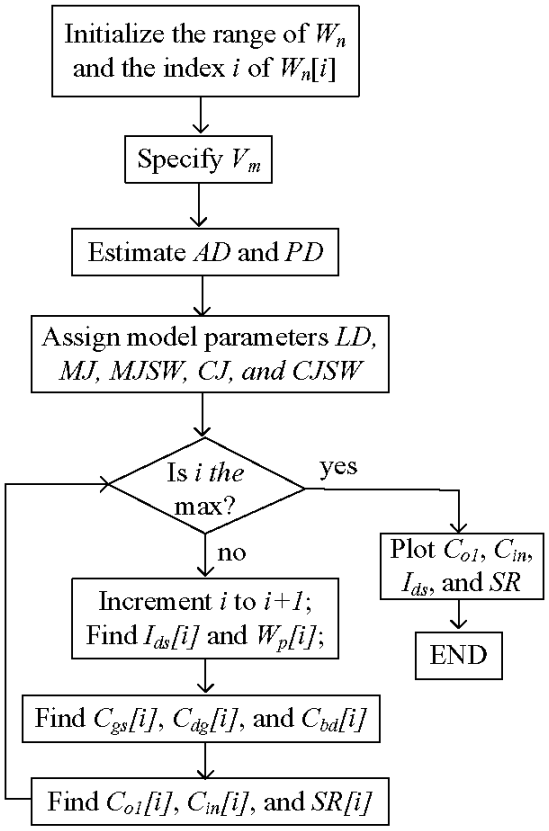


Figure 5-7 Diagram for Matlab Analysis

Figure 5-8 shows the results of the diagram of Matlab analysis using IBM 130 nm CMOS process parameters where the desirable threshold voltage V_m equal to 0.484 V. Figures 5-8 (a), (b) and (c) show that C_{o1} (fF), I_{ds} (μA) and C_{in} (fF) all monotonously increase as the NMOS width increases. It is shown from Figures 5-8(a) and (c) the bulk-to-drain capacitance is much larger than its gate-to-drain capacitance, especially in scaling-down CMOS processes. Figure 5-8 (d) illustrates that the slew-rate is saturated when the I_{ds} increases to 150 μA.

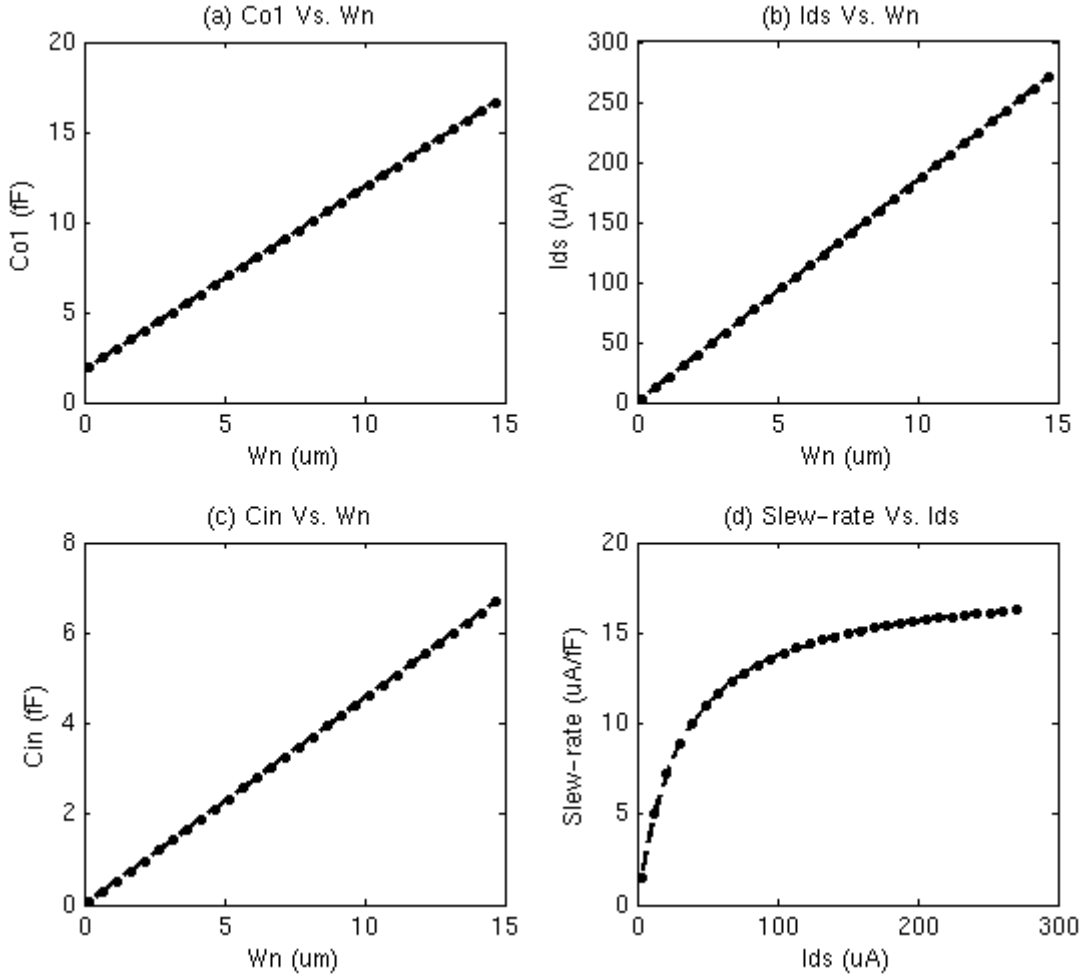


Figure 5-8 I_{ds}, C_{o1}, C_{in}, Slew-Rate Vs. NMOS width at V_m = 0.484v

A large slew-rate is desired for efficient dynamic offset suppression and high speed conversion. If the slew-rate is saturated, then there is no need to increase the I_{ds} by increasing the size of transistors which in turn will increase the input capacitance C_{in} of CDC, as shown in Figure 5-8 (c). The total input capacitance of CDCs determines the input signal frequency bandwidth of flash ADC. Therefore, a threshold current $I_{ds\text{-}threshold}$ of CDC is defined as the value of I_{ds} when the slew-rate become saturated.

When the slew rate becomes saturated, it does not decrease as I_{ds} is increasing. However, in Cadence simulation, the slew rate decreases when the values of I_{ds} are either too small or too large which are shown in Figures 5-9 and 5-10.

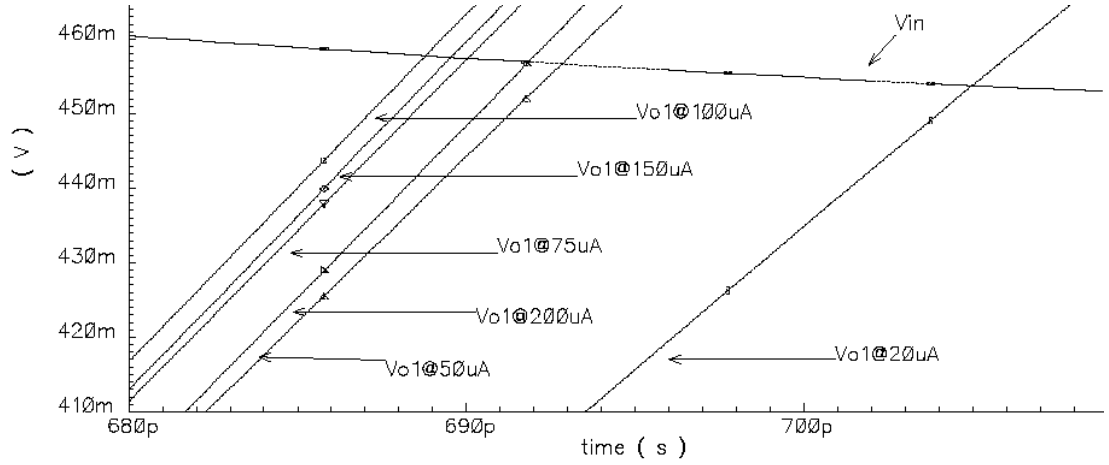


Figure 5-9 Comparison of V_{o1} Rising Speeds with V_m=0.484v and Various I_{ds}

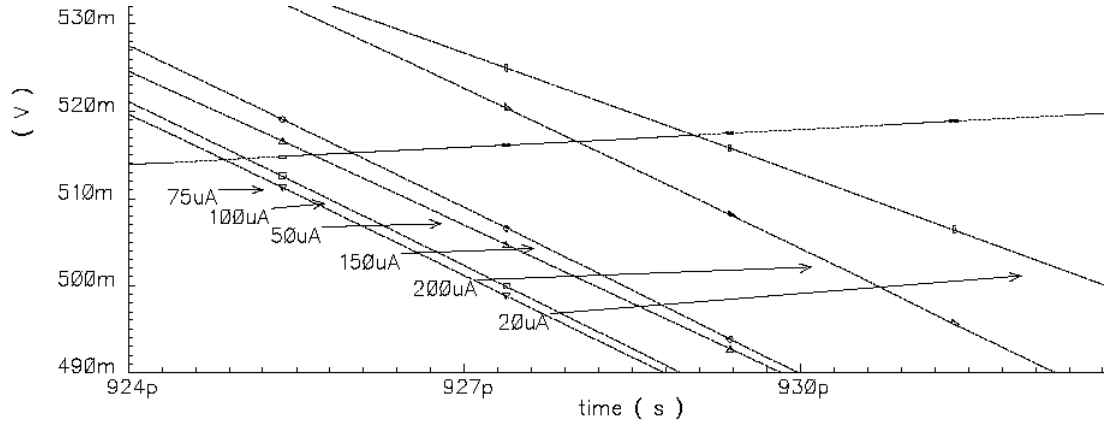


Figure 5-10 Comparison of V_{o1} Falling Speeds with V_m=0.484v and Various I_{ds}

Following the diagram for Matlab analysis in Figure 5-7, we enumerate 6 different values of I_{ds} and design 6 different CDC comparators for a same threshold voltage V_m of 0.484 V. Figures 5- 9 and 5-10 show switching speed comparison of the 6 quantization inverters with 6 different I_{ds} (20 µA, 50 µA, 75 µA, 100 µA, 150 µA, and 200. The input sinusoidal signal frequency is 1 GHz. Figures 5-9 and 5-10 compare the rising and the falling speed at the output V_{o1} of the 6 quantization inverters. It is observed that 50µA is close to the saturated current $I_{ds\text{-}threshold}$ and the comparator with I_{ds} = 50µA is only 3ps slower than the fastest one. As shown in Table 5-1, the comparator with I_{ds} = 100µA has the fastest rising speed and the comparator with I_{ds} = 75µA has the fastest falling speed. The optimum value of I_{ds} to achieve the fastest switching speed is in the range of [75µA, 100µA]. This enumeration method can be repeated with I_{ds} in the small range of [75µA, 100µA] for optimum value of I_{ds}.

Table 5-1 Comparison of Switching Speeds of V_m = 0.484v and 6 different I_{ds}

#			1	2	3	4	5	6
I_{ds} (μA)			20	50	75	100	150	200
V_m(v)	speed	rising	6	5	**3**	**1**	2	4
	Order	falling	6	3	**1**	**2**	4	5
0.484	W_p (μm)		0.49	1.09	1.58	2.08	3.07	4.08
	W_n (μm)		1.04	2.71	4.09	5.48	8.24	11.03

The $I_{ds\text{-}threshold}$ obtained from Cadence simulation is slightly smaller than the value calculated from the diagram of Matlab analysis. The above optimization process is for one CDC comparator and the threshold voltage V_m equals to 0.484v. The same optimization process can be repeated for all CDC comparators with different threshold voltages V_m. Ideally, a same value of I_{ds} for all CDC quantization inverters is desired to suppress dynamic offsets. In reality, the $I_{ds\text{-}threshold}$ values of all CDCs are likely different because their threshold voltages V_m are different. Therefore, we need to find an optimal value of I_{ds} for each CDC.

5.4.2 Optimum I_{ds}

The design optimization developed for dynamic offset suppression is applied and validated on a 4-b 2.5-GSPS CMOS flash ADC. Table 5-2 shows switching speed comparison of three CDCs with V_m in the range of [0.45v, 0.668v]. $V_m(i)$ with i = 1…15 represents the threshold voltages of 15 CDCs in the 4-b flash ADC. $V_m(1)$ is the threshold voltage of the first CDC; $V_m(15)$ is the threshold voltage of the last CDC. The transistor

widths of NMOS and PMOS, I_{ds}, and the switching speed order (rising and falling) for each CDC are indicated in Table 5-2. The optimum I_{ds} of $V_m(1)$ and $V_m(15)$ is determined in the range [75μA, 100μA].

Table 5-2 Comparison of Switching Speeds for Various V_m

#	I_{ds} (μA)	$V_m(1)$ = 0.45v				$V_m(8)$ = 0.569v				$V_m(15)$ = 0.688v			
		W_n (μm)	W_p (μm)	speed order		W_n (μm)	W_p (μm)	speed order		W_n (μm)	W_p (μm)	speed order	
				rising	falling			rising	falling			rising	falling
1	20	1.86	0.42	6	4	0.30	0.76	6	6	0.18	2.72	1	6
2	50	4.74	0.94	3	3	0.94	1.70	5	5	0.36	4.51	5	3
3	75	7.15	1.36	**1**	**1**	1.44	2.50	4	4	0.58	6.78	**4**	**1**
4	100	9.58	1.78	**2**	**2**	1.93	3.32	3	3	0.79	9.08	**2**	**2**
5	150	14.43	2.63	4	5	2.91	4.96	2	2	1.21	13.69	3	4
6	200	19.33	3.49	5	6	3.91	6.63	**1**	**1**	1.63	18.29	6	5

Assume $I_{ds\text{-}threshold}(i)$ with i =1…15 represents the threshold currents of 15 CDC quantization inverters. We'll have

$$I_{ds-threshold}(1) < I_{ds-threshold}(2)..... < I_{ds-threshold}(8) \tag{5.26}$$

$$I_{ds-threshold}(8) > .....I_{ds-threshold}(14) > I_{ds-threshold}(15) \tag{5.27}$$

Thus, the CDCs with $V_m(1)$ and $V_m(15)$ become critical designs in the CDC design since their switching speeds determine the speed of the ADC.

From Table 5-2 it is shown that both $I_{ds\text{-}threshold}(1)$ and $I_{ds\text{-}threshold}(15)$ are smaller than 150μA because the switching speed decreases when I_{ds} is 150μA. $I_{ds\text{-}threshold}(8)$ is larger than 200μA because the switching speed increases when I_{ds} is 200μA. Then the optimum

I_{ds} for this 4-b flash ADC with the threshold voltage range [0.45v, 0.688v] is determined in the range of [75μA, 100μA]. In this 4-b flash ADC, I_{ds} = 80μA as its optimum value is being used for the CDC design. Optimization of I_{ds} also optimizes the dynamic offset suppression in the CDC design.

Three 4-bit flash ADCs with I_{ds} of 50μA, 80μA and 150μA and input dynamic range of [0.45v, 0.688v] was designed. Comparison of SFDR of these three ADCs is presented in Figure 5-11.

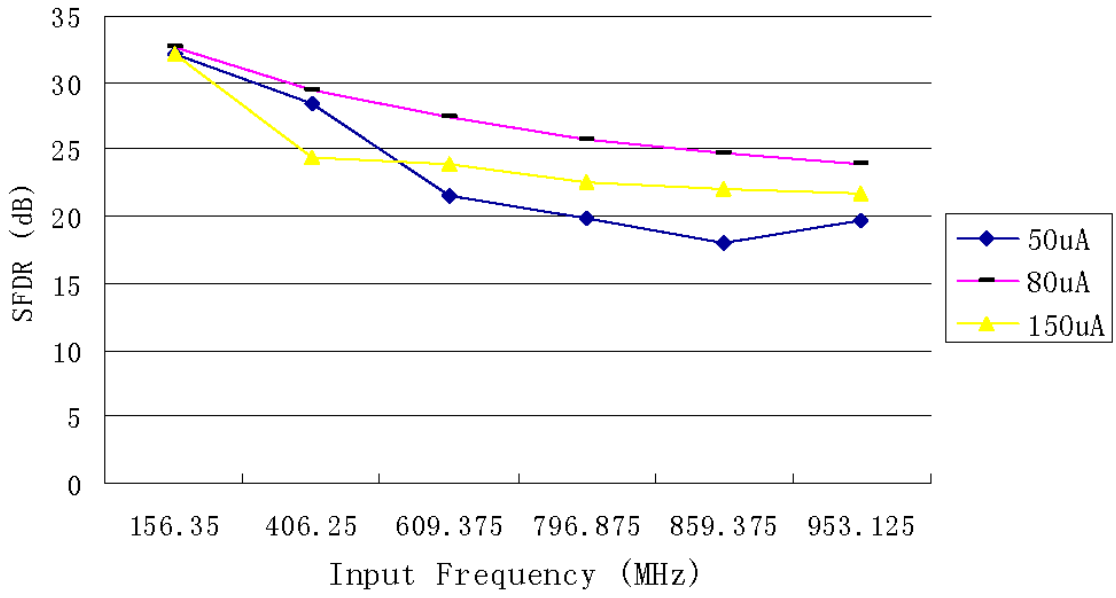


Figure 5-11 Comparison of ADC SFDR for Optimization of I_{ds}

The results testify the optimization of dynamic offset suppression in CDC design. As discussed above the I_{ds} of 80μA is an optimum value for designing the quantization inverter for efficient dynamic offset suppression. Its ADC achieves the highest SFDR. The quantization inverters with I_{ds} of 50μA and 150μA have smaller slew rates and

results in 3 to 5 dB down of SFDR compared with the quantization inverter with I_{ds} of 80μA as shown in Figure 5-10.

5.5 Summary

The CDC implemented by inverter configuration, with dynamic offset suppression and optimization, is characterized in the following:

- No external resistor or capacitance array needed for generating reference voltages, which results in small area and avoids resistive thermal noise which usually generates a reference voltage offset.
- Fewer analog components, which is robust for DSP interference in SoC applications and no requirement of device linearity in scaling-down CMOS technologies.
- Small static power consumption, which makes it suitable for SoC and battery-powered applications.
- No additional hardware needed for dynamic offset suppression
- Less possibility for the metastable output due to high switching speed of CDCs.

Chapter 6 CDC-BASED PIPELINED FLASH ADC ARCHITECTURE

The speed of monolithic flash ADCs is limited by the comparator switching speed and the propagation delay of the logic in the encoder [51]. This CDC-based pipelined flash ADC architecture adopts high-speed comparators and pipelined clocked DCVSPG encoders to achieve a maximum sampling frequency. Using the dynamic offset suppression technique and circuit optimization method presented in Chapter 5, the CDC-based pipelined flash ADC can achieve 2-GHz signal bandwidth. The architecture is well suited for integrated SoC applications due to its full compatibility with digital CMOS technology, low power and small area.

6.1 Architecture

Two block diagrams of the CDC-based pipelined CMOS flash ADC architecture are shown in Figures 6-1 and 6-2. Both block diagrams have the same architecture except different encoding schemes. The direct-binary encoder takes an 1-of-n code after the 1/0 boundary detector and outputs a binary code (Figure 6-1). The Gray encoder takes the thermometer code, decodes it into Gray code, and then outputs a binary code (Figure 6-2).

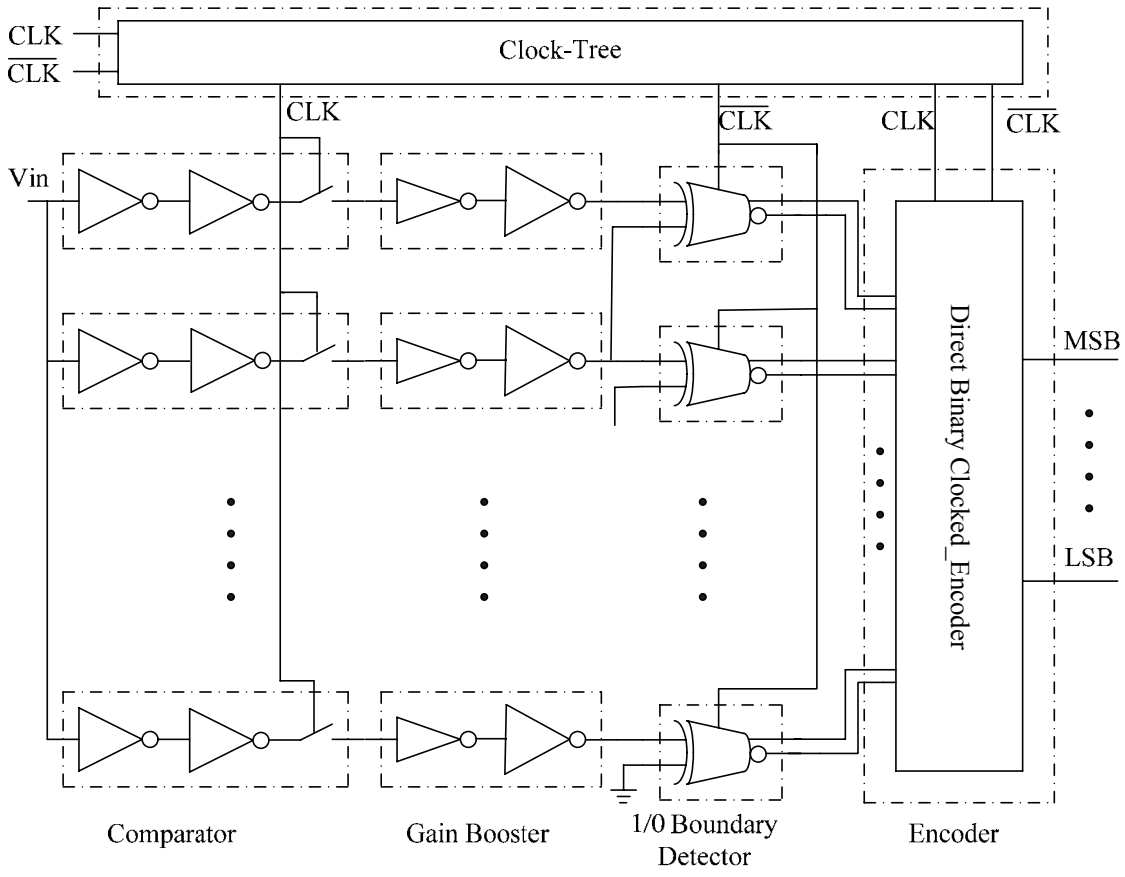


Figure 6-1 Architecture of CDC-based Pipelined ADC with Direct-Binary Encoding

Figure 6-1 presents an ADC architecture with direct-binary encoding which includes an array of CDC comparators, an array of gain booster, an array of 1/0 boundary detectors, a direct-binary clocked encoder, and a clocked tree. For a 4-b, 1-GHz signal bandwidth ADC in 130-nm CMOS process, the maximum tolerant aperture error is about 20 ps. The clock signal from the clock tree has transient time about 15 ps. Therefore the ADC output should have less possibility of sparkles.

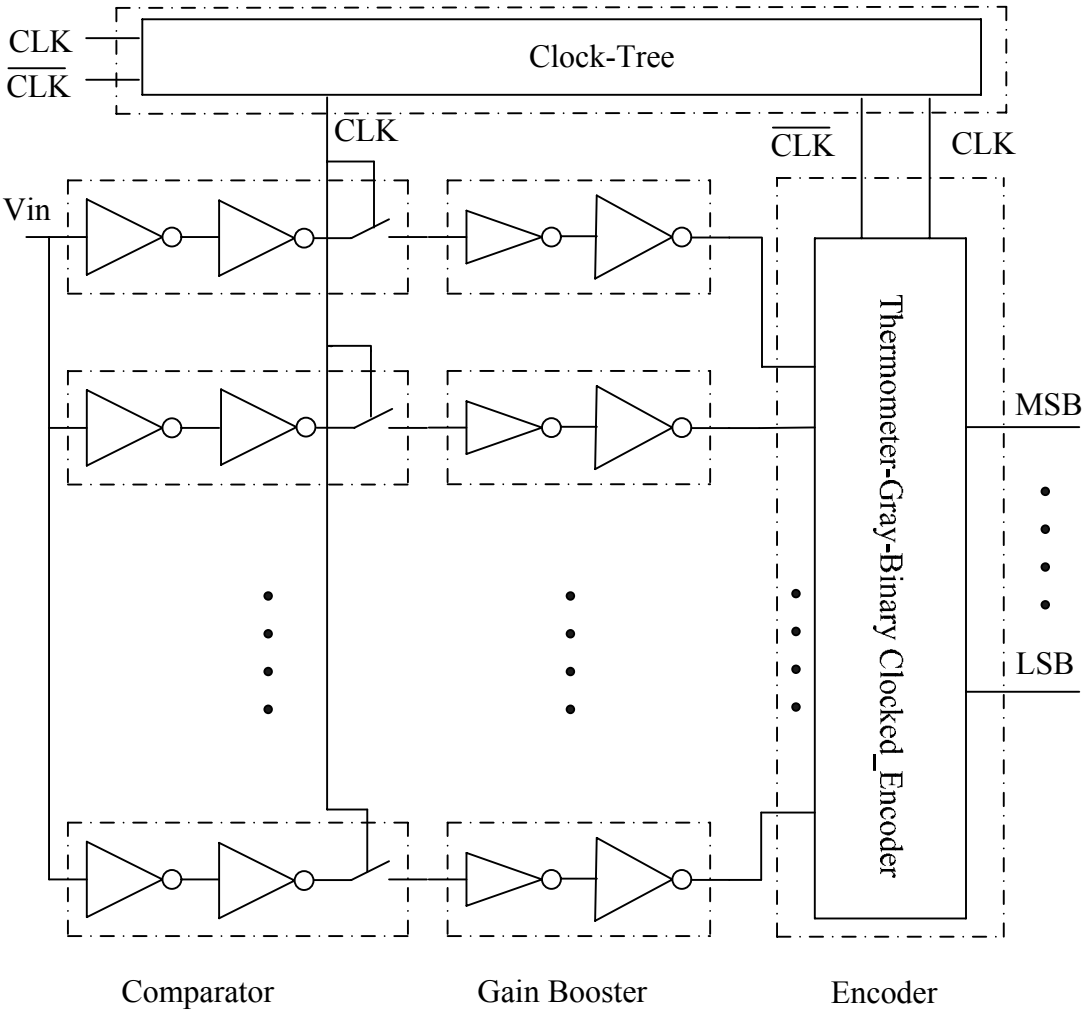


Figure 6-2 Architecture of CDC-based Pipelined ADC with Gray Encoding

The ADCs with Gray encoding in Figure 6-2 have better dynamic performances than the ADCs with direct-binary encoding because of better sparkles and meta-stability suppression. The timing diagram of CDC-based pipelined CMOS flash ADC is shown in Figure 6-3. The clock signals *CLK* and *CLKB* are the inputs to the CDCs and the pipelined encoder, no matter either direct-binary encoding or Gray encoding is employed.

The associated timing diagram of CDC-based pipelined CMOS flash ADC is shown in Figure 6-3. The clock signals *CLK* and *CLKB* are the inputs for the CDCs with complementary digital CMOS clock and the pipelined encoder regardless of direct-binary encoding or Gray encoding.

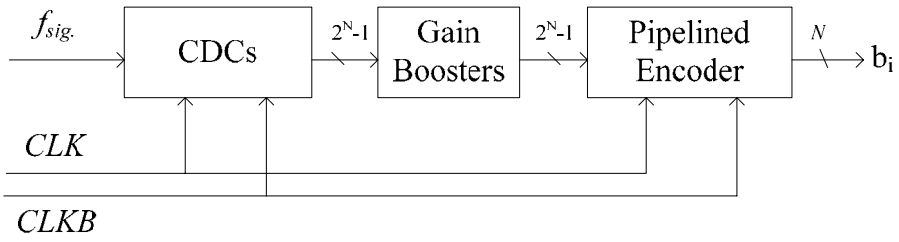


Figure 6-3 Timing Diagram of CDC-based Pipelined CMOS Flash ADC

The sampling rate in the proposed CDC-based pipelined CMOS flash ADC is determined by the propagation delay in the pipelined encoder plus the clock transition time, not the switching speed of the CDCs. The pipelined encoder is implemented in DCVSPG clocked logic. One clocked DCVSPG logic has about 57 ps transition time (propagation delay) in IBM 130 nm digital CMOS process. The total transition time including the clock transition time is about 87 ps. The gain booster is two cascaded inverters and the propagation delay is much less than 87 ps. The regenerative speed of the CDCs with inverter configuration, optimized with high slew rate, should be in the speed range of an inverter, which is about 10 to 20 ps. The signal transition time in the ADC would be less than 100 ps. Thus the maximum sampling frequency of the CDC-based pipelined CMOS flash ADC could be up to 5-GSPS.

6.2 CDCs and Gain Boosters

The CDC design is presented in the previous chapter. The first quantization inverter of CDC is optimized with dynamic offset suppression. The second inverter of CDC is designed to sharpen the quantized output and also a small load to the quantization inverter in order to have a small dynamic offset.

The signal bandwidth of a CDC-based pipelined CMOS flash ADC varies when the input voltage range changes. Experiments of observing the 3-dB bandwidth and the input voltage ranges of CDCs in 130 nm CMOS process are conducted and the results are shown in Table 6-1.

Table 6-1 Relation between Input Voltage Range and 3-dB Bandwidth of CDCs in IBM 130-nm Digital CMOS Process

#.	Input Voltage Range (mV)		3-dB Bandwidth (GHz)	Resolution of Implementation	Annotation
1	**238**	450---688	**1.1**	4-b	Initial research objective with 1-GHz bandwidth
2	**248**	445---693	**1.0**	6-b	1-GHz bandwidth 6-b ADC
3	**119**	501---620	**2.0**	4-b	4-GSPS for 2-GHz bandwidth
4	**15**	562---576	**2.0**	4-b	1mv LSB 4-b ADC

6.3 Pipelined DCVSPG Encoders

To achieve high sampling frequency in flash ADCs, pipelined encoder divides the total propagation time into small time segments. In this research, two pipelined encoding schemes, both implemented in clocked logic gates using differential cascade voltage switch with pass-gate (DCVSPG) logic [56], are presented. This section first introduces the pipelined encoder architecture and then presents the DCVSPG logic implementation.

6.3.1 Encoder Architecture

Two pipelined DCVSPG encoders are proposed. The first encoder is direct-binary balanced encoder which takes an 1-of-n code after 1/0 boundary detector. The 1/0 boundary detector is a two-input XOR gate converting a thermometer code to a 1-of-n code. A 4-bit direct-binary balanced encoder is shown in Figure 6-4.

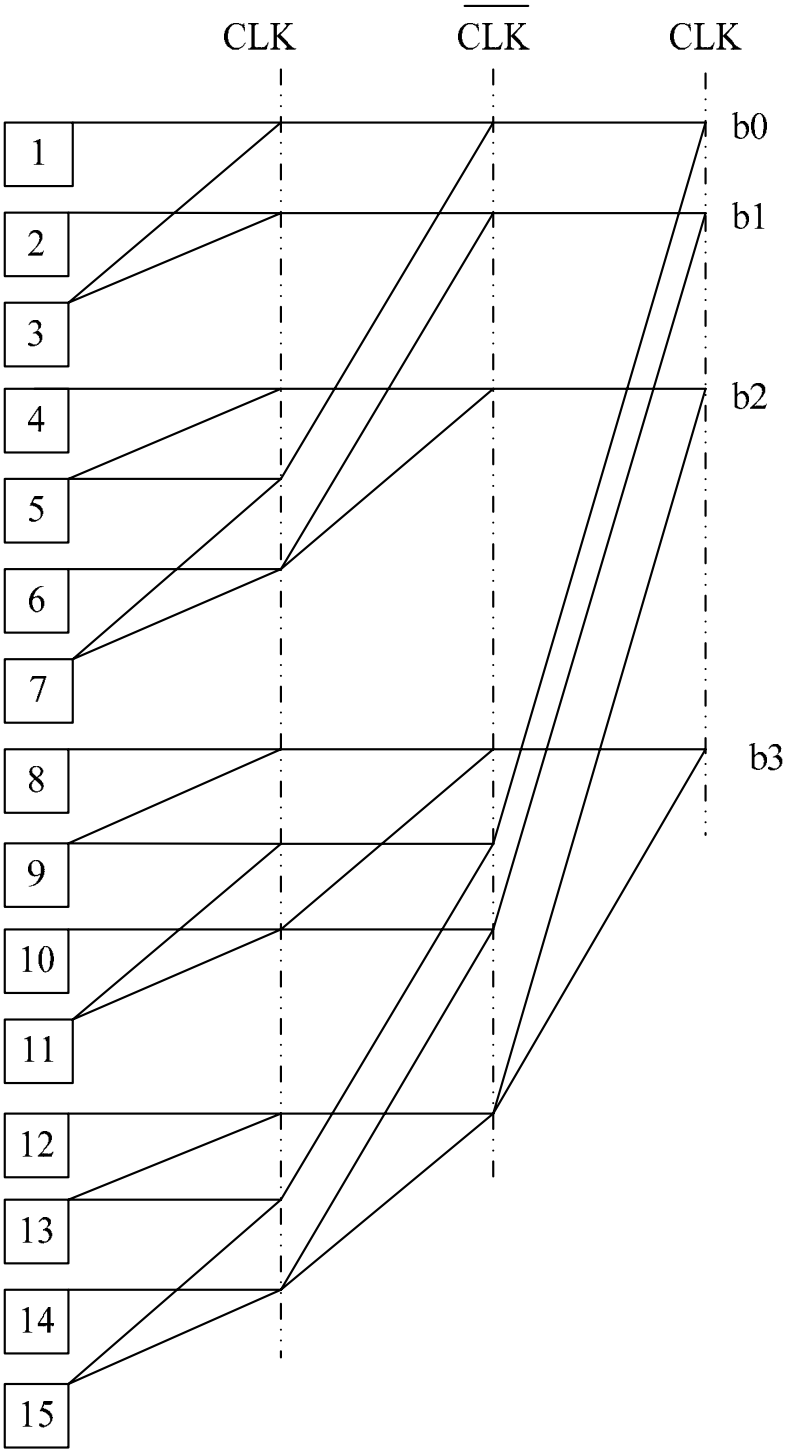


Figure 6-4 4-b Direct-Binary Balanced Encoder

The binary output code $b_3b_2b_1b_0$ can be expressed in a balanced distribution in terms of 1-of-n code as follows:

$$b_0=[(1+3)+(5+7)] + [(9+11)+(13+15)]$$

$$b_1=[(2+3)+(6+7)] + [(10+11)+(14+15)]$$

$$b_2=[(4+5)+(6+7)] + [12+13)+(14+15)]$$

$$b_3=[(8+9)+(10+11)] + [(12+13)+(14+15)]$$

These balanced logic functions are suitable for pipeline implementation.

The second encoder is thermometer-Gray-binary (TGB) encoder which include two-step encoding processes: 1) thermometer-to-Gray code, and 2) Gray-to-binary code. A 4-bit pipelined thermometer-Gray-binary encoder to illustrate the two-step encoding processes is shown in Figures 6-5 and 6-6. The Gray code outputs $G_3G_2G_1G_0$ of the thermometer-to-Gray code in Figure 6-5 can be expressed by thermometer codes as follows.

$$G_3 = T_8 \tag{6.1}$$

$$G_2 = T_4 \overline{T_{12}} \tag{6.2}$$

$$G_1 = T_2 \overline{T_6} + T_{10} \overline{T_{14}} \tag{6.3}$$

$$G_0 = T_1 \overline{T_3} + T_5 \overline{T_7} + T_9 \overline{T_{11}} + T_{13} \overline{T_{15}} \tag{6.4}$$

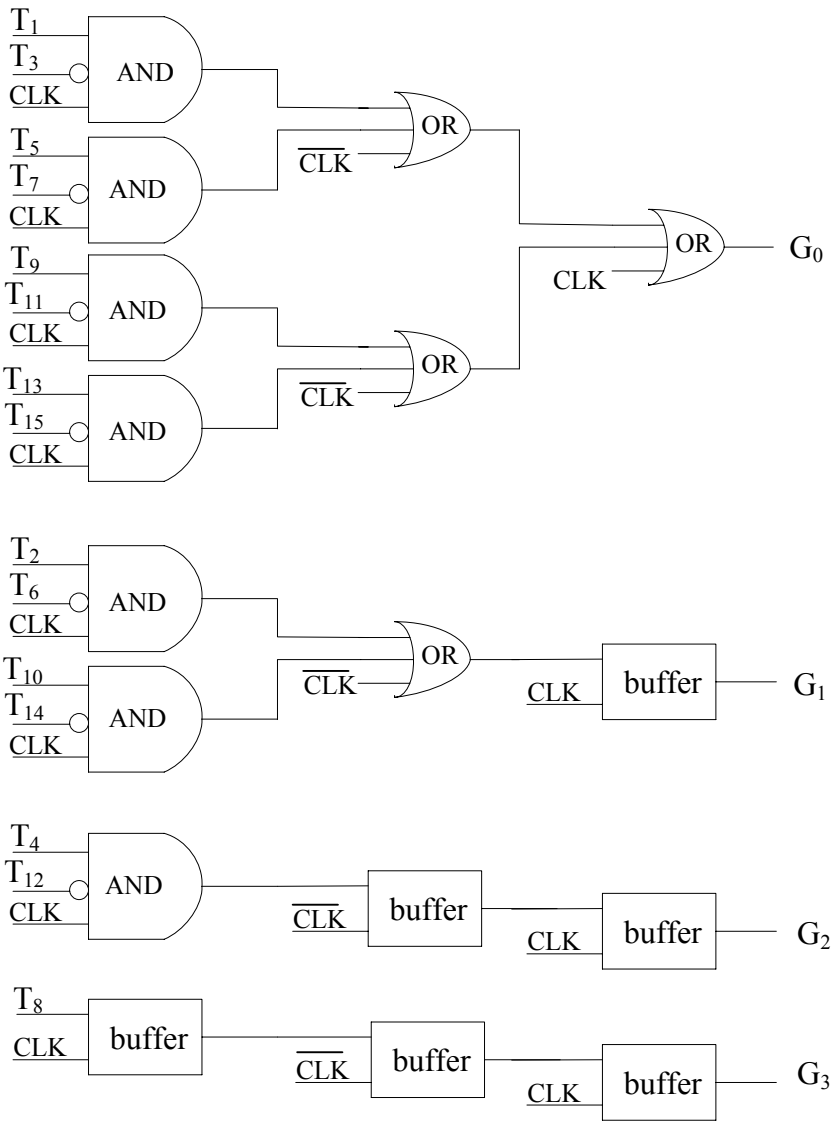


Figure 6-5 Pipelined Encoding of Thermometer-to-Gray Code

The binary code outputs $b_3b_2b_1b_0$ of the pipelined encoding in Figure 6-6 are expressed in terms of Gray code as follows

$$b_3 = G_3 \tag{6.5}$$

$$b_2 = [G_3 \oplus G_2] \tag{6.6}$$

$$b_1 = [G_3 \oplus G_2] \oplus G_1 \tag{6.7}$$

$$b_0 = [G_3 \oplus G_2] \oplus [G_1 \oplus G_0] \tag{6.8}$$

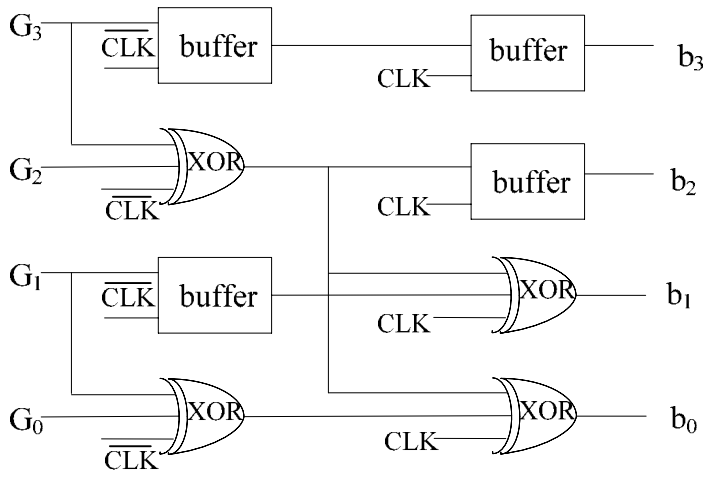


Figure 6-6 Pipelined Encoding of Gray-to-Binary Code

As discussed in chapter 4, the encoding scheme with Gray code has advantage of suppressing the sparkles and meta-stability, especially for high slew rate signals in wideband applications. In the next chapter, the majorities of implementation of the proposed CDC-based pipelined CMOS flash ADC are realized with Gray encoding scheme, while one 4-b implementation has both encoders and its performances are compared.

6.3.2 DCVSPG Logic Circuit Blocks

In the above two pipelined encoders, logic functions involved include *AND, OR, XOR*, and *Buffer* and all are implemented by the DCVSPG logic shown in Figures 6-7, 6-8, 6-9 and 6-10.

There are several advantages of the DCVSPG logic. Due to the cross-coupled pMOS device load, the DCVSPG logic is designed to have a built-in latch structure. The output is latched at the previous output value when the clock is "0." Both Q and QN are produced nearly at the same instance, which avoids an extra inverter delay to generate the QN from

the Q. The DCVSPG encoder is a ratioless logic, and the output obtained has no glitches. It has superior performance with power and area, especially suitable for pipelined by eliminating extra latches.

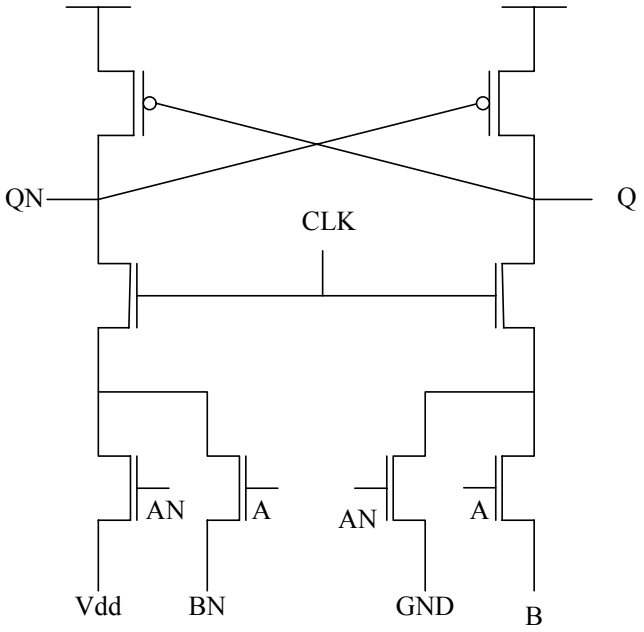


Figure 6-7 Clocked DCVSPG AND/NAND Gate

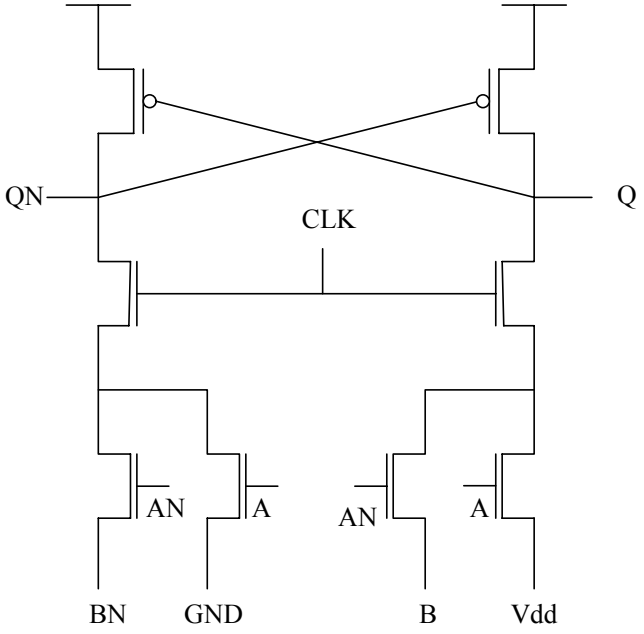


Figure 6-8 Clocked DCVSPG OR/NOR Gate

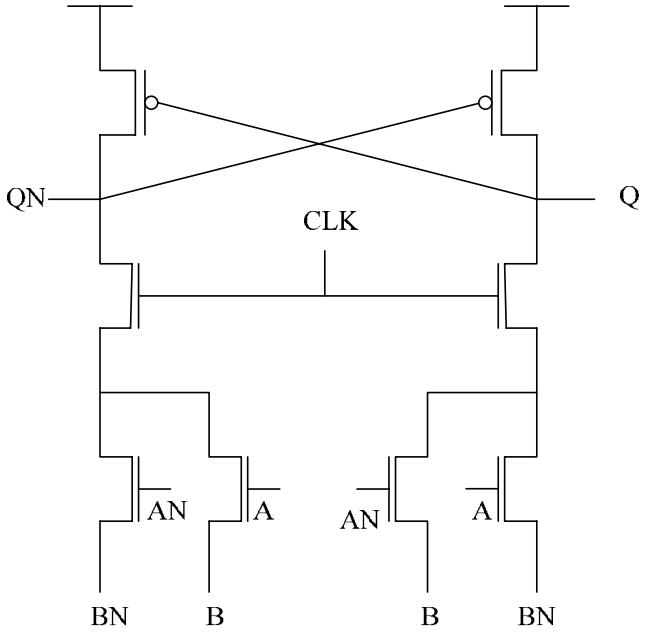


Figure 6-9 Clocked DCVSPG XOR/XNOR Gate

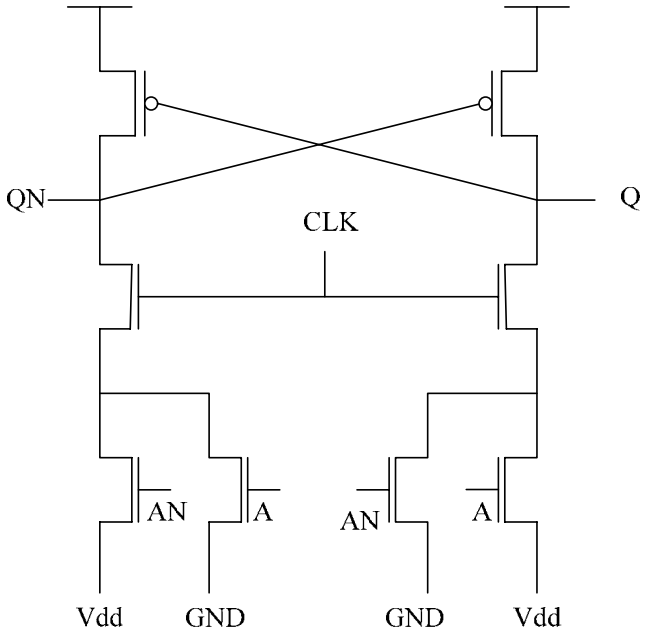


Figure 6-10 Clocked DCVSPG Buffer

6.4 Clock Tree

Since flash ADCs provide an intrinsic sampling function for dynamic input signals, the clock timing uncertainty needs to be considered. The timing uncertainty Δt_{MAX} for a 6-b, 1-GHz bandwidth ADC is less than 5-ps and for a 4-b, 2-GHz bandwidth ADC is less than 10-ps from Equ.(4.7).

An inverter clock tree is designed. Simulations reveal rise/fall times of the clock signal are about 20 ps (slew rate 50G v/s) shown in Figure 6-11. The output load for each clock leaf is two DCVSPG gates. The average slew rate before the output load is 55G v/s. With the load, the average slew rate is 50G v/s. The maximum threshold voltage variation of the switch must be less than 300 mV according to Equ.(6.9).

$$\Delta V_T \le \frac{\Delta t_{\max}}{t_{rise/fall}} \left(V_{dd} - V_{ss} \right) \tag{6.9}$$

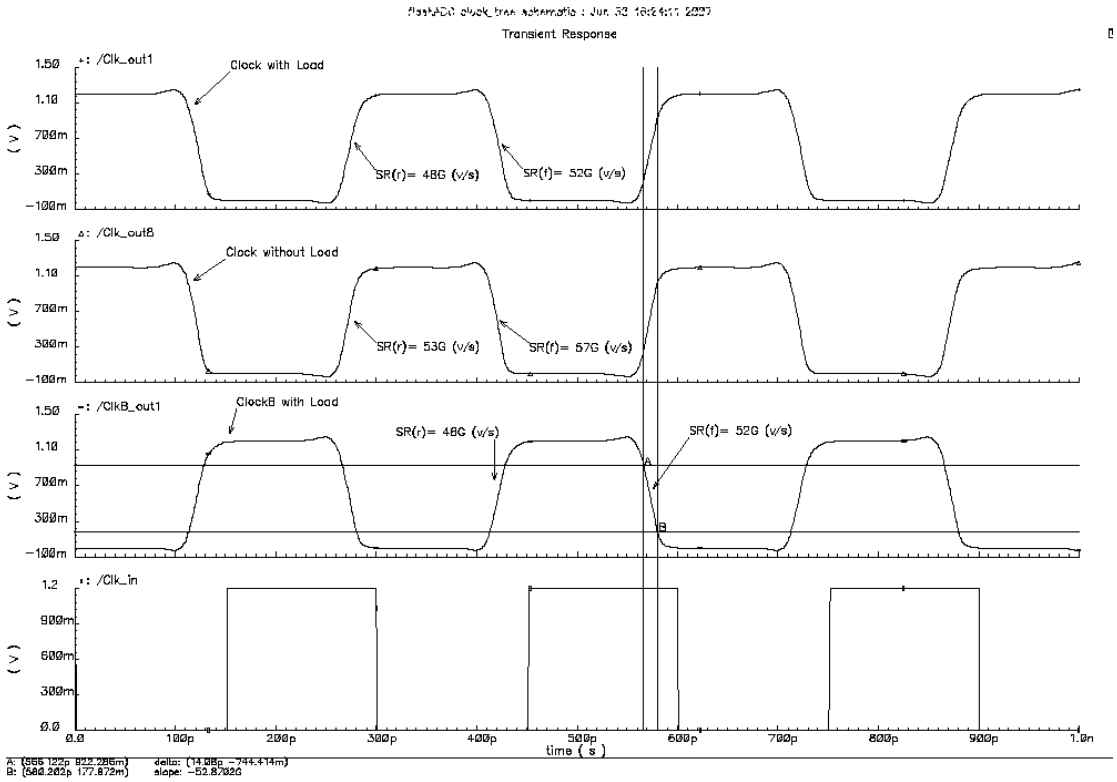


Figure 6-11　Performance of Clock Tree

Chapter 7 IMPLEMENTATION AND PERFORMANCE OF CDC-BASED PIPELINED CMOS FLASH ADC ARCHITECTURE

This chapter presents three implementations of CDC-based pipelined CMOS flash ADCs is 130 nm CMOS process. First, a 4-b, 1-GHz bandwidth 2.5-GSPS ADC is the initial research objective used in a digital receiver-on-a-chip. Both schematic and post-layout performances are compared. Secondly, a 4-b, 2-GHz bandwidth 4-GSPS ADC is implemented and compared with the state-of-the-art 4-b ADCs in literature. Lastly, a 6-b CDC-based pipelined ADC is implemented and its performance is evaluated. This 15mv-input-range CDC-based pipelined ADC also shows the performance difference of using balanced direct binary (BDB) encoder and thermometer-Gray-binary (TGB) encoder.

7.1 4-b 1.25-GHz Bandwidth 2.5-GSPS ADC

This 4-b 1-GHz bandwidth 2.5-GSPS flash ADC is proposed and implemented in three versions. In the first version, the ADC is pipelined to achieve a 2.5 GHz sampling rate and the inverter comparators are sized manually [40]. Using the dynamic offset suppression technique in the design of inverter comparators, the second version ADC improves a SFDR of 10 dB in the Nyquist bandwidth [41]. By optimization of dynamic offset suppression

and using a thermometer-Gray- binary encoding scheme, the third version ADC achieves an ENOB of 3 bits and a SFDR of 25 dB.

7.1.1 Implementation and Performance

This ADC is designed and simulated in 130 nm CMOS process using Cadence Spectre and VSDE 4.1 (Aptivia). Figure 7-1 is the schematic diagram of 4-b CDC-based pipelined ADC, which includes 15 CDC comparators, a TGB encoder and a clock tree. Figure 7-2 is the schematic of the TGB encoder, which include a Gray-to-binary encoding diagram (Figure 7-3). Figure 7-4 is the schematic of the clock tree. Figure 7-5 is the schematic of 15 CDC comparators, with an input voltage range of V_m from 0.45v to 0.688v and an optimum I_{ds} of 80μA.

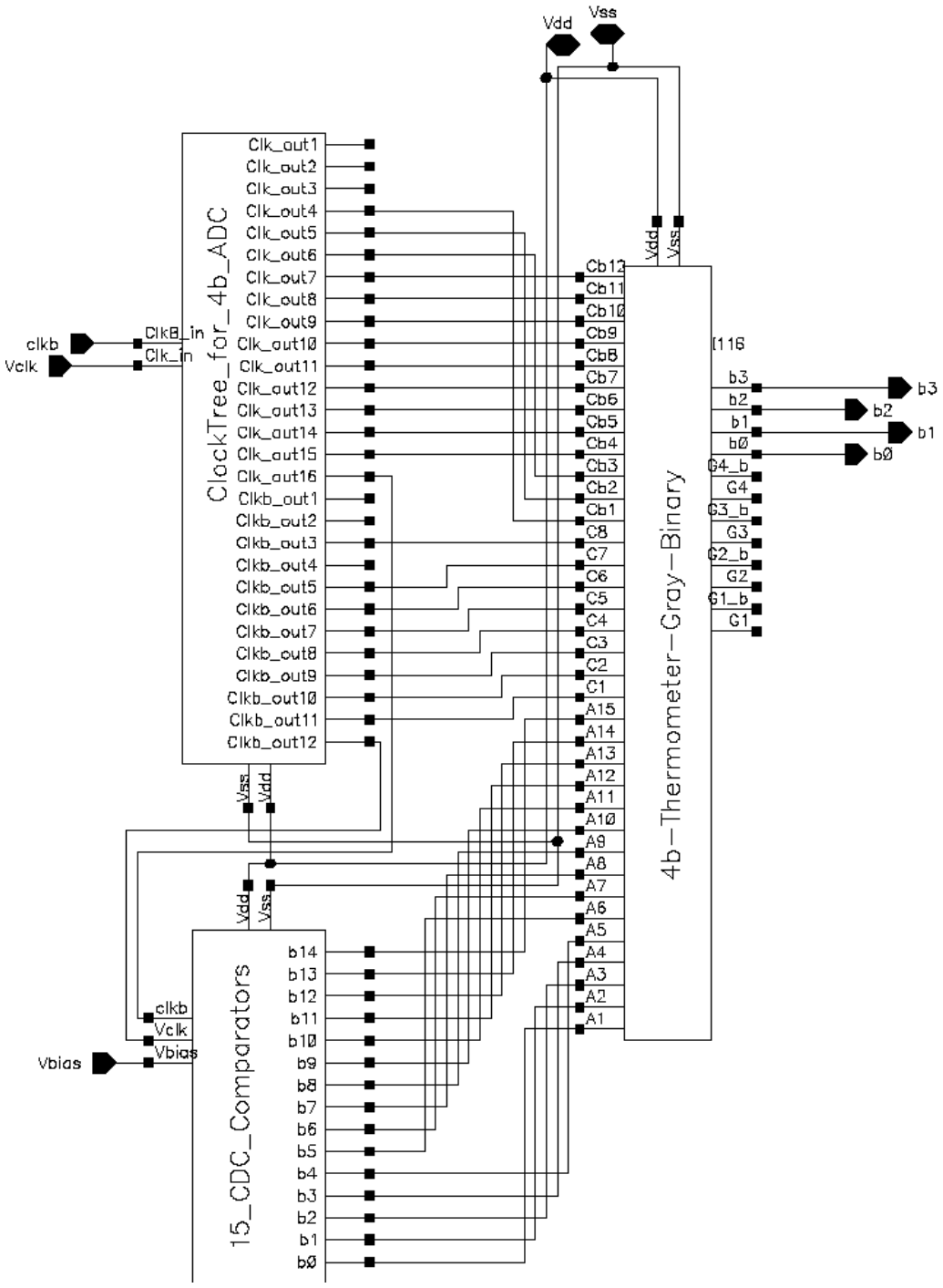


Figure 7-1 Schematic of 4-b 1.25-GHz Bandwidth 2.5-GSPS ADC

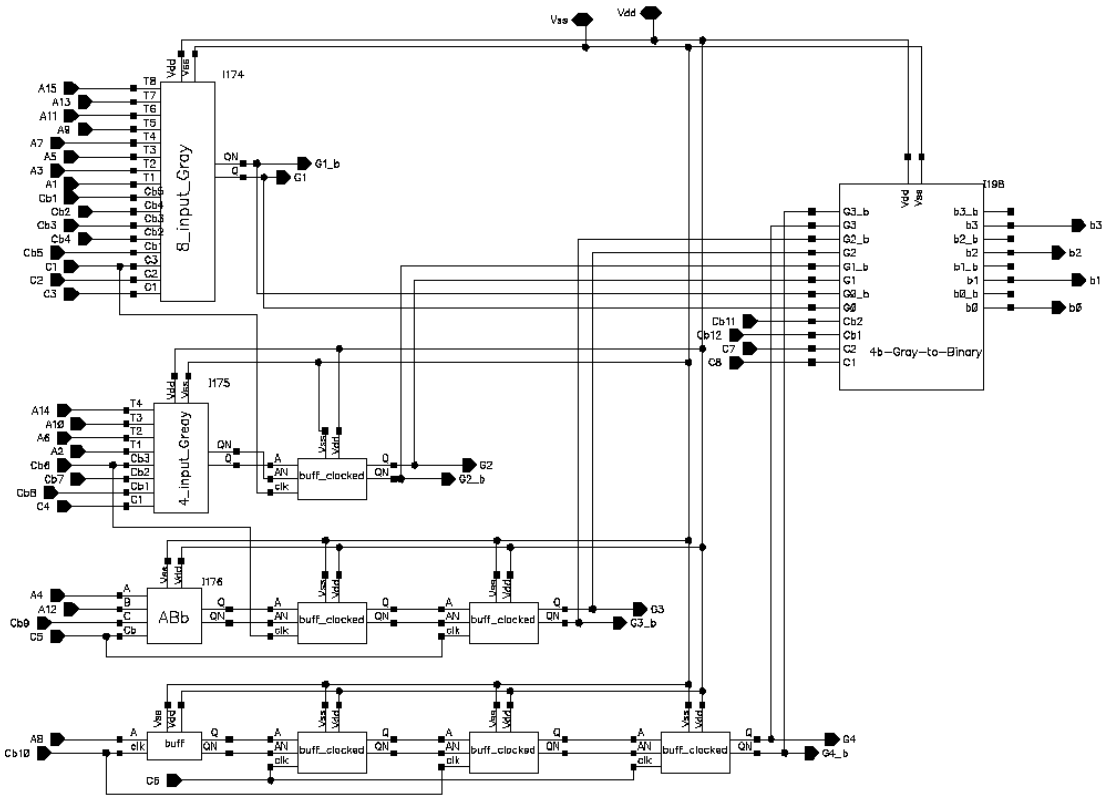


Figure 7-2 Schematic of TGB 4-b Encoder

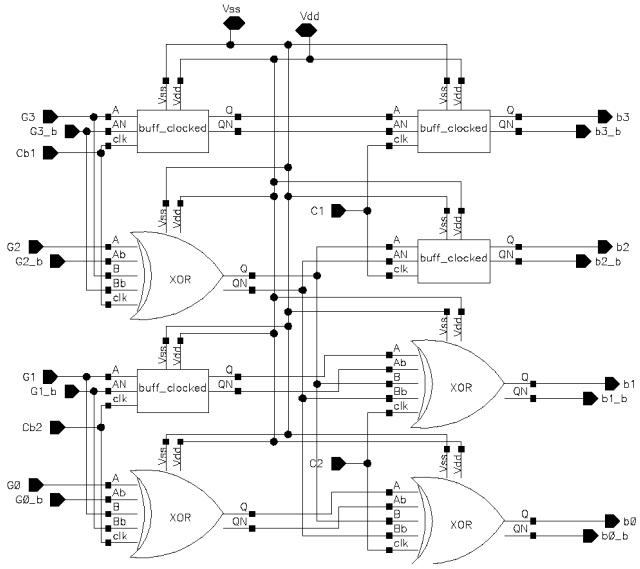


Figure 7-3 Schematic of Gray-to-Binary Encoding

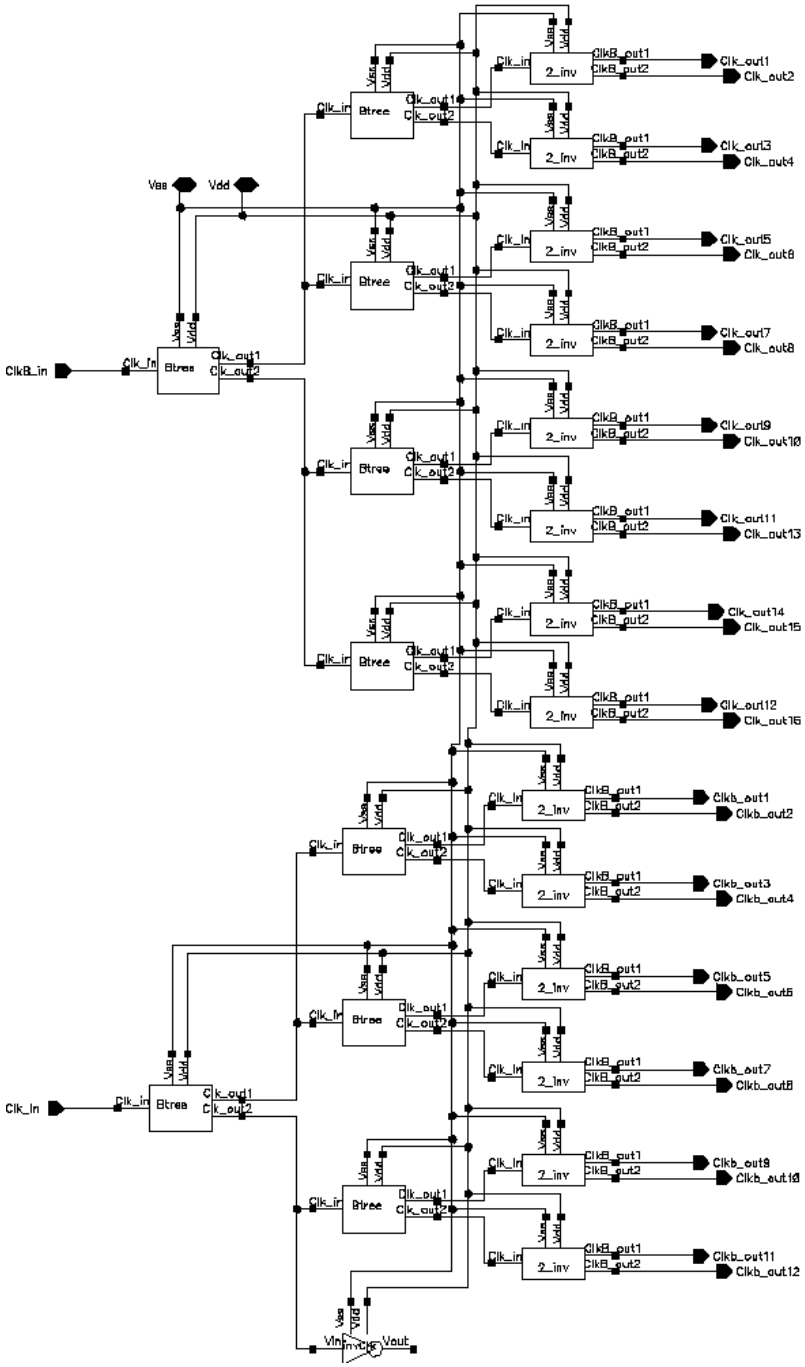


Figure 7-4 Schematic of Clock Tree for a 4-b ADC

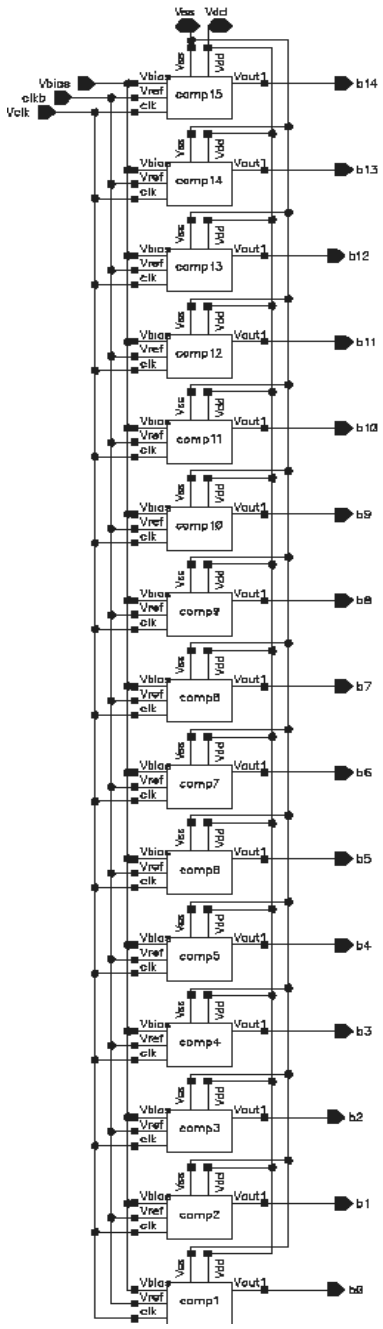


Figure 7-5 Schematic of 15 CDC Comparators

Figure 7-6 shows the simulated voltage transfer characteristics of 15 quantization inverters in 15 CDCs. The 15 quantization levels are equally-spaced. Figures 7-7 (a) and (b) show the simulated outputs V_{o1} and V_{o2} of the 15 CDCs (Figure 5-3) at an input signal frequency of 100 MHz. The outputs V_{o1} and V_{o2} of 15 comparators are nearly equally-spaced after the dynamic offsets suppressed, which warrant a small differential nonlinearity (DNL) error.

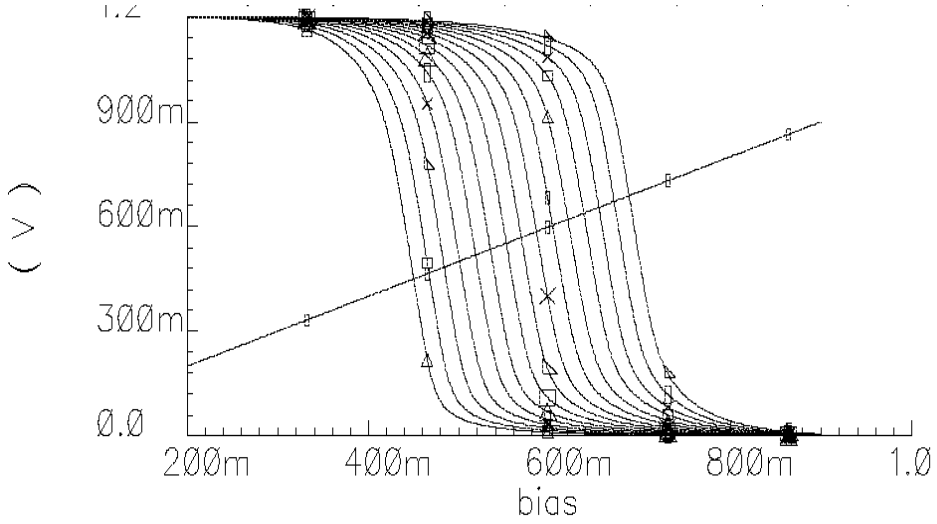
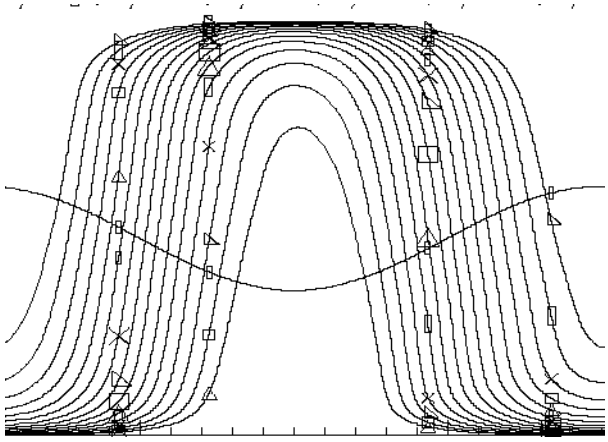
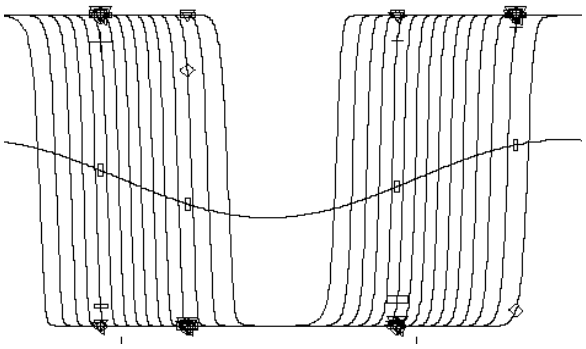


Figure 7-6 Outputs of 15 Quantization Inverters at DC



(a) output V_{o1}



(b) output V_{o2}

Figure 7-7　Outputs of CDC Comparators at 100MHz Sinusoidal Signal

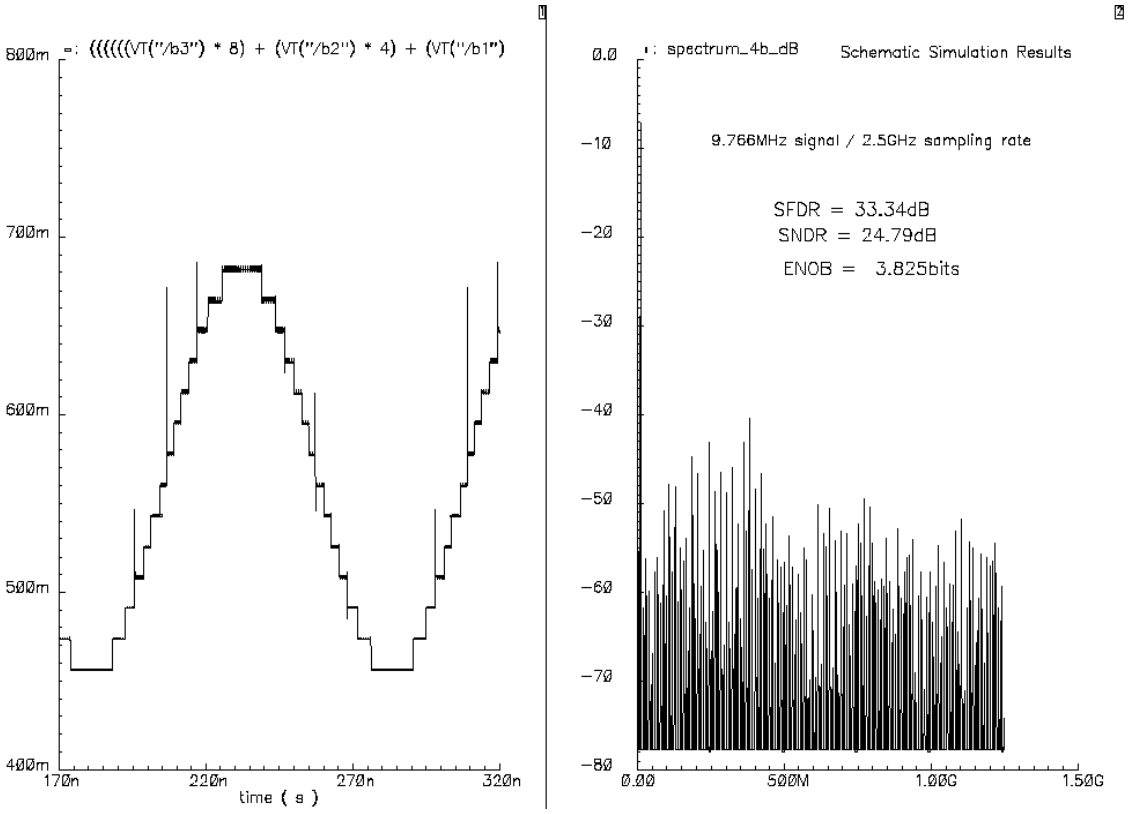


Figure 7-8　Reconstructed Signal and Spectrum of 9.766MHz Signal

Figure 7-8 is the reconstructed signal and the spectrum of 9.77MHz signal. The reconstructed signal figure shows the 15 comparators function correctly. The 4-b ADC achieves an ENOB of 3.8 bits and a SFDR of 33.34dB as shown in the spectrum figure. Figures 7-9, 7-10 and 7-11 show the reconstructed spectra for three input signal frequencies of 250 MHz, 952 MHz and 1.248 GHz. The 4-b ADC achieves (SFDR, ENOB) of (29.57 dB, 3.4 bits), (24.09 dB, 2.9 bits) and (22.25 dB, 2.8 bits), respectively.

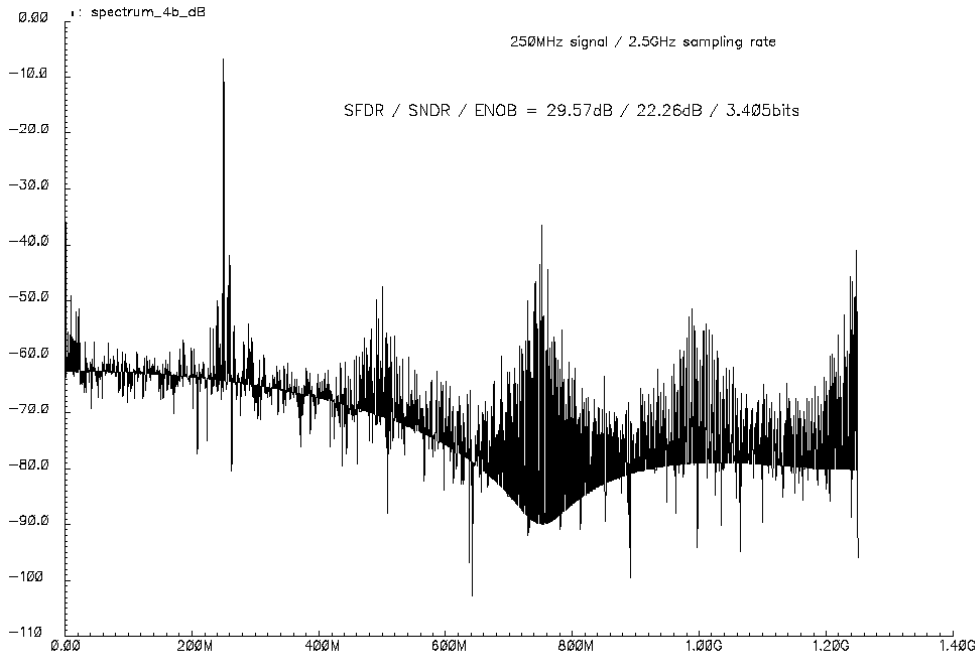


Figure 7-9 Spectrum of 250MHz Signal

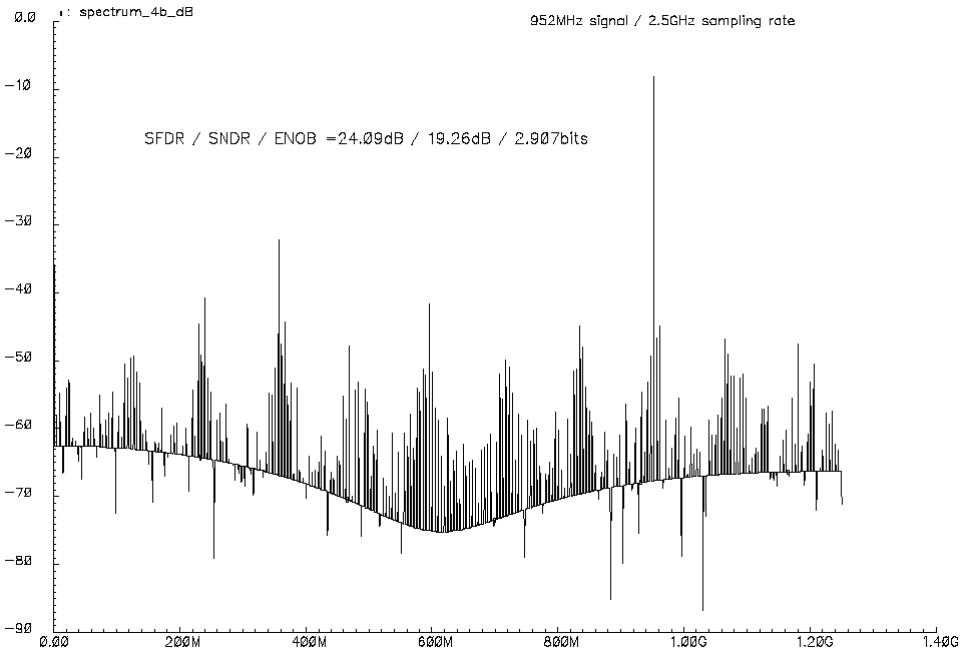


Figure 7-10 Spectrum of 952 MHz signal

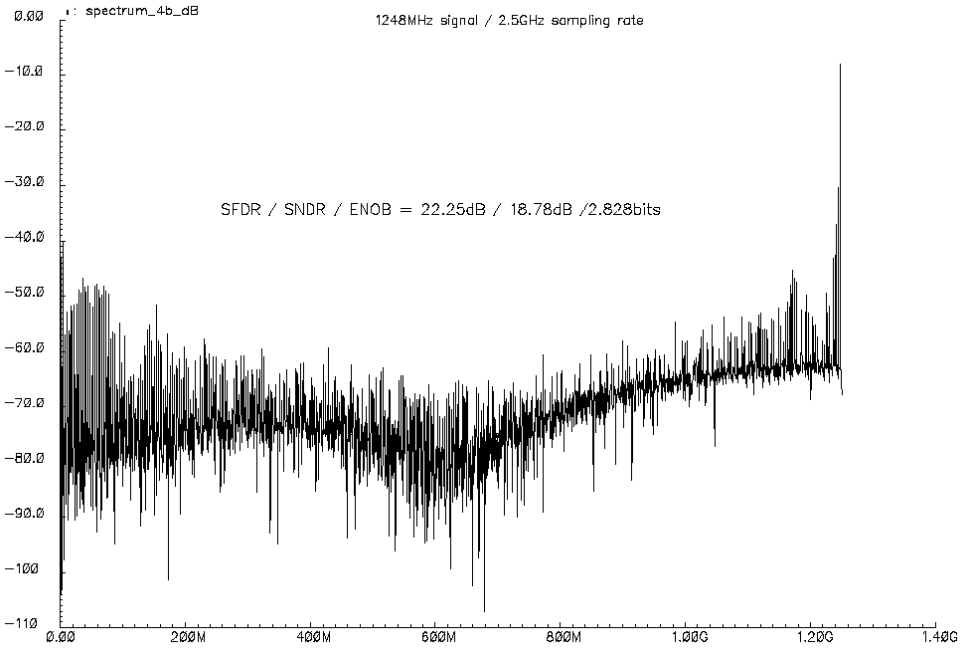


Figure 7-11 Spectrum of 1.248GHz Signal

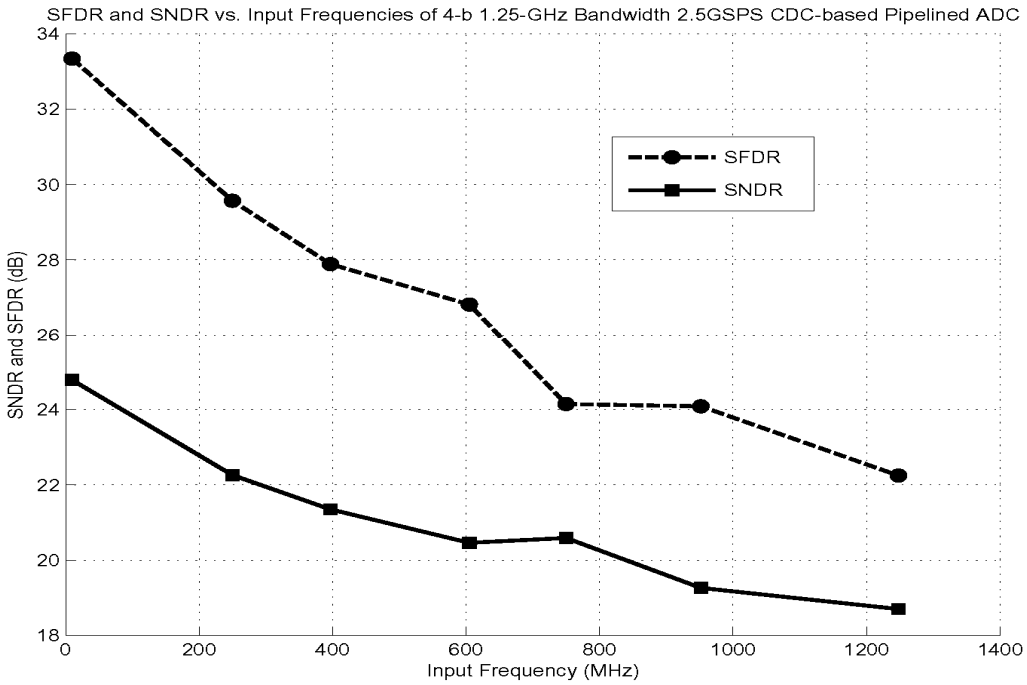


Figure 7-12 SFDR and SNDR vs. Input Frequencies

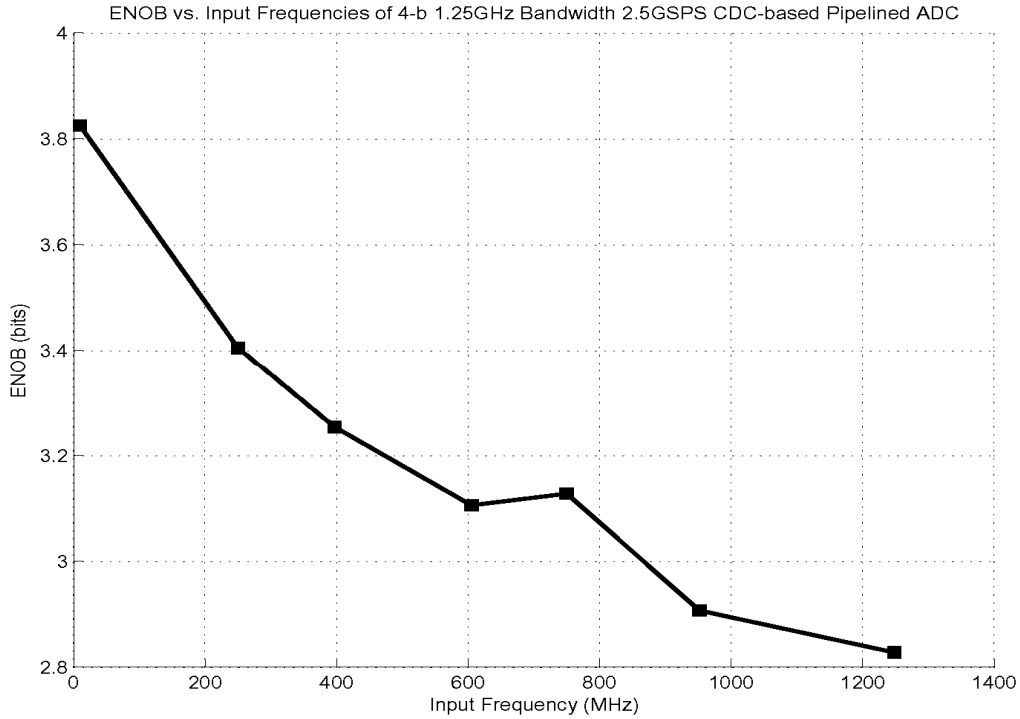


Figure 7-13 ENOB vs. Input Frequencies

Both SFDR and SNDR for input signal frequency up to 1.25 GHz at a 2.5GHz conversion rate are plotted in Figure 7-12. ENOB vs. input signal frequency is plotted in Figure 7-13. The SFDR of the 4-b ADC is above 22 dB and the ENOB is above 2.8 bits for the Nyquist bandwidth.

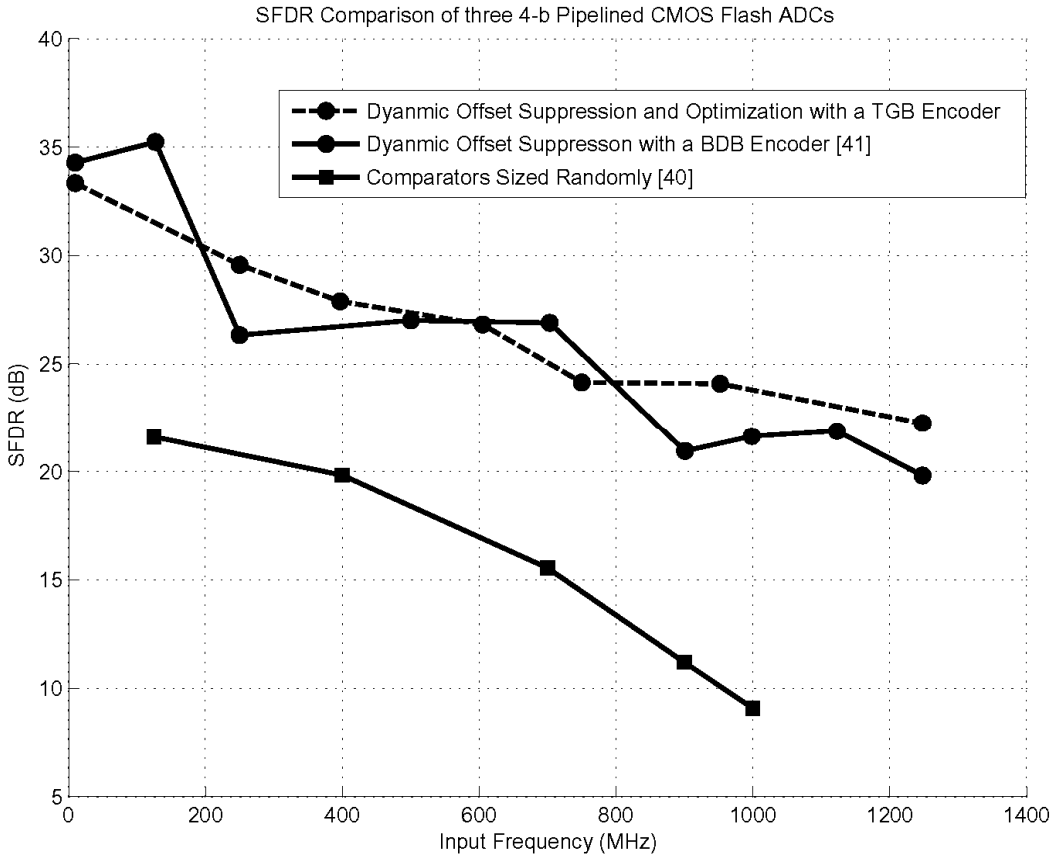


Figure 7-14 SFDR Comparison of three ADCs

Figures 7-14 and 7-15 compare SFDR and SNDR of the 4-b 1.25 GHz bandwidth 2.5 GSPS CDC-based pipelined ADC with its previous versions [40, 41]. Compared with the previous version ADC of dynamic offset suppression and BDB encoder [41], the SFDR is improved by 2 to 3 dB at input frequencies above 800 MHz as shown in Figure 7-14. The SNDR is also improved by 5 dB at input frequencies below 800 MHz, and 12 dB at input

frequencies above 800 MHz, as shown in Figure 7-15. The ENOB is improved almost by 1 bit at input frequencies below 800 MHz and 2 bits at frequencies above 800 MHz. Compared with the other version ADC with comparators sized manually [40], the SFDR is improved by 8 dB at low input frequencies and 13 dB at high input frequencies, as shown in Figure 7-14. The performance improvements are summarized in Table 7-2.

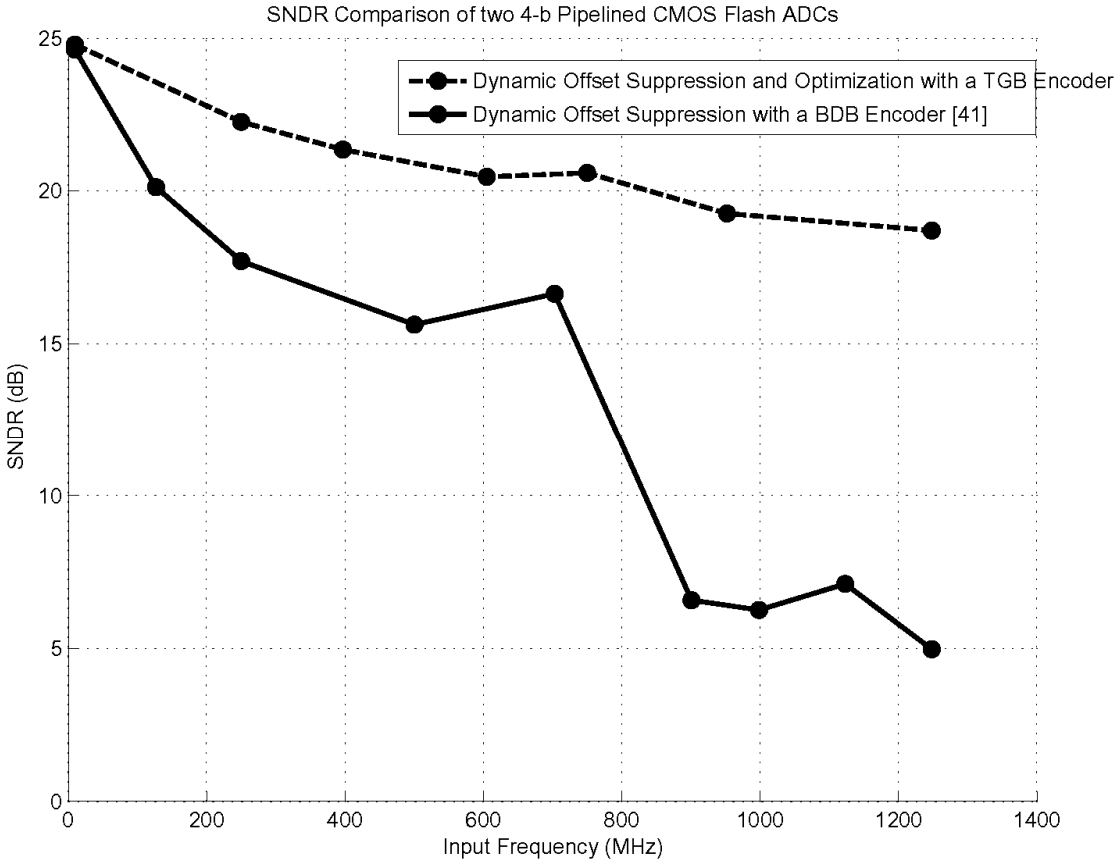


Figure 7-15 SNDR Comparison of two ADCs

Table 7-1 Dynamic Performance Improvement Rate of the ADC with Dynamic Offset Suppression and Design Optimization

CDC design method	SFDR Improvement		SNDR / ENOB Improvement	
	Low Frequency	High Frequency	Lower Than 800MHz	Higher Than 800MHz
Sizing Randomly	----	----		
Dynamic offset Suppression	36%	**144%**	----	----
Design Optimization	----	----	33%	**170%**

7.1.2 Schematic vs. Post Layout Performance

The layout of the 4-b 1.25 GHz bandwidth 2.5 GSPS CDC-based pipelined ADC in Figure 7-16 includes a clock tree on the top, 15 comparators in the middle, and a TGB encoder on the bottom. The transistor sizes of the ADC in this layout are same as their sizes in the schematic design. The area of layout is 220 μm x106 μm and the power is 7.9 mW for a 2.5 GHz clock.

The dynamic performance of the layout is reported in the following figures and compared with its schematic counterpart in the end of the section.

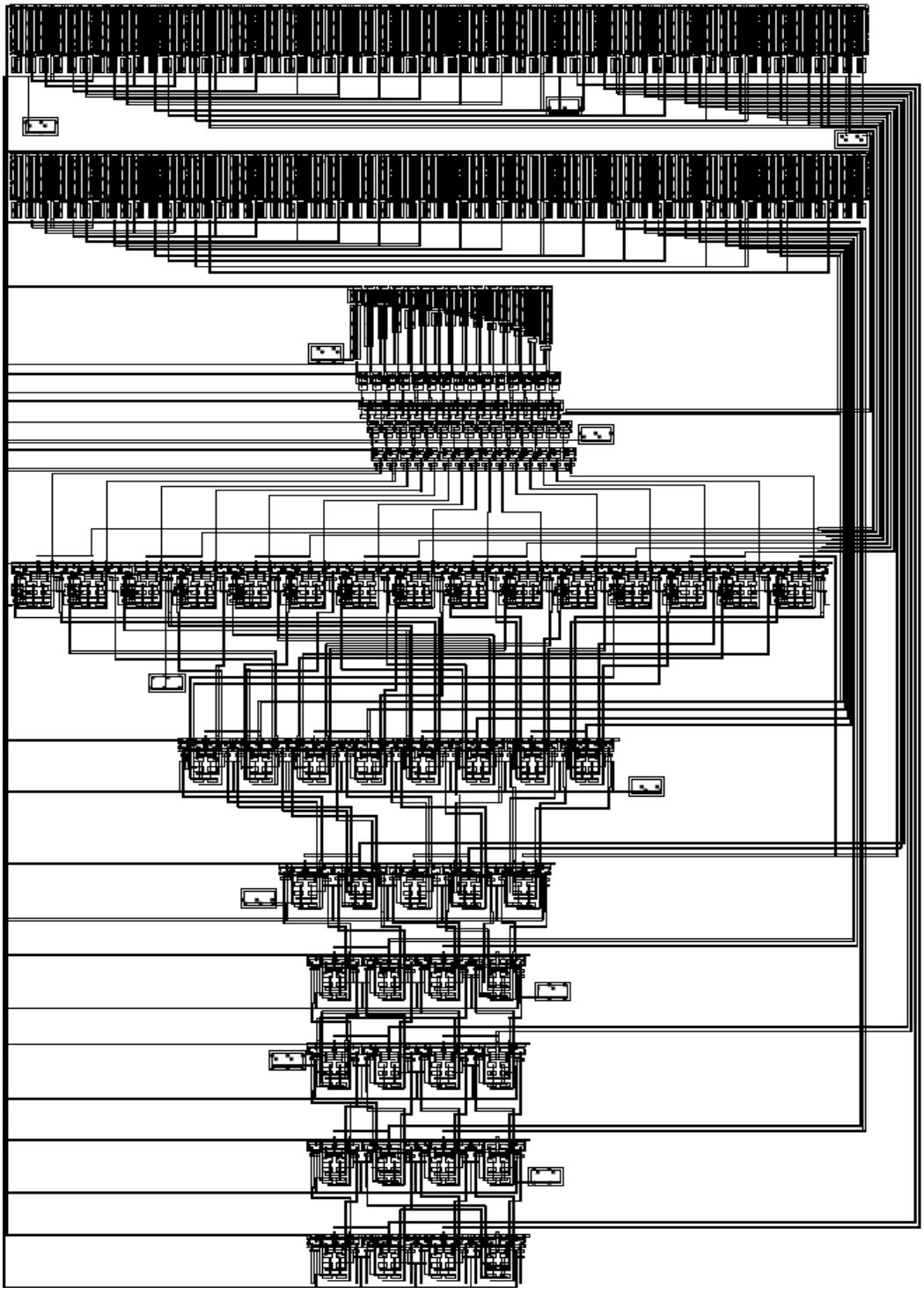


Figure 7-16 Layout of the ADC

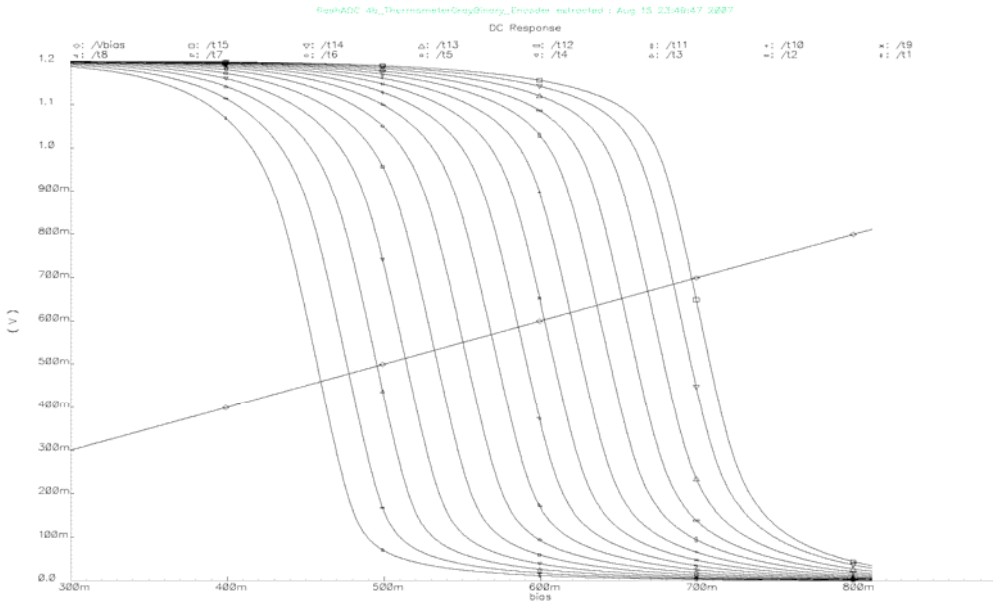


Figure 7-17 Outputs of 15 Quantization Inverters at DC Post Layout Simulation

The 15 quantization levels of the layout are shown in Figure 7-17 from the post layout DC simulation. The quantization levels of the first 14 comparators are almost equally spaced. The quantization level of the15th comparator is slightly close to the one of the 14th comparator.

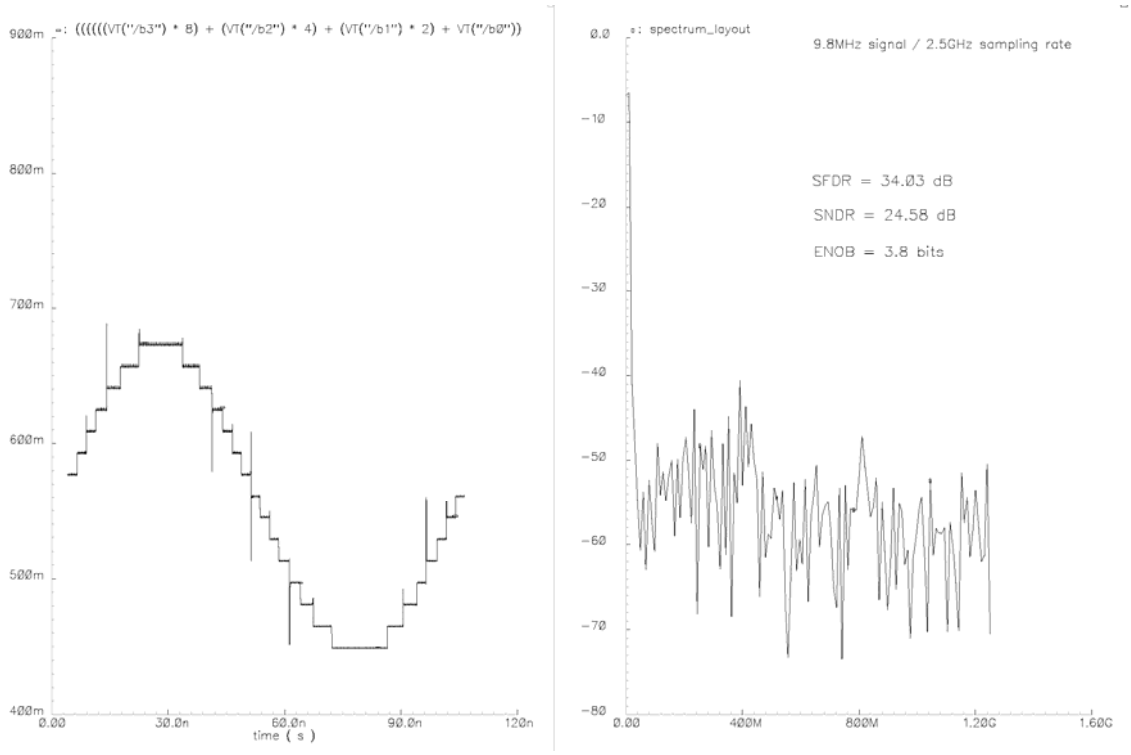


Figure 7-18 Reconstructed Signal and Spectrum of 9.8MHz signal

The 15 quantization levels of the layout are also presented in Figure 7-18 from the post layout transient simulation with a 9.8 MHz input signal. The 15 comparators function correctly and the ADC achieves a SFDR of 34.03 dB, a SNDR of 24.58 dB, and an ENOB of 3.8 bits. Comparing the layout performance (SFDR / SNDR / ENOB = 34.03 dB / 24.58 dB / 3.8 bits) (Figure 7-18) with the schematic performance (SFDR / SNDR / ENOB = 33.34 dB / 24.79 dB / 3.825 bits) (Figure 7-8), it is shown the dynamic performances of ADC of schematic and layout are close at an input signal of 9.8 MHz.

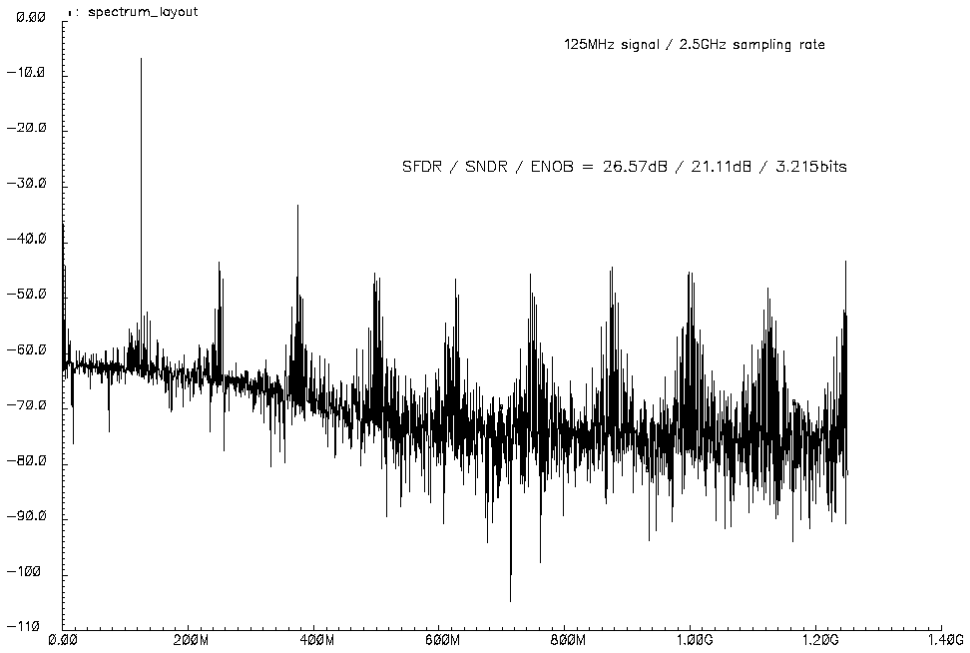


Figure 7-19 Spectrum of 125MHz Signal

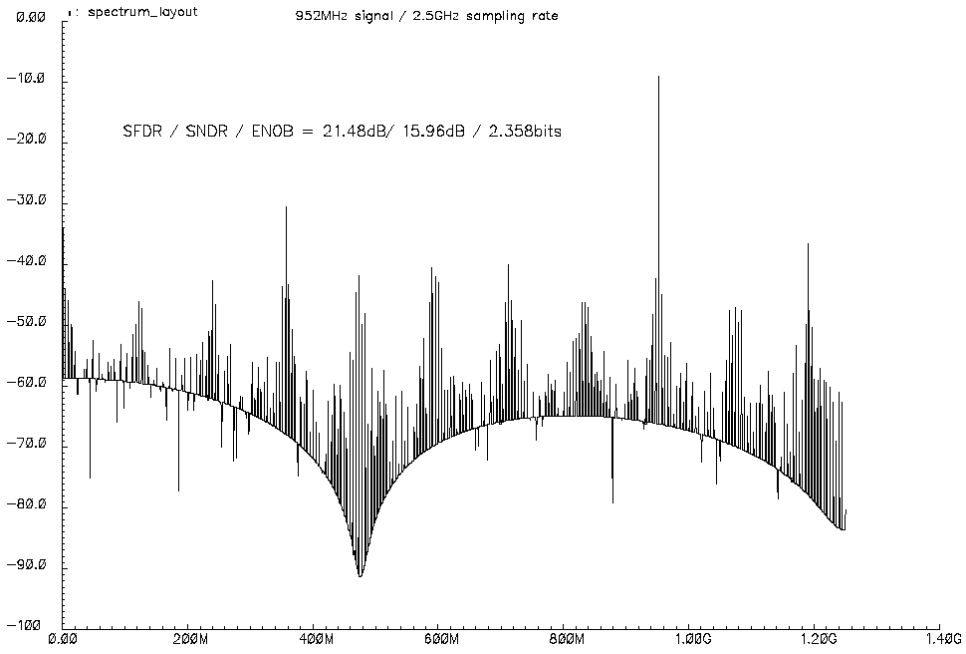


Figure 7-20 Spectrum of 952MHz Signal

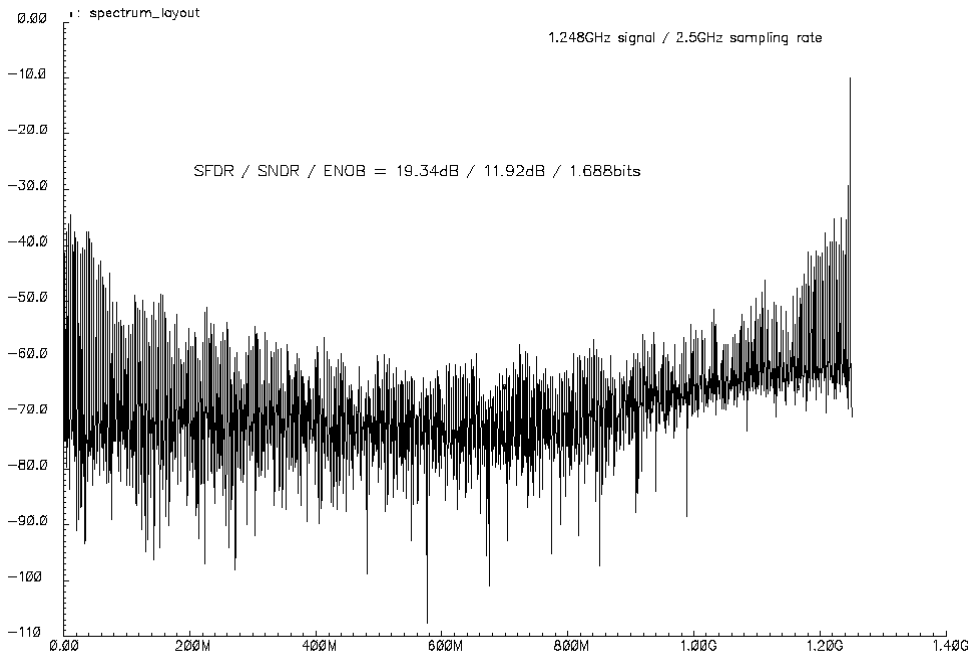


Figure 7-21 Spectrum of 1.248GHz Signal

Figures 7-19, 7-20 and 7-21 show the reconstructed spectra for input signal frequencies of 150 MHz, 952 MHz and 1.248 GHz. The ADC achieves (SFDR, ENOB) of (26.57 dB, 3.215 bits), (21.48 dB, 2.358 bits) and (19.34 dB, 1.7 bits), respectively.

Comparing the performance of SFDR and ENOB of the layout at input signal frequencies 952 MHz and 1.248 GHz shown in Figures 7-20 and 7-21 with the ones of the schematic shown in Figures 7-10 and 7-11, it is shown that the SFDR is decreased by 3.5 dB and 2.9 dB and the ENOB is decreased by 0.5 bit and 1 bit at input signal of 952 MHz and 1.248 GHz respectively. The results are summarized in Table 7-2.

Table 7- 2 Performance Comparison of Schematic and Layout
at 952 MHz and 1.248 GHz

Signal Freq. (MHz)	Schematic		Layout		Degradation	
	SFDR (dB)	ENOB (bits)	SFDR (dB)	ENOB (bits)	SFDR (dB)	ENOB (bits)
952	24.09	2.907	21.48	2.358	3.5 (14%)	0.55 (17%)
1248	22.25	2.828	19.34	1.7	2.9 (13%)	1.13 (35%)

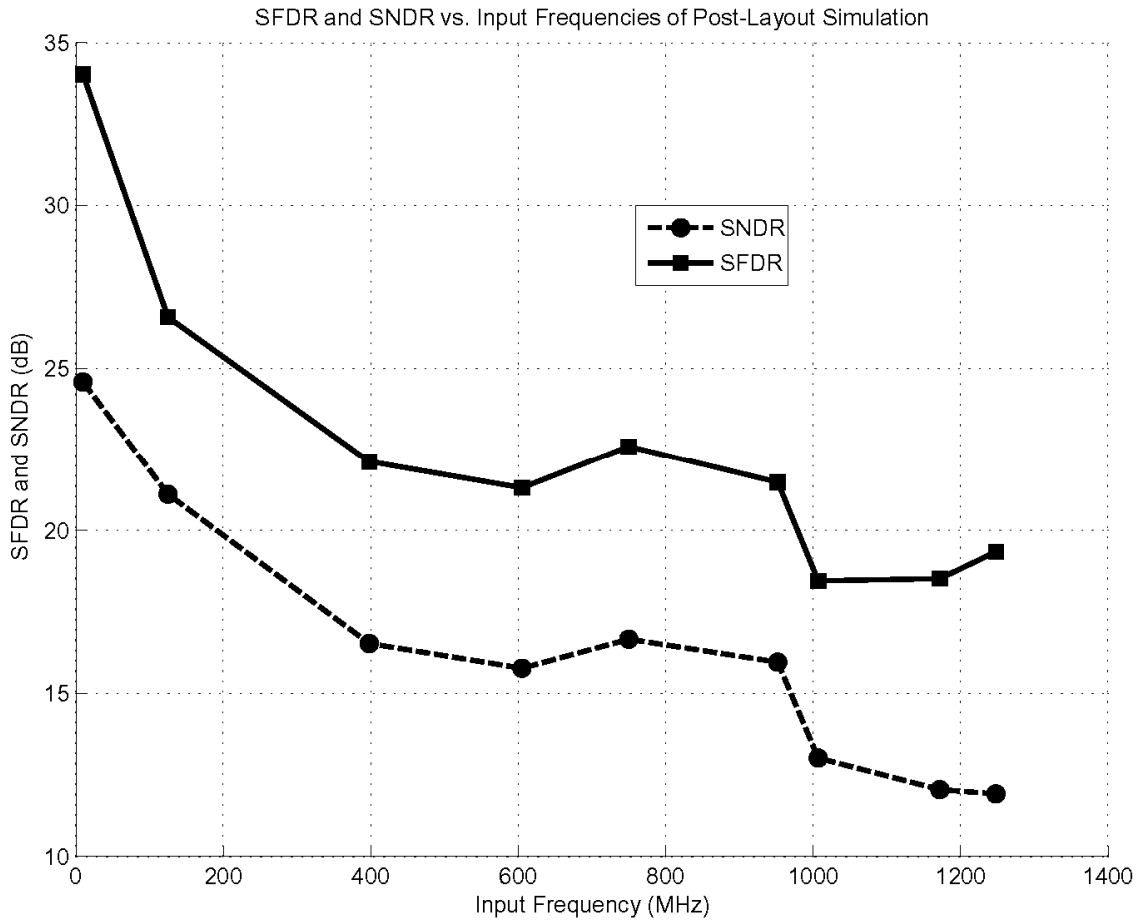


Figure 7-22 SFDR and SNDF vs. Input Frequencies

Both SFDR and SNDR of the layout for input signal frequency up to 1.25 GHz at 2.5 GHz conversion rate are plotted in Figure 7-22. The plot shows that SFDR is above 18 dB and SNDR above 12 dB up to the Nyquist frequency.

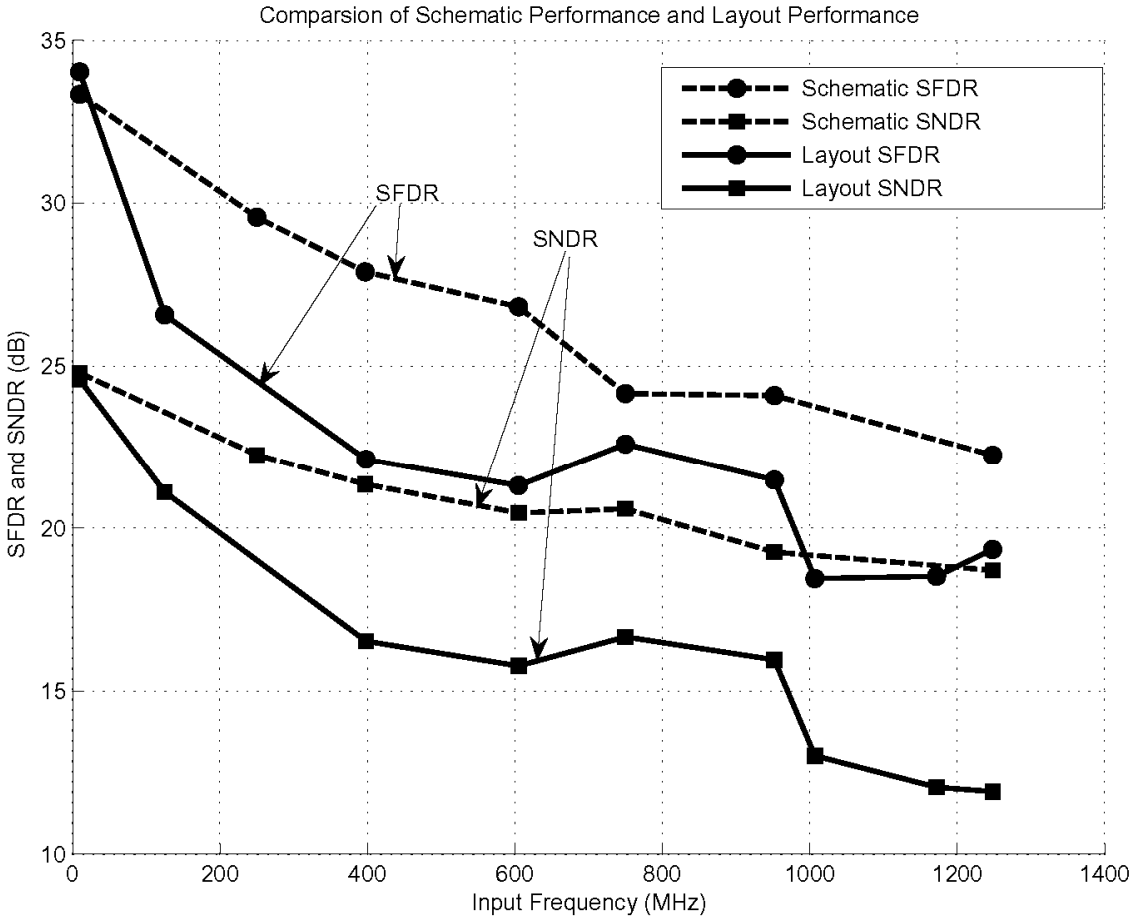


Figure 7-23 SFDR and SNDR Comparison between Schematic and Layout

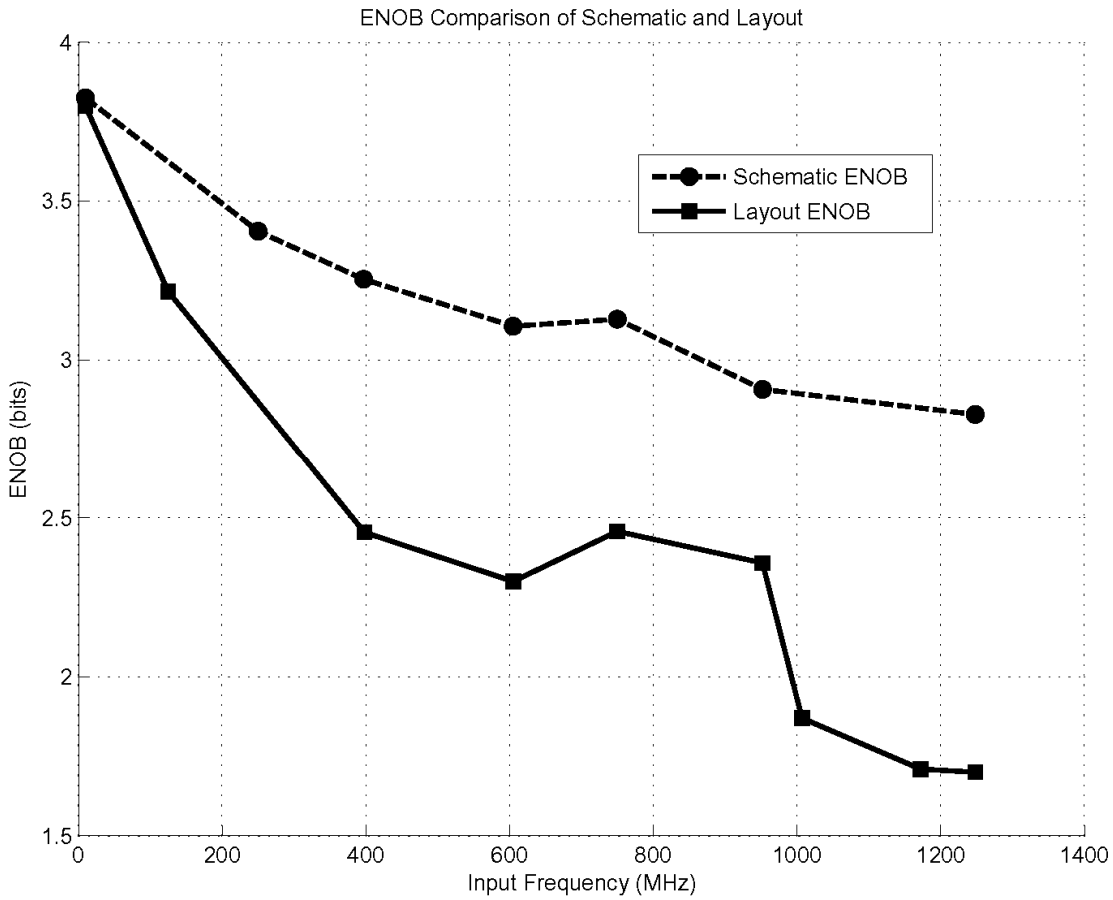


Figure 7-24 ENOB Comparison between Schematic and Layout

Figures 7-23 and 7-24 compare the layout performance of the 4-b 1.25 GHz bandwidth 2.5 GSPS CDC-based pipelined ADC with its schematic counterpart. The SFDR and SNDR of the layout both degrade about an average 5 dB compared to the schematic. That means the ENOB degrades about 0.8 bit. That's about 22% SFDR degradation and 28% SNDR and ENOB degradation. From Figures 7-23 and 7-24, the degradation increases with the input frequency exceeds 1 GHz. The maximum degradation of SNDR and ENOB is 35% at 1.248 GHz, calculated in Table 7-1. The performance degradation is summarized in Table 7-3.

Table 7-3 Dynamic Performance Degradation Rate of the Layout Compared to the Schematic of the ADC

ADC Implementation	SFDR Degradation		SNDR/ENOB Degradation	
	Lower Than 1 GHz	Higher Than 1 GHz	Lower Than 1 GHz	Higher Than 1 GHz
Schematic	----	----	----	----
Layout	22%	20%	22%	35%

7.2 4-b 2-GHz Bandwidth 4GSPS ADC

The maximum 3-dB frequency of the comparators in 130nm IBM digital process is approximated to 2-GHz in Chapter 5 during the optimal analysis. In Chapter 6, the maximum speed of the digital circuit is 5-GHz. By the Nyquist theory, a 4-GSPS is required for the bandwidth as 2-GHz. Thus this ADC is a special case of 4-b ADCs using IBM 130nm digital process due to its bandwidth at the highest limitation. The implementation and performance of the ADC are presented in this section, which is followed by performance comparison of the ADC with 4-b ADCs in literature.

7.2.1 Implementation and Performance

This ADC is designed and simulated using Cadence Spectre and VSDE 4.1 (Aptivia). Figure 7-25 is the schematic diagram of 4-b 2-GHz 4-GSPS CDC-based pipelined ADC, which includes a set of CDC comparators, a TGB encoder and a clock tree. The schematic of the TGB 4-b encoder is the same as the previous ADC shown in Figure 7-2. The schematic of the clock tree in the ADC is the same as the one shown in Figure 7-4. The schematic of 15 CDC comparators is the same as the one in Figure 7-5, with a different input voltage range of V_m from 0.501v to 0.62v and an optimum I_{ds} 150μA.

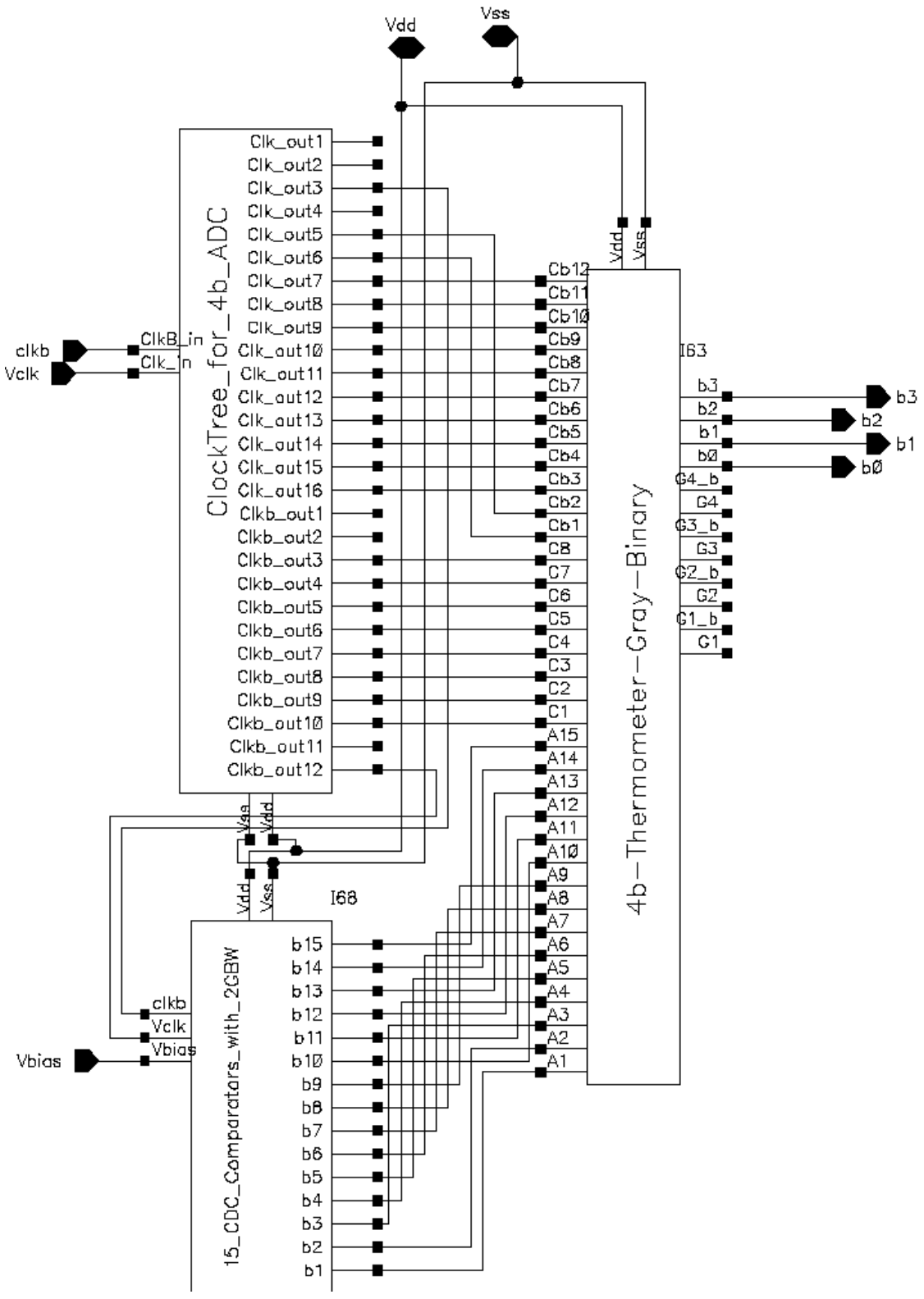


Figure 7-25 Schematic of 4-b 2-GHz Bandwidth 4-GSPS ADC

Figure 7-26 to Figure 7-30 show the reconstructed spectra for various frequency input signals and the ADC achieves SFDR, SNDR and ENOB at those frequencies presented in Table 7-4.

Table 7-4 Dynamic Performance Parameters of Various Frequency Signals

Sig. freq. (MHz)	200.2	1374	1600	1807	1997
SFDR (dB)	33.01	28.42	26.9	29.28	24.99
SNDR (dB)	23.8	21.08	20.05	19	18.6
ENOB (dB)	3.66	3.209	3.03	2.864	2.798

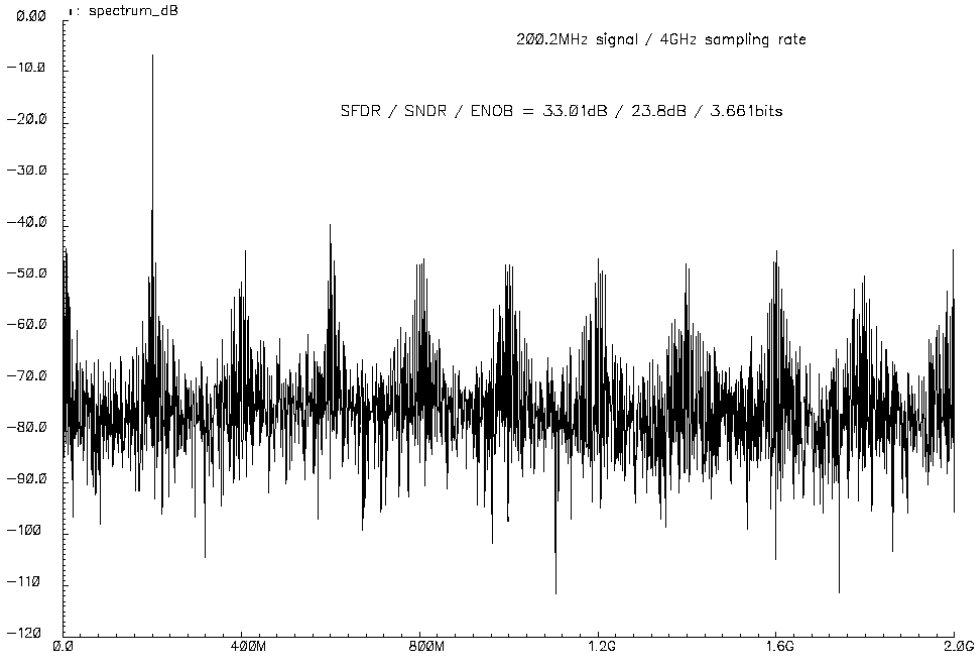


Figure 7-26 Spectrum of 200MHz Signal

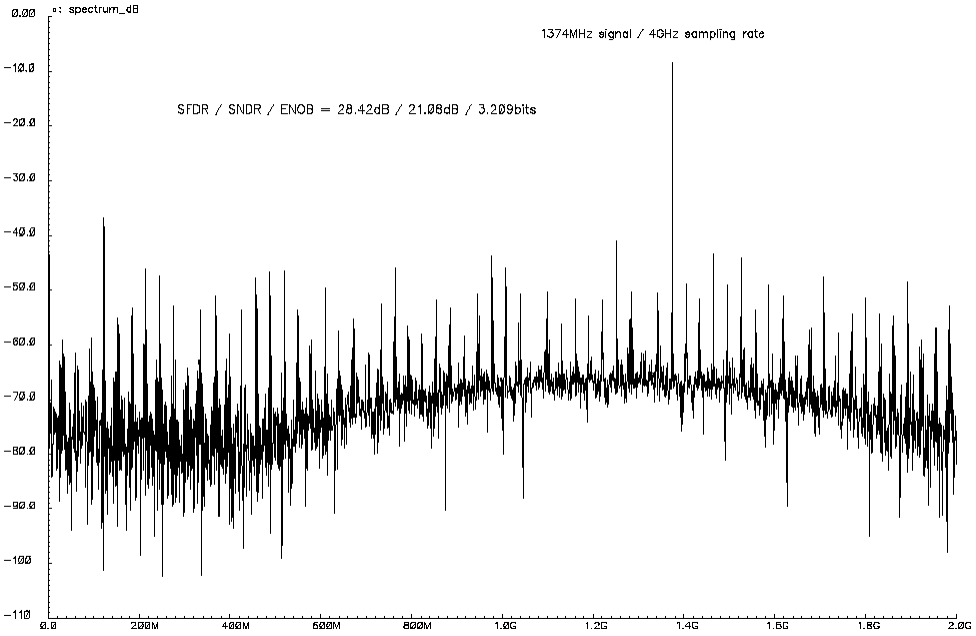


Figure 7-27 Spectrum of 1.374GHz Signal

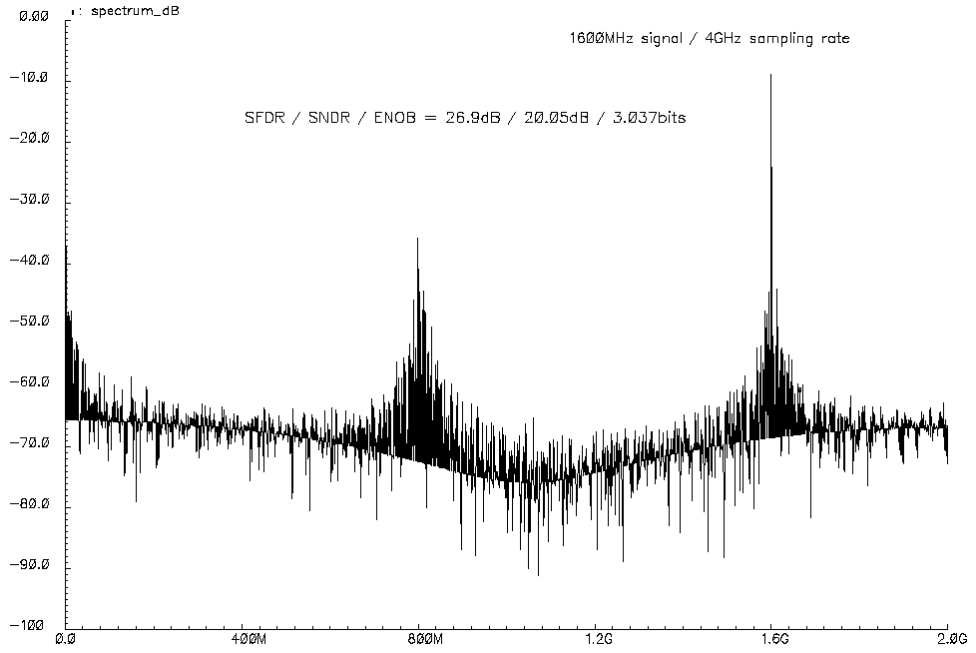


Figure 7-28 Spectrum of 1.6GHz Signal

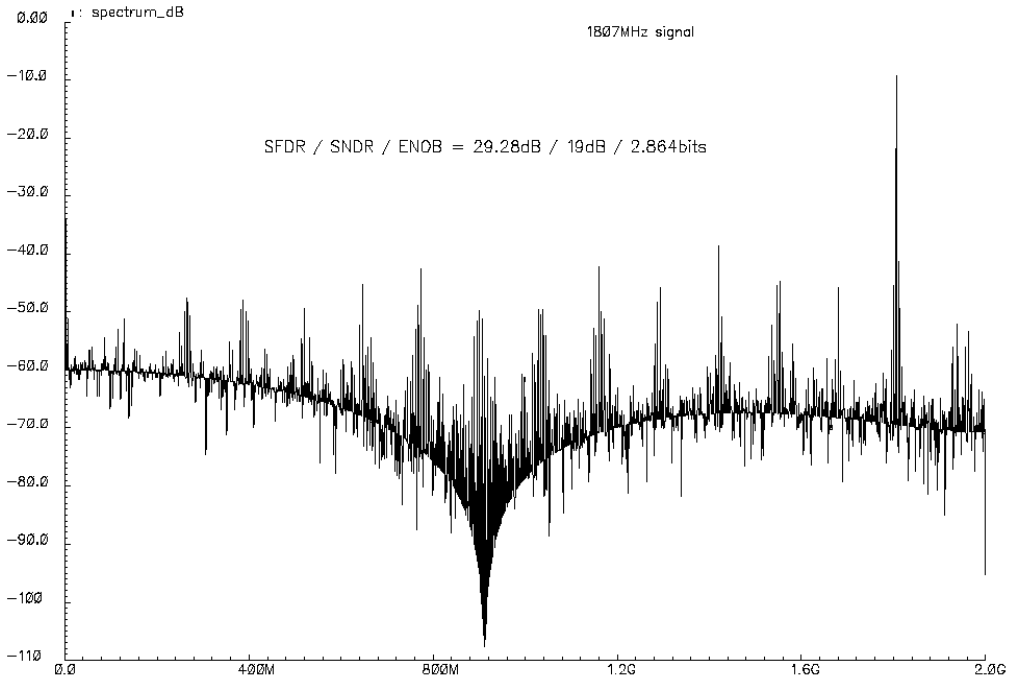


Figure 7-29 Spectrum of 1.807GHz Signal

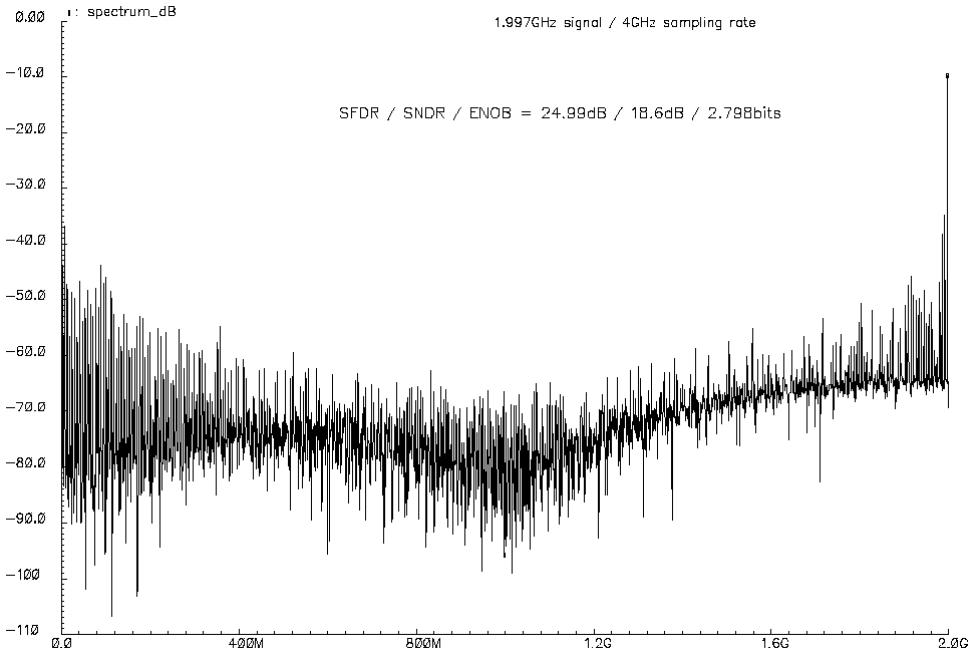


Figure 7-30 Spectrum of 1.997GHz Signal

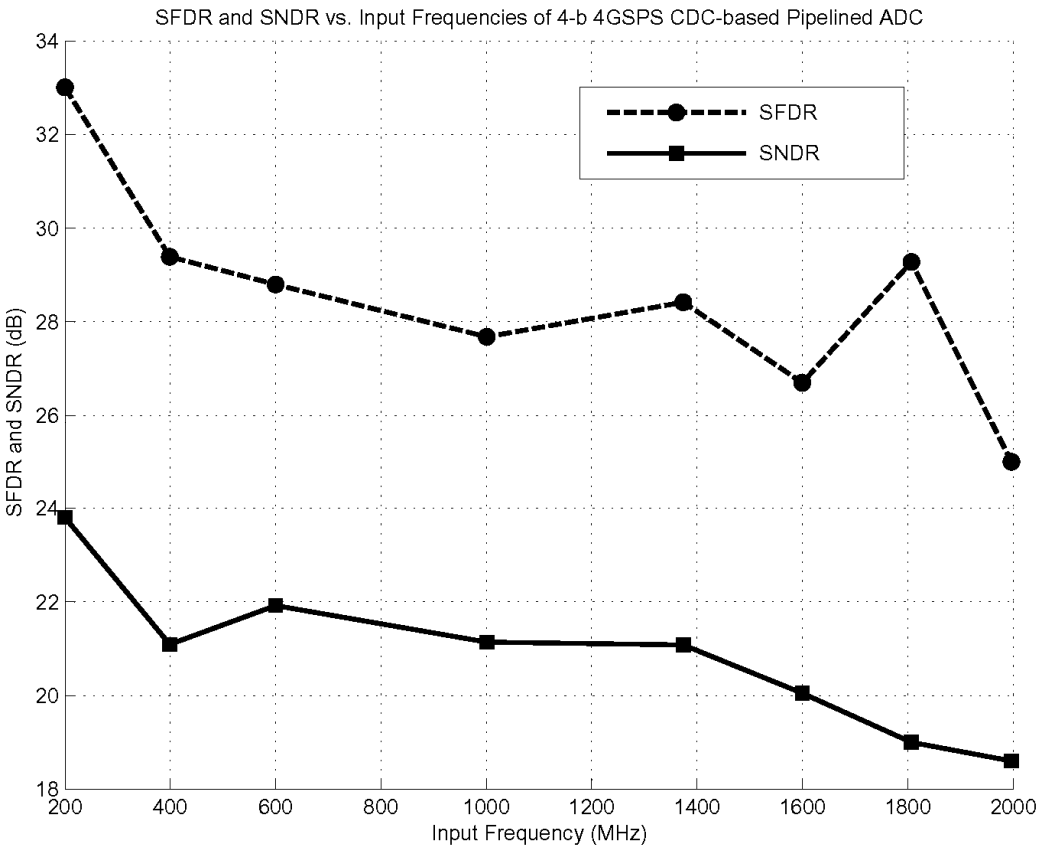


Figure 7-31 SFDR and SNDR vs. Input Frequencies

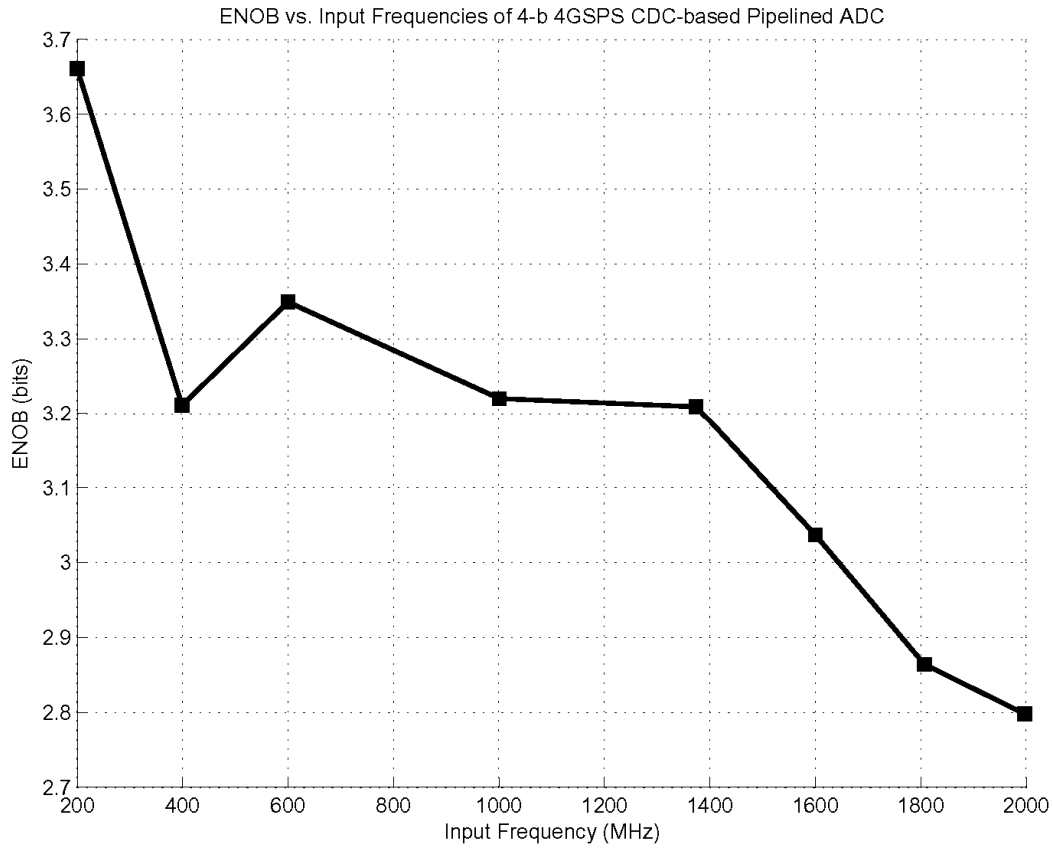


Figure 7-32 ENOB vs. Input Frequencies

Both SFDR and SNDR for input signal frequency up to 2 GHz at 4 GHz conversion rate are plotted in Figure 7-31. ENOB for input signal frequency is plotted in Figure 7-32. The plots show that, with 4 GHz sampling frequency, SFDR of the ADC is stable at 28 dB and ENOB above 3.2 bits with frequency up to 1.4 GHz. The ENOB is above 3.0 bits with frequency up to 1.6 GHz, and above 2.8 bits for the Nyquist bandwidth.

7.2.2 Comparison with 4-b Flash ADCs in Literature

Table 7-5 Comparison with 4-b Flash ADCs in Literature

Ref. (year)		Proposed ADC	[57] '05	[59] '04	[58] '99
Primary Character-istics	Res. (bits)	4	4	4	4
	GSPS (GHz)	4	5	3.2	12
	Tech. (μm)	0.13	0.18	0.18	0.25
	Vdd (V)	1.2	1.8	1.8	2.5
Performance Parameters	ERBW (MHz)	2000	--	1300	--
	SFDR (dB)	33@200MHz 25@2GHz	--	--	--
	ENOB (bits)	3.7@200MHz 3.2@1.37GHz 2.8@2GHz	3.65@DC	3.6@DC 2.4@1.3GHz	3.34@DC
	Power (mW)	11	70	130	1000
	Area (mm^2)	0.022	0.2	0.4	9
Architecture		monolithic	monolithic	2-way time-interleaved	8-way time-interleaved
Digital Calibration		no	no	yes, off-chip	yes, on-chip

The proposed 4-b ADC implemented in the process with smaller feature size, compared with three referenced ADCs in Table 7-5, with wider bandwidth, lower power and area consumption in a monolithic architecture with no need of digital calibration. The overall performance is superior over the other 4-b ADCs in the table.

7.3 6-b 4GSPS ADC

After the validity of the comparator dynamic offset suppression and design optimization is proved in the section 7.1 with a 4-b 2.5-GSPS ADC, a 4-b 4-GSPS ADC with a higher speed is implemented and presented in the section 7.2. In this section, a 6-b 2.5-GSPS ADC with a higher resolution is implemented in 130nm IBM digital process. Theoretically, this 6-b ADC can be sampled in a clock signal up to 4-GHz.

7.3.1 Implementation and Performance

This ADC is designed and simulated using Cadence Spectre and VSDE 4.1 (Aptivia). Figure 7-33 is the schematic diagram of 6-b 2.5-GSPS CDC-based pipelined ADC, which includes a set of CDC comparators, a TGB encoder and a clock tree. The schematic of the TGB 6-b encoder is shown in Figure 7-34, which includes a 6-b Gray-to-binary encoding as in Figure 7-35. The schematic of the clock tree in the ADC is shown in Figure 7-37, which is much larger than the clock tree for 4-b ADCs. The schematic of 63 CDC comparators is the same as the one in Figure 7-36, with an input voltage range of V_m from 0.445v to 0.693v with a quantization step size (LSB) equal to 4mv. The CDCs are sized with I_{ds} as 80μA.

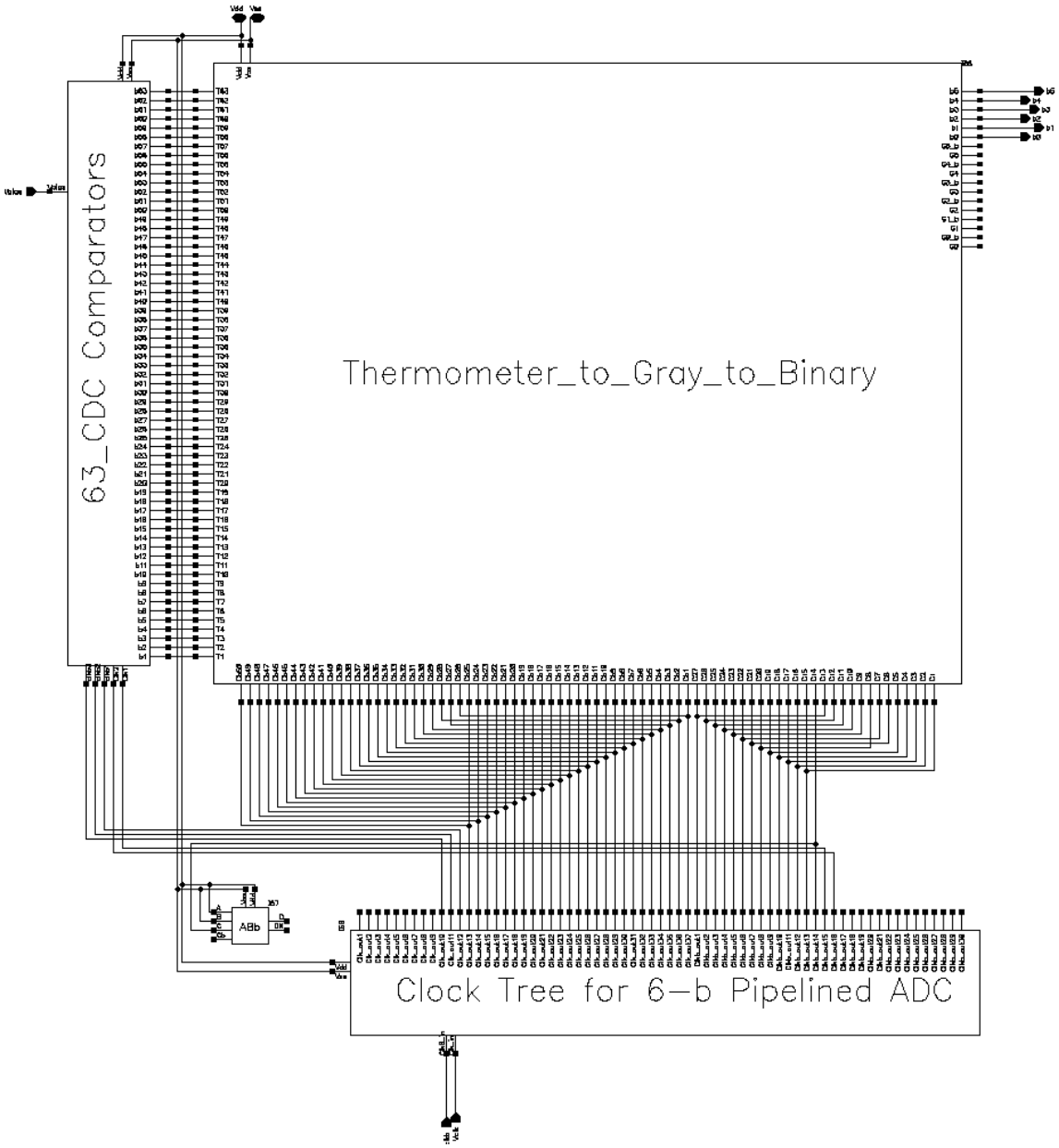


Figure 7-33 Schematic of 6-b 2.5GSPS ADC

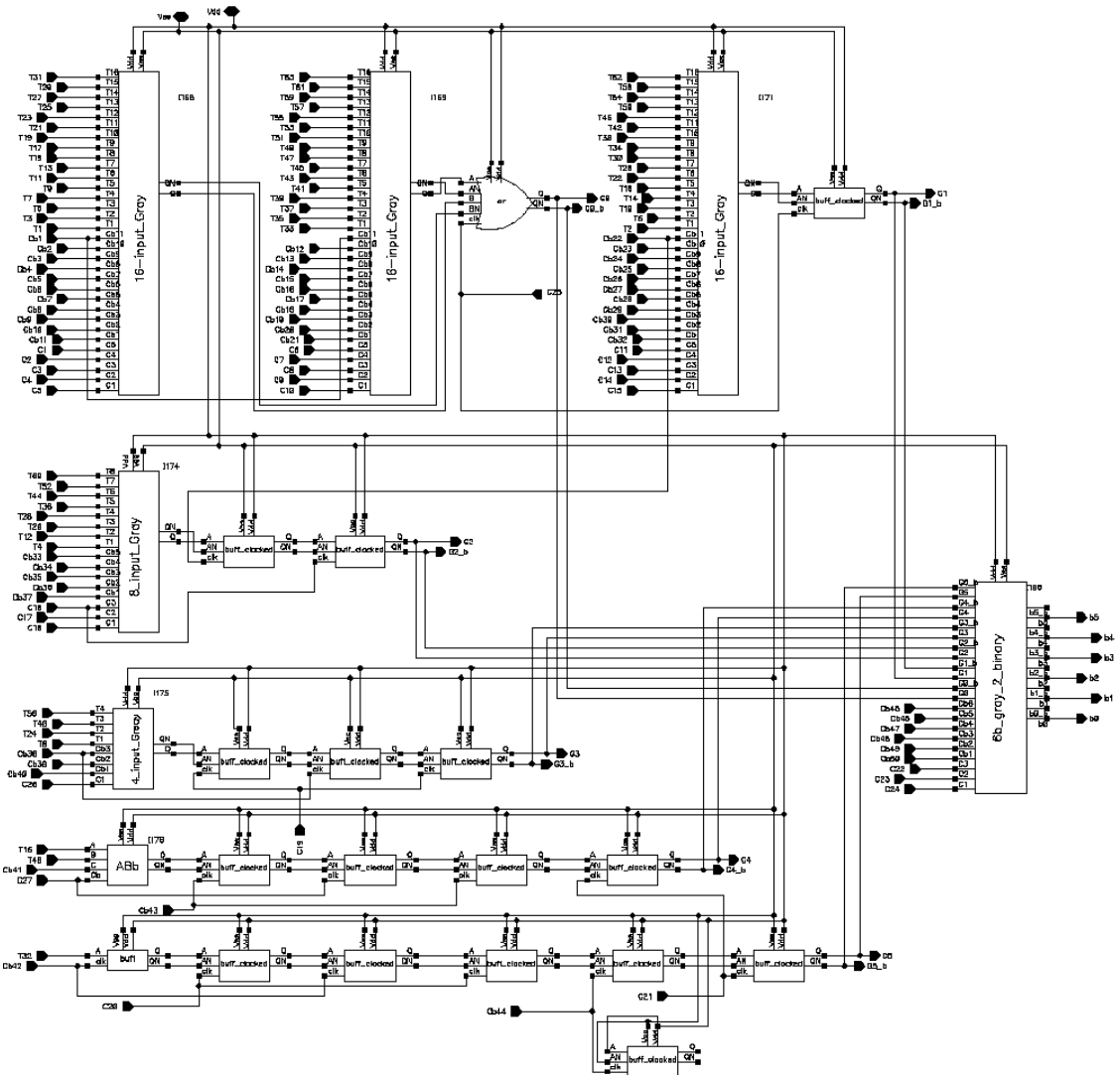


Figure 7-34　Schematic of 6-b TGB Encoder

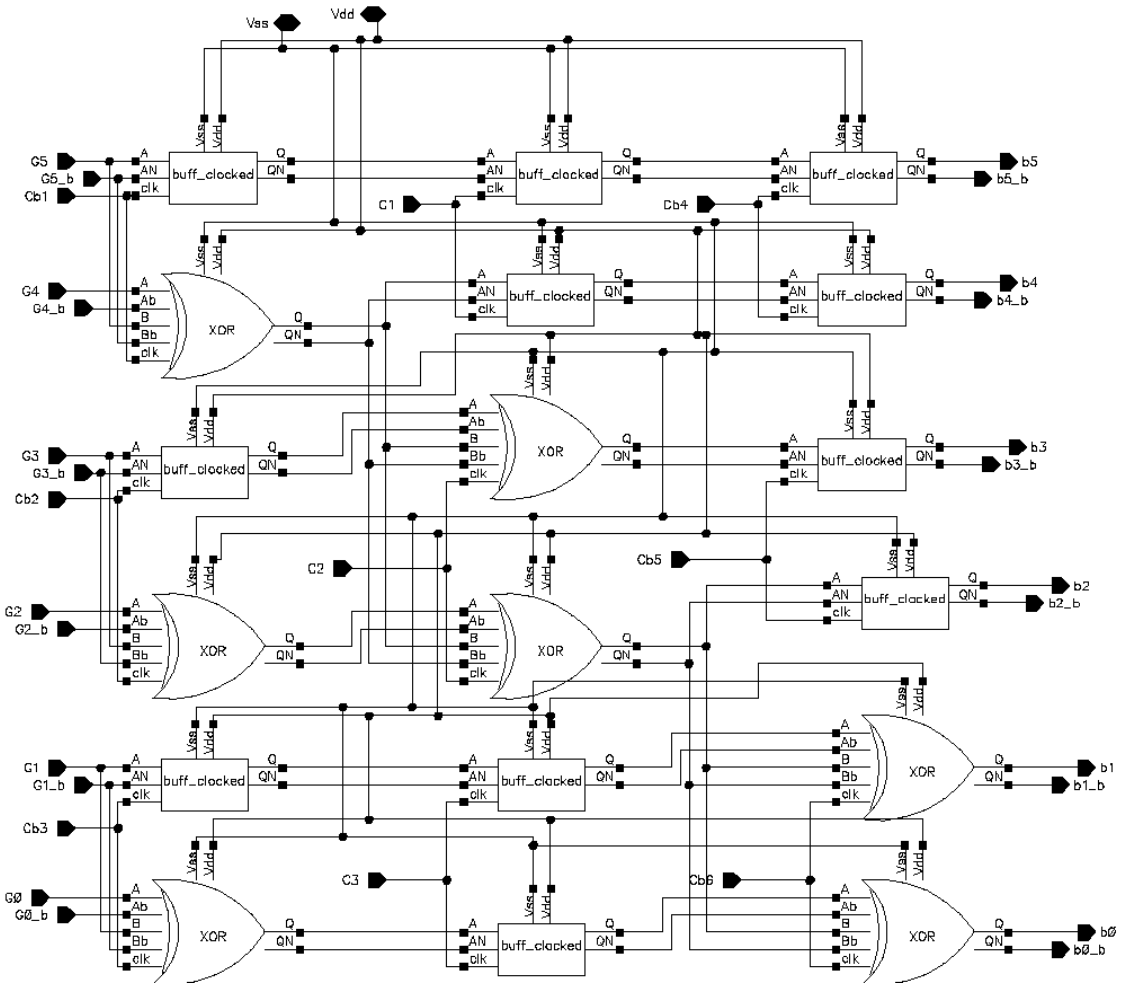


Figure 7-35 Schematic of 6-b Gray-to-Binary Encoding

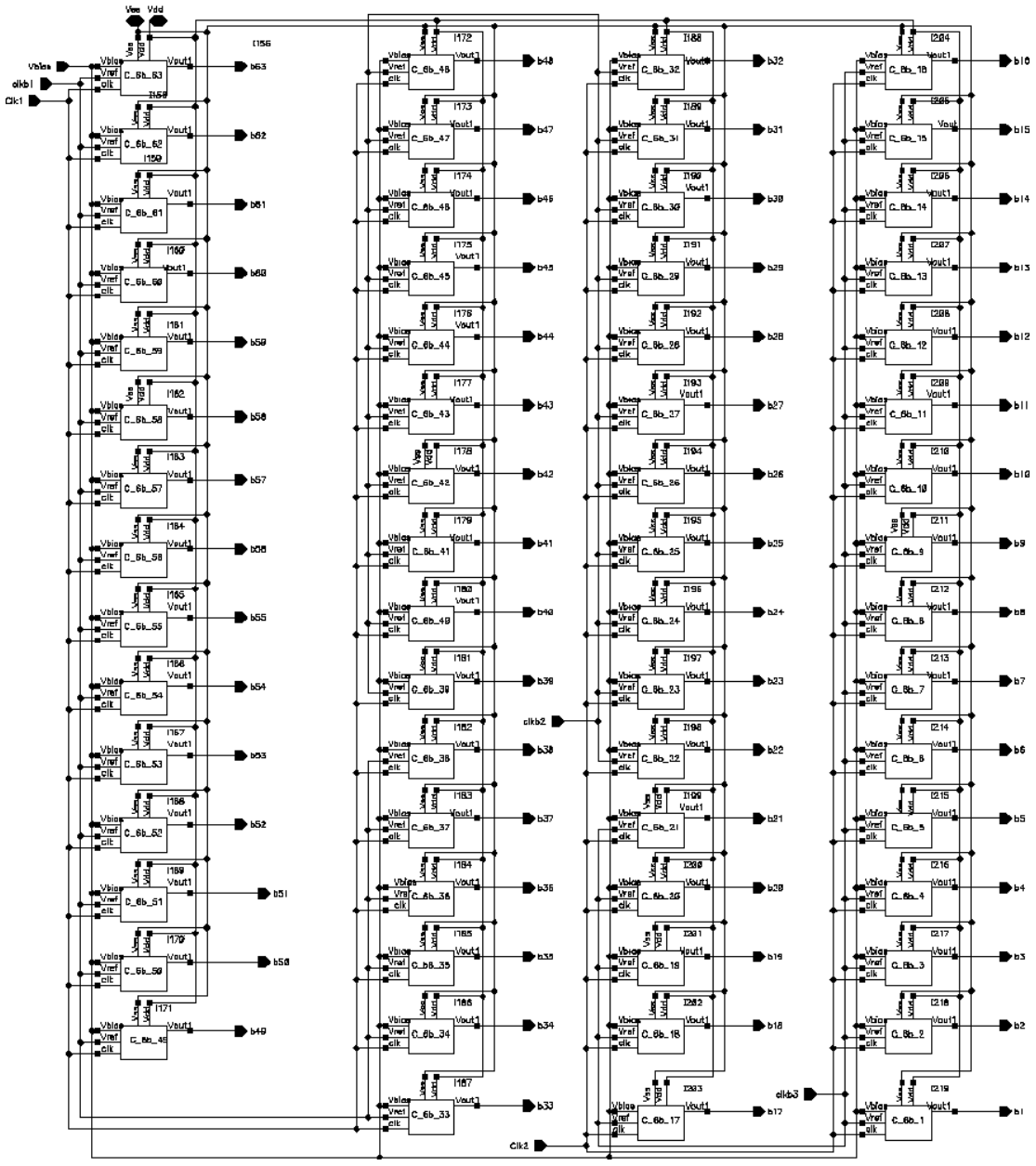


Figure 7-36　63 CDC Comparators

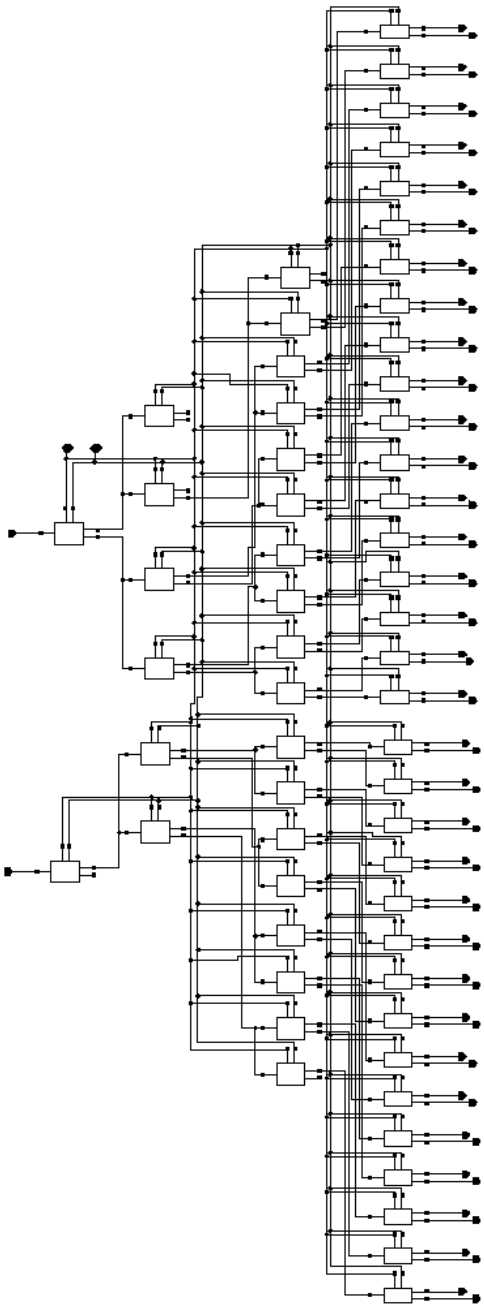


Figure 7-37 Schematic of Clock Tree for 6-b ADC

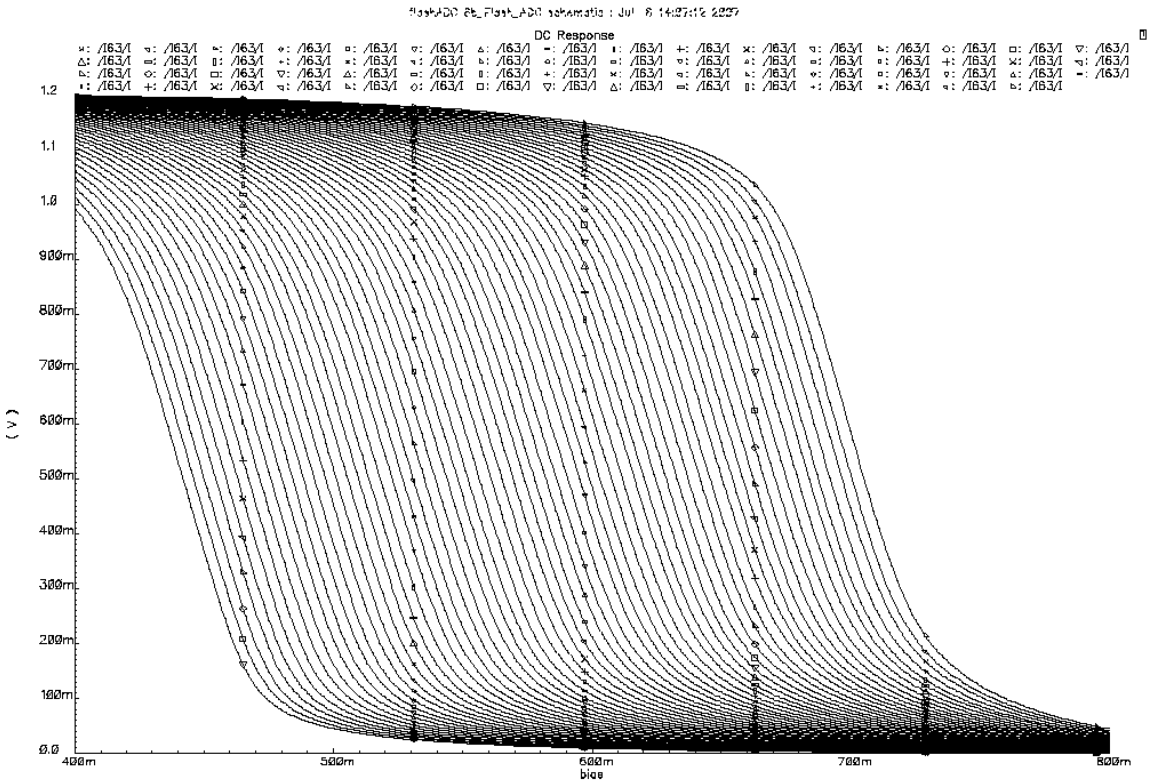


Figure 7-38 Outputs of 63 Quantization Inverters at DC

Figure 7-38 presents the voltage transfer characteristics of 63 quantization inverters in the 6-b ADC. The threshold voltages are regularly spaced.

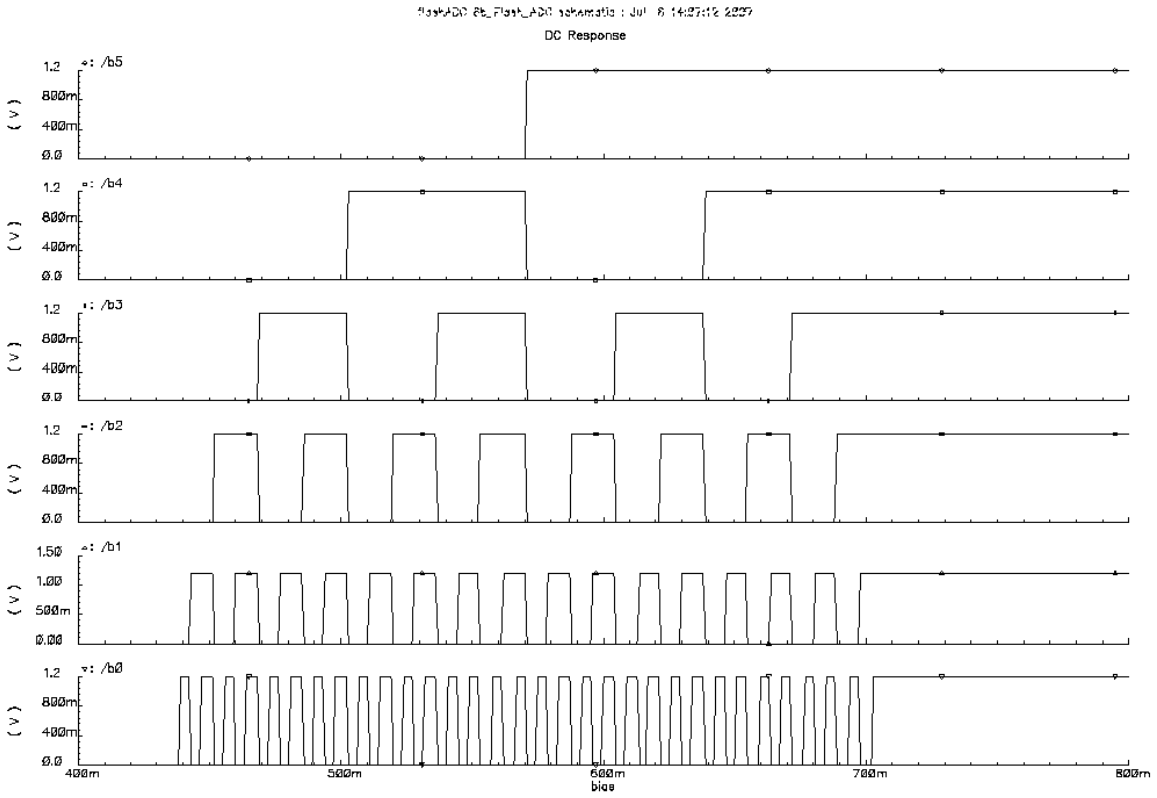


Figure 7-39 Outputs of 6 bits at DC

Figure 7-39 presents 6-bit outputs of the 6-b ADC at DC simulation. It proves that the 6-b ADC functions correctly.

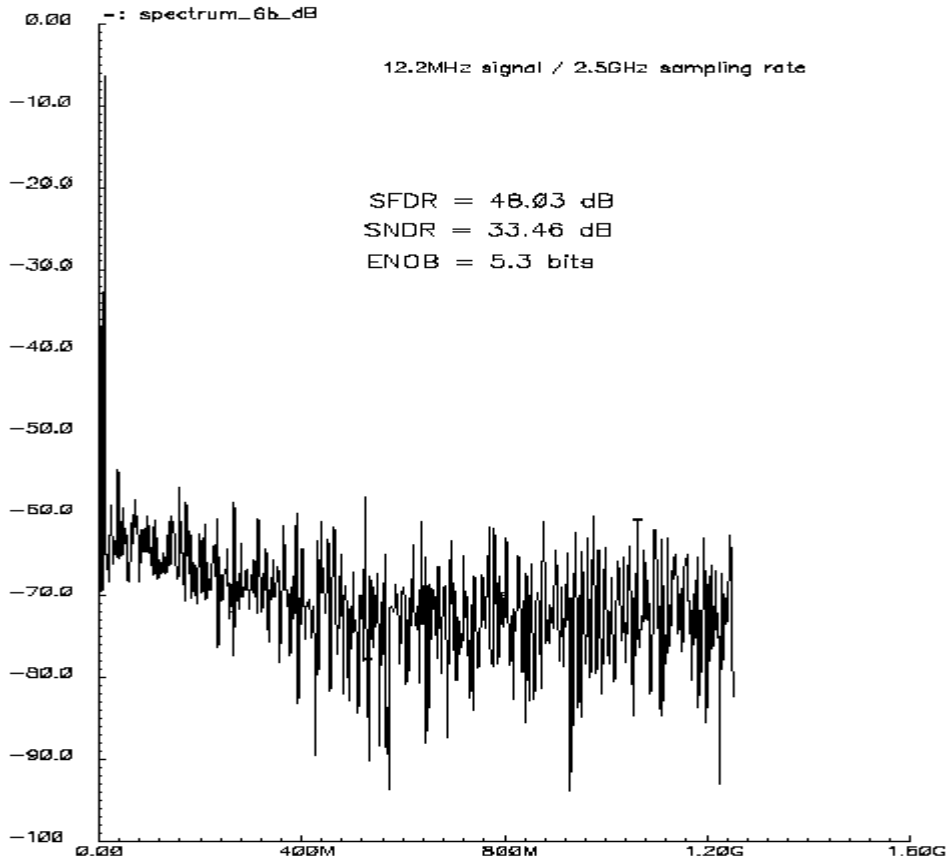


Figure 7-40 Spectrum of 12.2MHz Signal

Figure 7-40 presents a reconstructed spectrum of 12.2MHz input signal with 2.5-GSPS of the 6-b ADC. The ADC achieves 48.03-dB SFDR, 33.46-dB SNDR and 5.3-bits ENOB. The noise floor is at about -60-dB lower than the signal power.

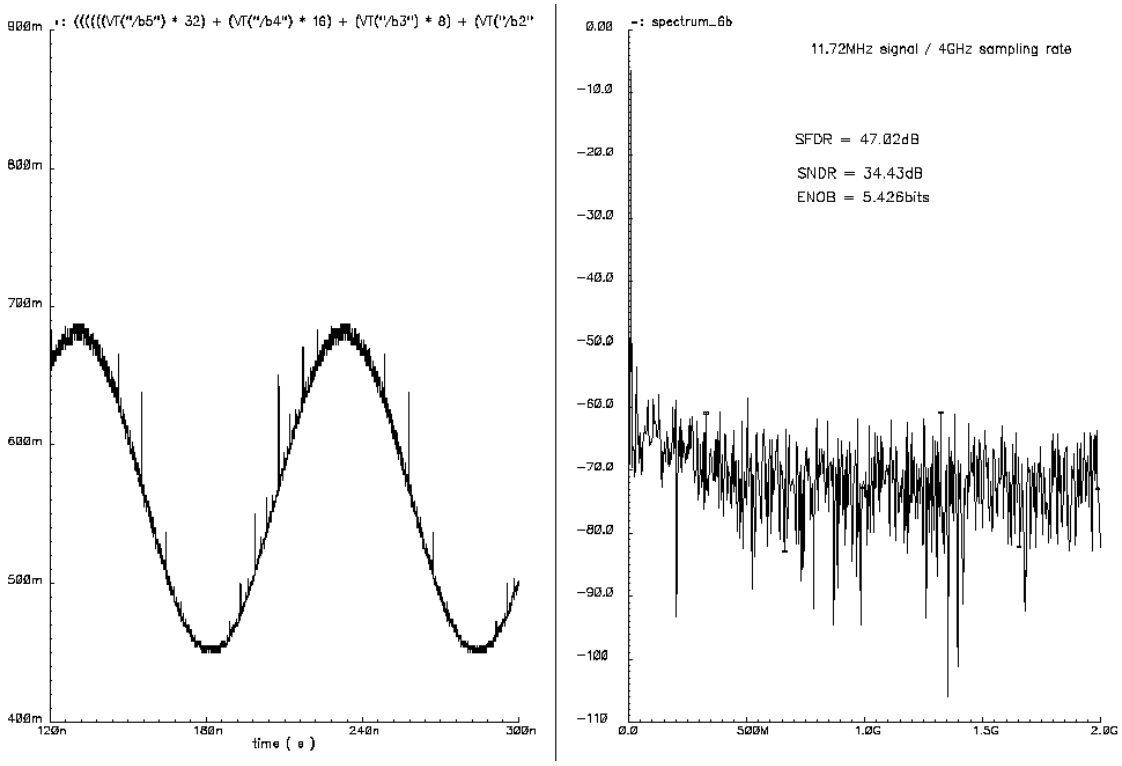


Figure 7-41 Reconstructed Signal and Spectrum of 11.7MHz signal

Figure 7-41 presents a reconstructed signal and spectrum of 11.7MHz input signal with 4-GSPS of the 6-b ADC. The ADC achieves 47.02-dB SFDR, 34.43-dB SNDR and 5.4-bits ENOB. The noise floor is at about -60-dB lower than the signal power.

7.3.2 Comparison with State-of-the-art 6-b Flash ADC

There is one 6-b flash ADC designed in 0.13μm digital CMOS process at 1.2-GS/s reported in IEEE Journal of Solid-State Circuit in 2005 by Sandner and his colleagues from the Microelectronics Development Center of Siemens AG, now Infineon Technologies, in Villach, Austria [23]. The flash ADC employs differential comparators and capacitive interpolation architecture for low power and wide bandwidth. This 6-b flash ADC is used as a reference to evaluate the proposed 6-b CDC-based pipelined ADC as depicted in Table 7-6.

Table 7-6 Performance Comparison of 6-b Flash ADC in 130nm Digital CMOS Process

	Sandner 6-b Flash ADC [23] in 2005		Proposed 6-b CDC-based Pipelined Flash ADC	
Sampling Rate	@600MSps	@1.2GSps	@2.5GSPS	@4GSPS
DNL	< 0.4LSB	---	---	---
INL	< 0.6LSB	---	---	---
ENOB	5.6bits @DC	5.7bits @DC	5.3bits @12.2MHz	5.4bits @11.7MHz
ERBW	600MHz	700MHz	1.1 GHz	
POWER	90mW	160mW	20mW	28mW
FoM	1.5pJ/conv	2.2pJ/conv	0.23pJ/conv	0.17pJ/conv
Supply Voltage	1.5v		1.2v	
Reference Voltage	1.0v		0.24mv	
Chip Area	0.12mm^2		0.088mm^2 (estimated)	
Technology	Standard 0.13μm CMOS		IBM 130nm Digital CMOS	

The figure of merit (FoM) in the table of the performance comparison is calculated as

$$FoM = \frac{Power}{2^{ENOB,DC} \cdot 2 \cdot ERBW}[pJ / convstep] \tag{7.1}$$

The proposed CDC-based pipelined flash ADC achieves a FoM of 0.23pJ/conv. at 2.5-GSPS and 0.17pJ/conv. at 4-GSPS, with smaller area size and similar ENOB performance at 1.2v supply voltage. The overall performance of our 6-b flash ADC is better than the ADC in [23]. The achieved performance, large ERBW, high sampling rate, low power and area consumption, of the 6-b ADC implementation once again prove that the proposed CDC-based pipelined CMOS flash ADC architecture has superior features for wideband communication systems-on-a-chip.

7.4 Comparison of BDB and TGB Encoders

In this section, comparison of dynamic performance of a 15mv-input-range 4-b CDC-based pipelined CMOS flash ADC using the BDB encoder and using the TGB encoder is depicted in Figure 7-42 with 2.5 GSPS and in Figure 7-43 with 4 GSPS. Sampled at 4 GSPS, the SNDR of ADC using the TGB encoder is constantly 4 to 5 dB above the SNDR of ADC using the BDB encoder, which accounts for an increased ENOB by about 0.7 bit.

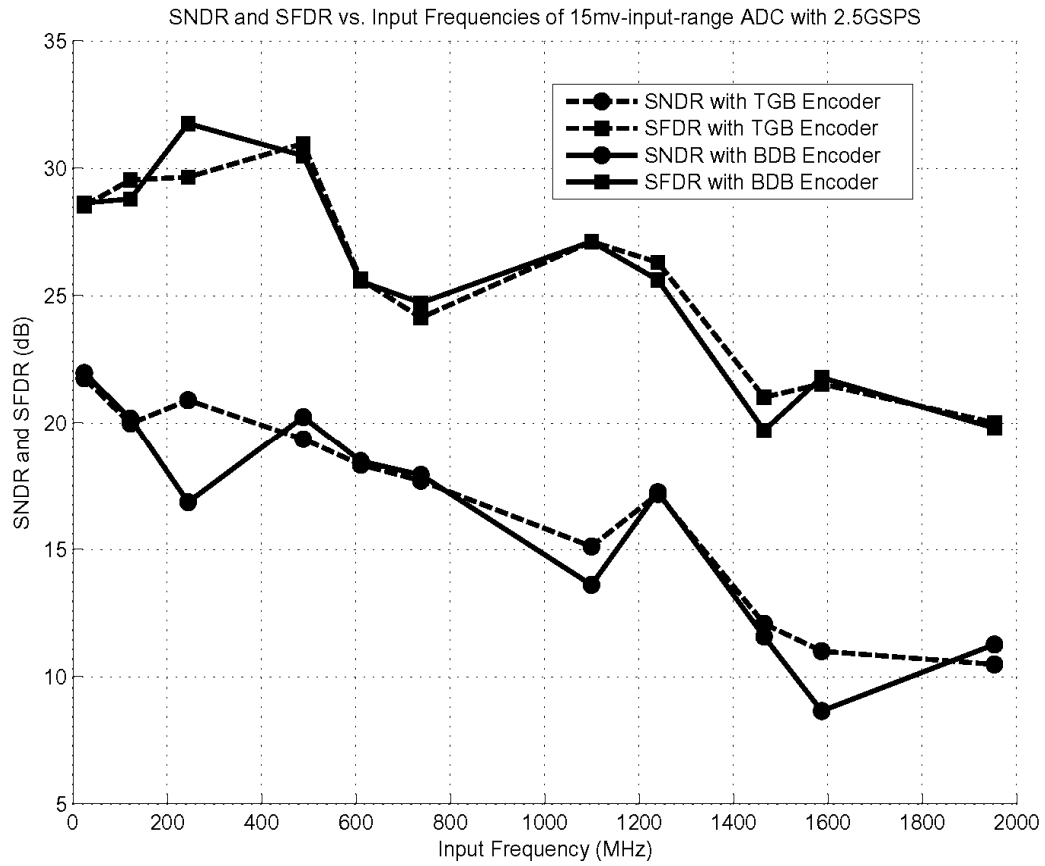


Figure 7-42 Performance Comparison of the ADC with 2.5-GSPS

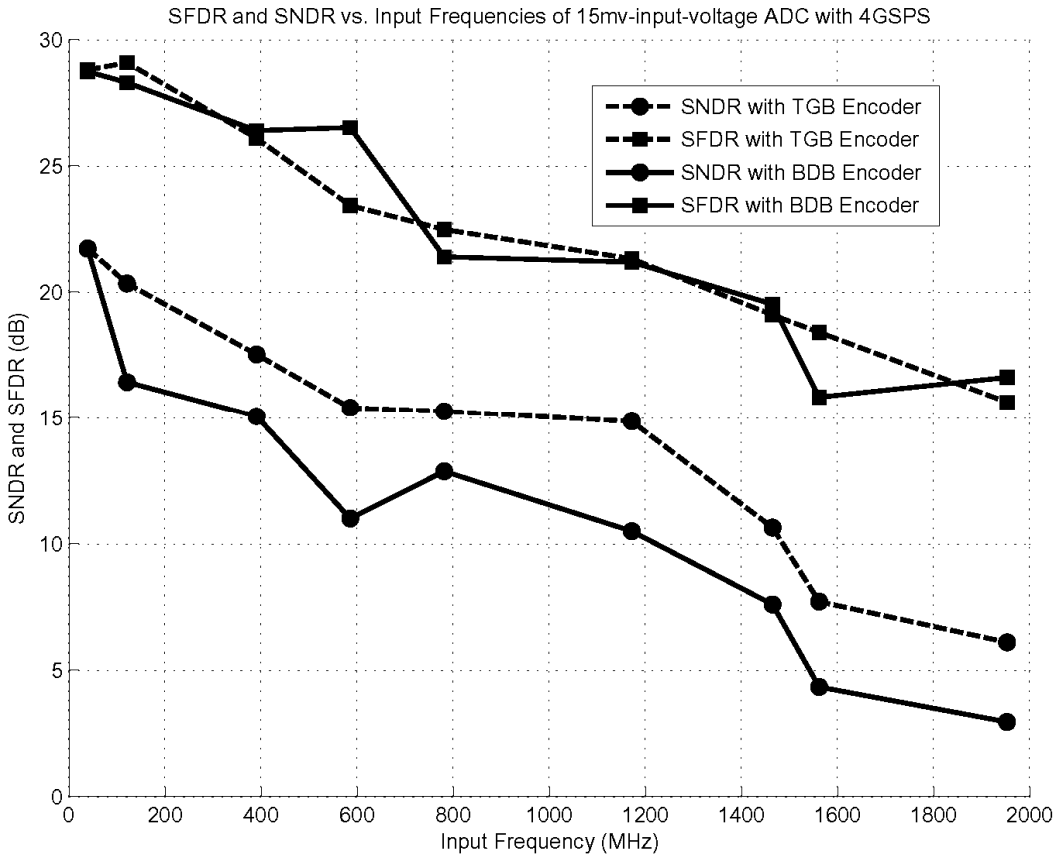


Figure 7-43 Performance Comparison of the ADC with 4-GSPS

Chapter 8 APPLICATION OF 4-B CDC-BASED PIPELINED ADC IN A DIGITAL WIDEBAND RECEIVER-ON-A-CHIP

This chapter presents a mixed-signal block-level simulation and verification flow in Cadence AMS integration platform for an 1 GHz bandwidth digital receiver which include the proposed 4-b 2.5 GSPS CDC-based pipelined CMOS flash ADC. This flow is different from Simulink-based system-level simulation, by considering the effects originating from analog and mixed signal subsystem. It involves low-level models of the devices in critical analog blocks. After the individual blocks have been designed, an AMS verification flow was built to integrate all the blocks in different level descriptions into one simulation platform for design verification. With Verilog-AMS, one can create and use modules that describe the high-level behavior and structure of analog, digital, and mixed-signal components and systems.

8.1 AMS Verification Flow

The AMS verification flow of the digital receiver is presented in Figure 8-1. It is composed of signal generation, data collection, and digital signal processing. An analog input signal and a clock signal need to be generated as the inputs to the receiver. In this verification, mono-tone or multi-tone sinusoidal input signals and a 2.5 GHz clock signal are created in Verilog-AMS code with real-time function. The data collection includes a 2.5

-GSPS ADC schematic design and a DEMUX in VHDL. The digital signal processor (DSP) includes a 256-point, 12-point kernel function FFT in VHDL, and a frequency detector in VHDL. The verification process stimulates the Spectre simulator, Verilog_AMS compiler, and VHDL compiler for analog and digital mixed signals in the flow. Both the analog signal and clock signal generator in Verilog_AMS are in Appendix B.

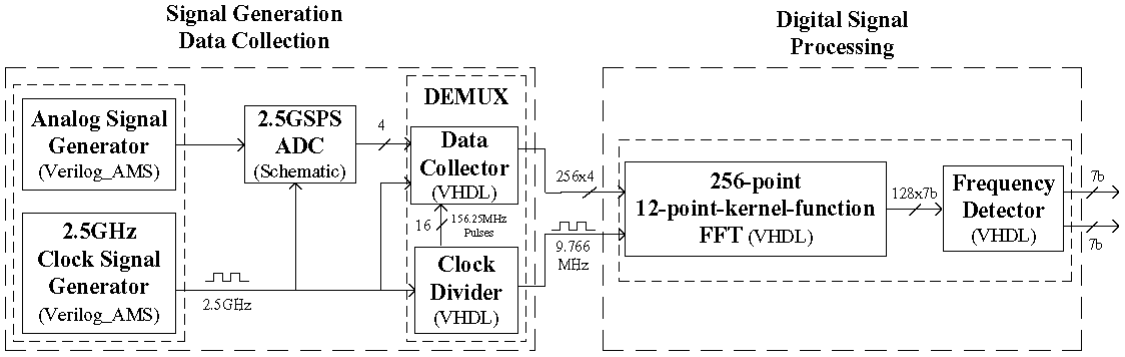


Figure 8-1 AMS Verification Flow of Digital Receiver

8.2 DEMUX Design

Figure 8-2 presents the diagram of DEMUX design. The DEMUX collects 256 sets of 4-b output data of ADC clocked at 2.5 GHz and feed them to the 256-point FFT clocked at 9.76 MHz. Four 16-b shift registers clocked at 2.5 GHz collect the data from the outputs of 4-b ADC, and transfer the data into sixteen 16x4 pipelined registers at every 6.4 ns. The sixteen 16x4 pipelined registers transfer the data to a 256x4 pipelined registers at every 102.4 ns. A total of 256x4 bits of data is then sent to the 256-point FFT.

.

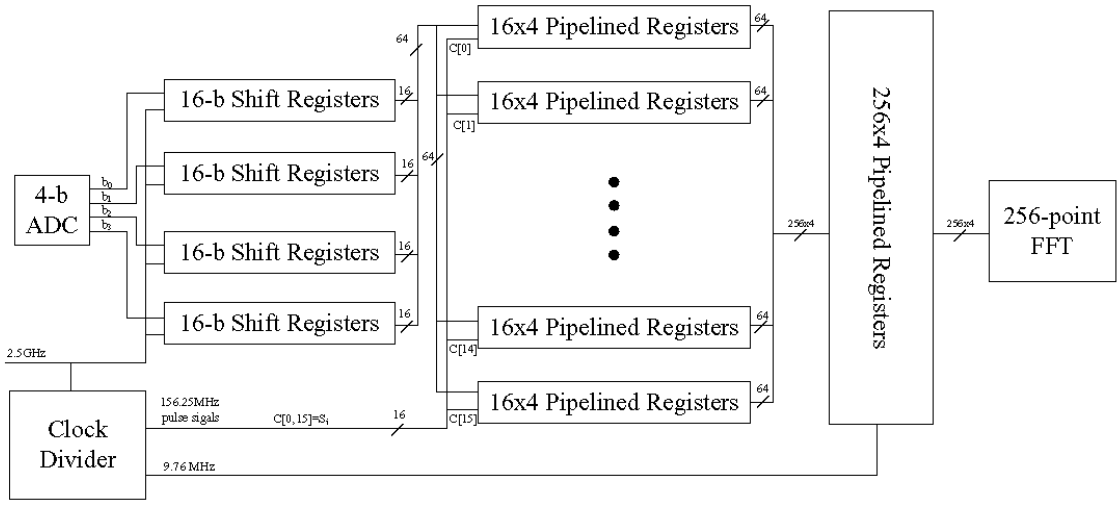


Figure 8-2 Diagram of DEMUX

The clock divider in the DEMUX functions as a timing controller. Four 16-b shift registers are clocked at 2.5GHz, and 256x4 pipelined registers are clocked at 9.76MHz. Sixteen 16x4 pipelined registers are clocked by sixteen 6.4 ns pulse signals C[0:15] as shown in Figure 8-3. The 16 states S_i of C[0:15] are presented in the state diagram shown in Figure 8-4. The DEMUX is programmed in VHDL. The code is attached in Appendix C.

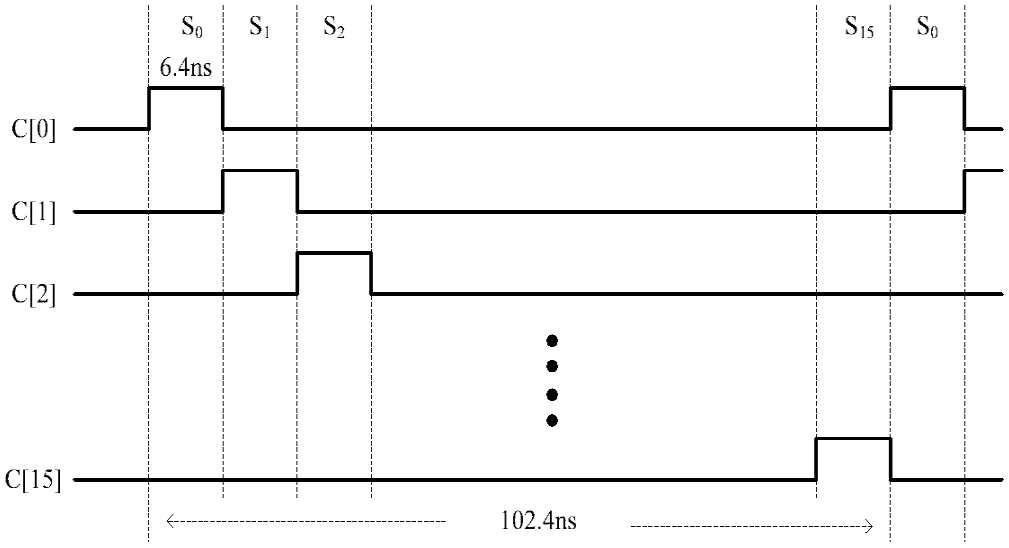


Figure 8-3 Sixteen 6.4ns Pulse Signals

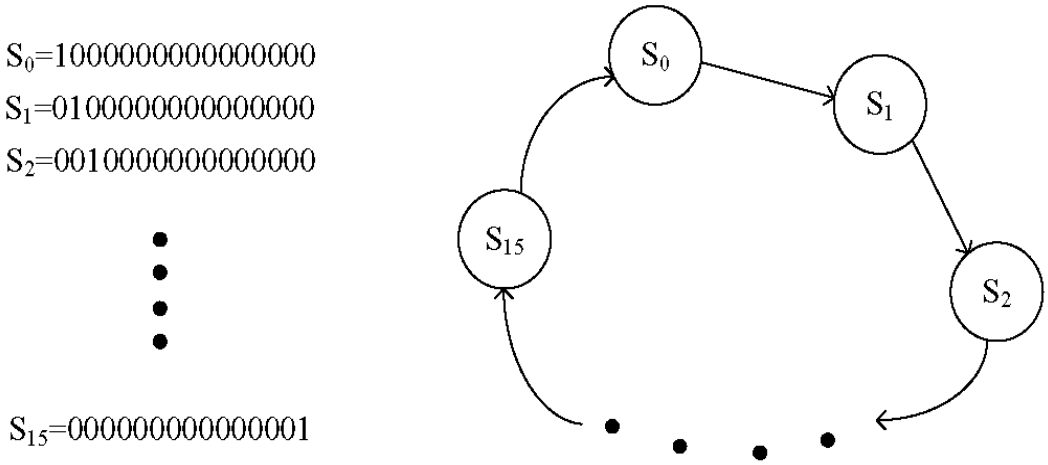


Figure 8-4 State Diagram of Pulse Signals

8.3 Verification Results

Figure 8-5 presents verification results of an input signal frequency of 1125 MHz. The signal is detected at the frequency bin of 115 of the FFT outputs. The frequency bin of 1125MHz is calculated as

$$\frac{1}{2.5GHz/256} \times 1125MHz = 115_d = 73_h .$$

The address y[6:0] of the signal with maximum amplitude is 73_h, and the output x_out[115] has an amplitude of $2F_h$ which is the highest peak amplitude as shown in Figure 8-5.

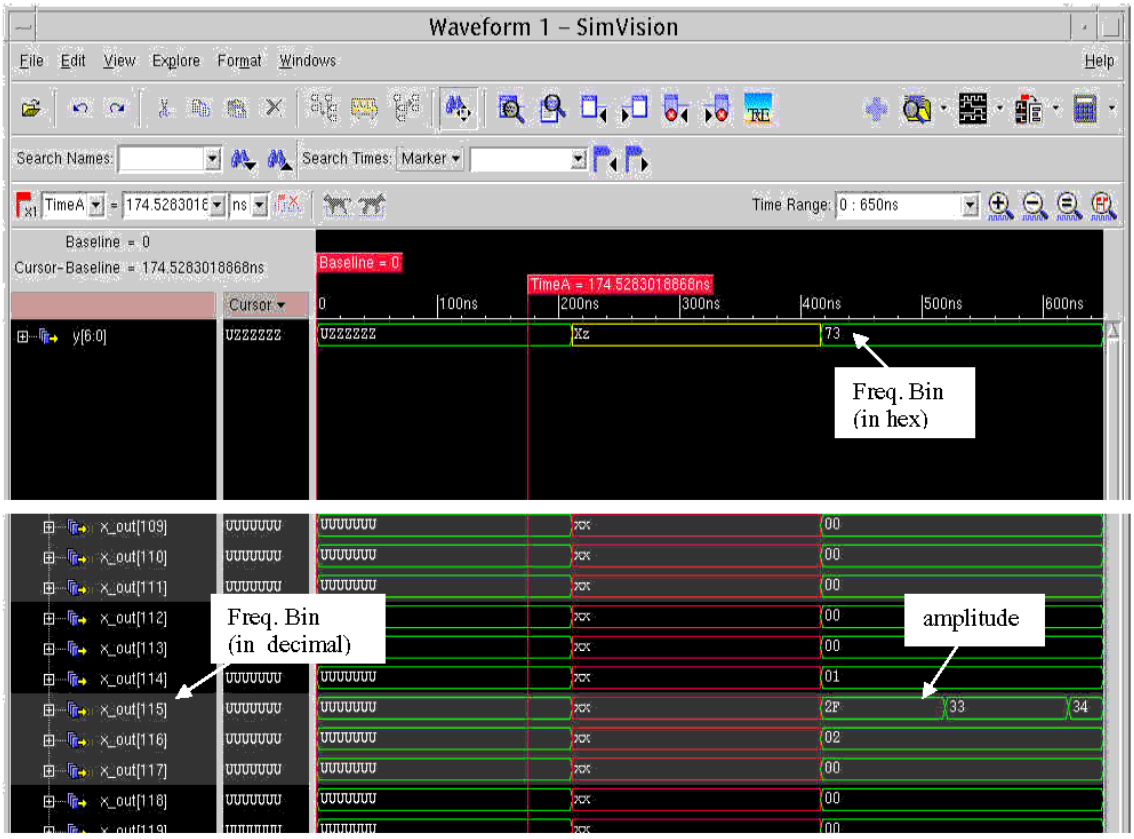


Figure 8-5 Bin and Peak Amplitude of 1125MHz Signal

Figure 8-6 presents verification results of two simultaneous signals of 700 MHz and 1 GHz. The two frequencies are detected by the receiver. The 700 MHz signal has the highest peak amplitude of 09_h at the frequency bin of 72_d and 1GHz signal has 2nd highest peak amplitude of 05_h at the frequency bin of 102_d.

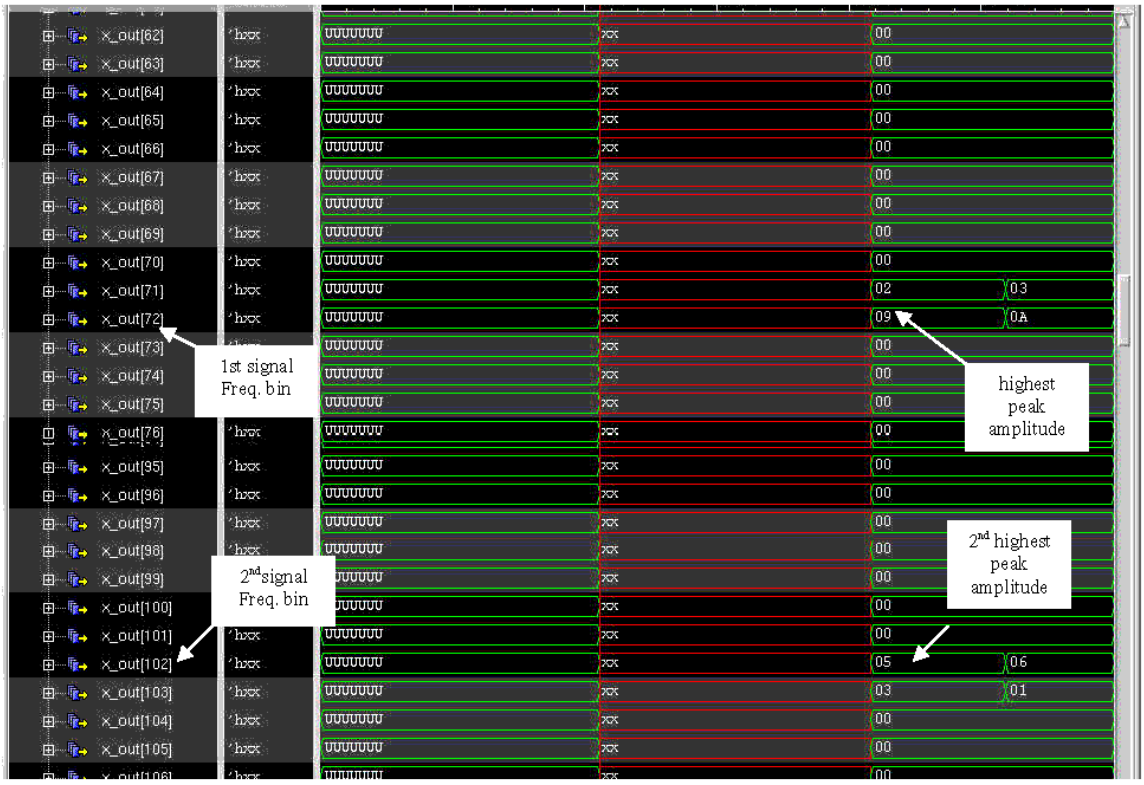


Figure 8-6 1st and 2nd Peak Amplitude of 700MHz and 1GHz

Chapter 9 CONCLUSIONS

Generally, high-speed flash ADCs use the differential amplifier structure in pre-amplifier and comparator design. There are two types of offsets in differential comparators. They are: 1) static and random offsets due to device mismatches, which in turn would be amplified to larger offsets in output, and 2) dynamic offsets at output of comparators, which occur due to clock switching in the regenerative latch.

In this dissertation a high-speed clocked digital comparator (CDC) with inverter configuration for high-speed high-performance CMOS flash ADCs was first proposed. Both the dynamic offset and the random static offset were reduced by a proposed dynamic offset suppression technique and a circuit optimization method. Employing a multi-stage pipelined architecture, a CDC-based pipelined CMOS flash ADC architecture was presented for high-speed conversion in low-voltage CMOS processes. Three CDC-based pipelined CMOS flash ADCs were implemented in 130 nm CMOS process. The experimental results reported the ADCs achieve high sampling rate, high SFDR, low power, a wide bandwidth up to the Nyquist frequency. These features make "all-digital" wideband communication SoC become practical.

The CDC comparator is a single-ended circuit design. Experiments of temperature variation on SFDR analysis of 4-b CDC-based ADCs were conducted. FFT Spectra of 400 MHz input signal with -40°C, 25°C, 75°C, 140°C were shown in Figures 9-1, 9-2, 9-3 and 9-4, respectively. Table 9-1 tables the SFDR results. It is shown that the SFDR is increased with the increase of temperature. Future work of CDC-based pipelined CMOS flash ADCs includes that, first, since the CDC comparator is the single-ended circuit design, the impact of the process, voltage and temperature variation on the performance of CDC-based ADCs will be investigated in detail. Secondly, the CDC comparator design automation is desired for two reasons; one for easily migrating the design into the different CMOS processes, and two for enhancing the optimization for the best dynamic performance.

Table 9-1 Temperature variation on SFDR analysis of 4-b 2.5-GSPS CDC-based ADCs with input signal frequency of 400 MHz

Temperature	-40°C	25°C	75°C	140°C
SFDR(dB)	24.22	29.8	32.32	31.32

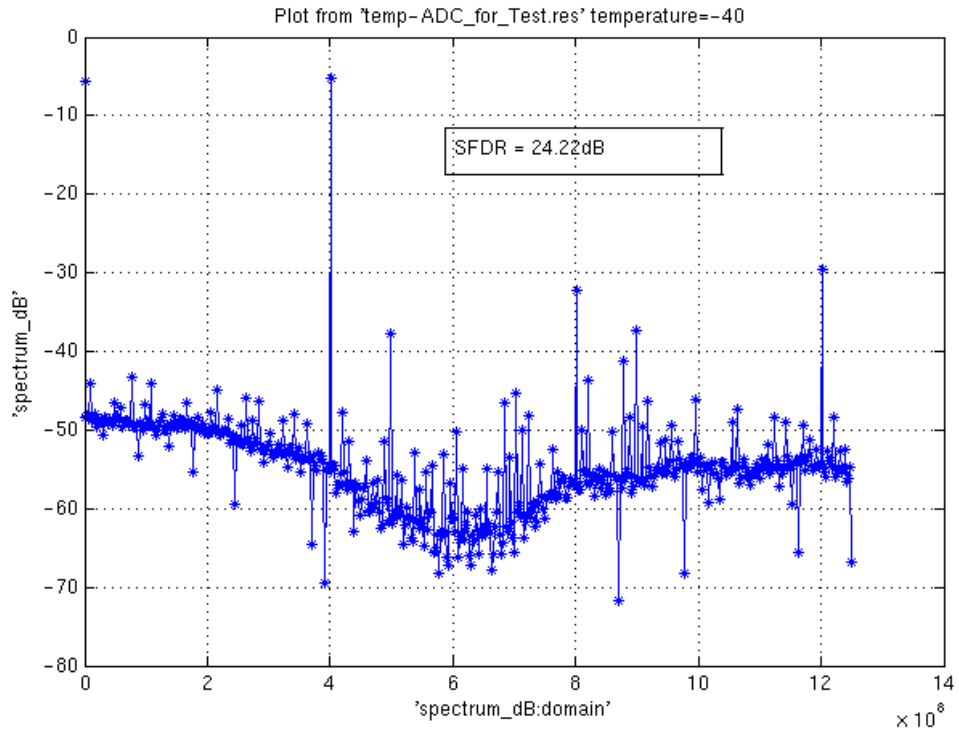


Figure 9-1 Spectrum of 400MHz Signal with -40°C

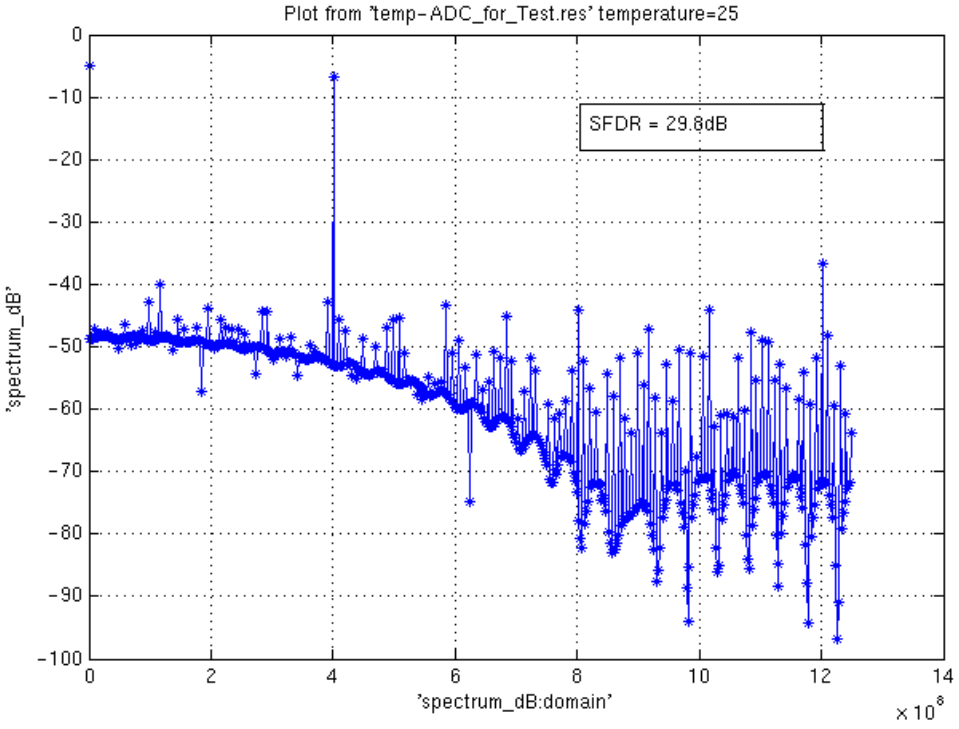


Figure 9-2 Spectrum of 400MHz Signal with 25°C

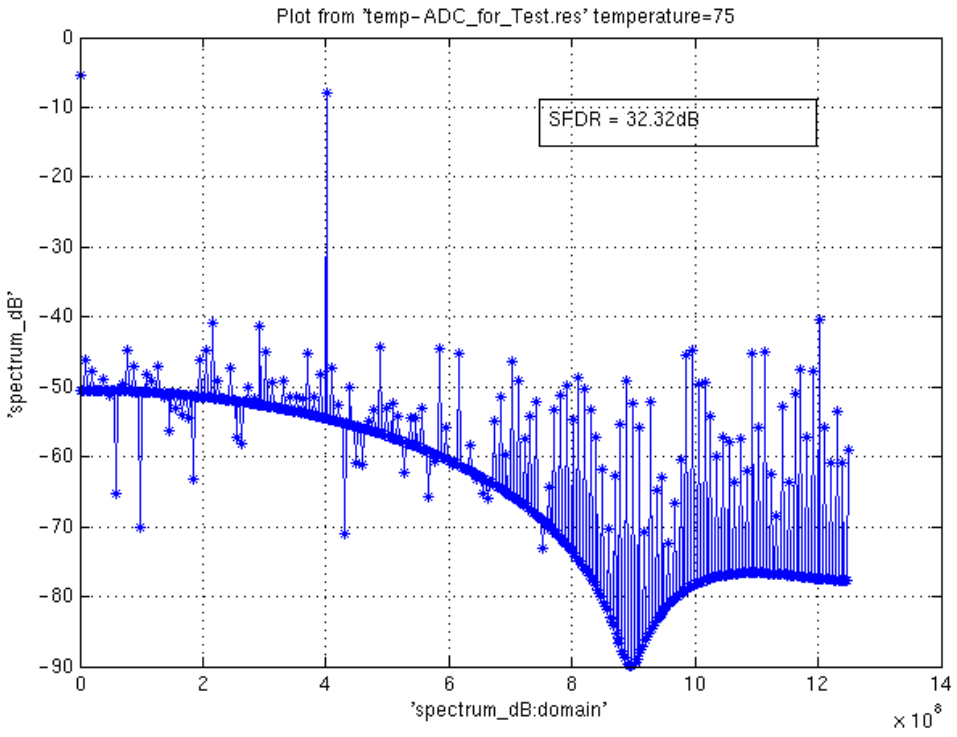


Figure 9-3 Spectrum of 400MHz Signal with 75°C

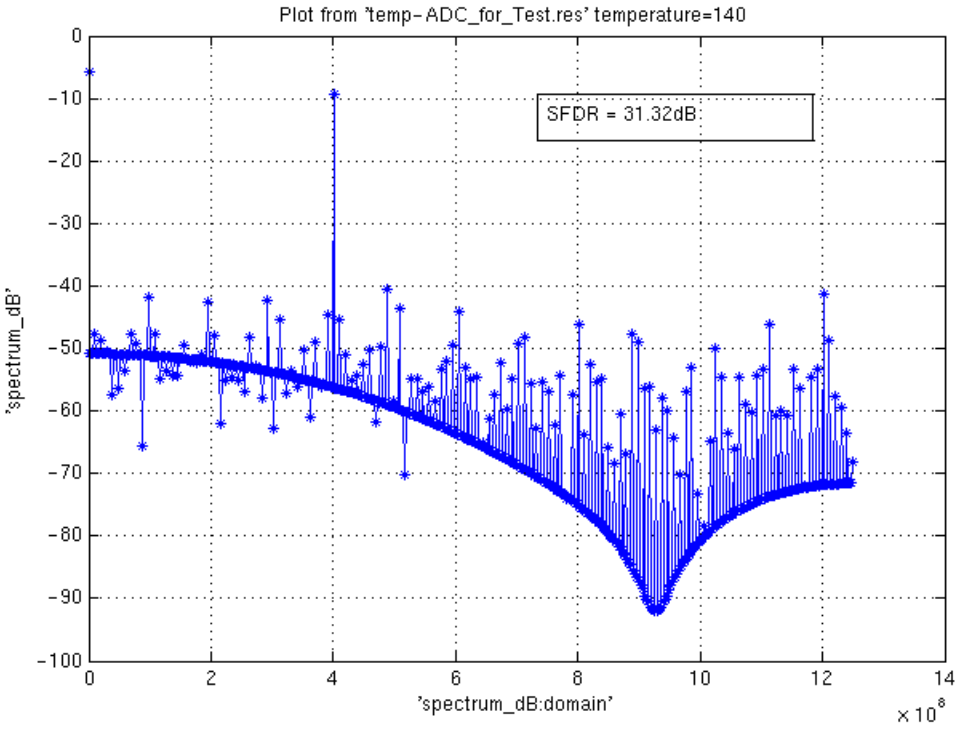


Figure 9-4 Spectrum of 400MHz Signal with 140°C

APPENDIX

A. Estimation of Source and Drain Areas and Peripheries

A simple rectangular MOSFET layout in an inverter is shown in Figure A-1 [10].

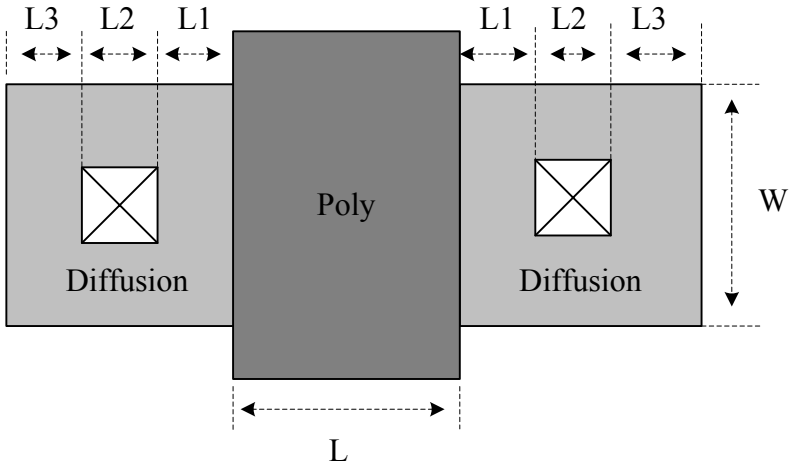


Figure A-1 Estimation of Source and Drain Areas and Peripheries

The minimum possible source or drain area would be that indicated by the sum of the lengths, L1, L2, and L3 times W. The lengths L1, L2 and L3 are related to the design rules for a given process and are as follows:

L1 = Minimum allowable distance between the contact in S/D and the poly

L2 = Width of a minimum size contact to diffusion

L3 = Minimum allowable distance from the contact in S/D to the edge of the S/D

The minimum area of the drain and source is (L1+L2+L3) × W and the corresponding periphery is 2(L1+L2+L3) + 2W.

B. Signal / Clock Generators in Verilog_AMS

1. Two-signal Generator

```
//Verilog-AMS HDL for "usrlib", "analogGen" "verilogams"
`include "constants.vams"
`include "disciplines.vams"
`timescale 1ns/100ps

module analogGen (analsig, Vdd, Vss );

    output analsig, Vdd, Vss;
    electrical analsig, Vdd, Vss;
    parameter real vdd = 1.2, vss = 0;
    parameter real frequency1 = 1e9, ampl1 = 0.1;
    parameter real frequency2 = 9e8, ampl2 = 0.14;
    real tempvar;

analog begin
    V(Vdd) <+ transition( vdd );
    V(Vss) <+ transition( vss );
    V(analsig) <+ (ampl1 * sin(2.0*`M_PI * frequency1 * $abstime)) + (ampl2 *
sin(2.0*`M_PI * frequency2 * $abstime))    + 0.6;
    $bound_step(0.02 / frequency1);
end

endmodule
```

2. Clock Generator

```
//Verilog-AMS HDL for "adc_ideal", "clk_digital" "verilogams"

`include "constants.vams"
`include "disciplines.vams"
`timescale 1ns/100ps
```

```
module clk_digital ( clk );

    output clk;
    logic clk;
    reg clock;

    parameter period = 0.4;

    assign clk = clock;
    initial
    begin
    clock = 1'b0;
    forever #(period/2) clock = ~clock;
    end

endmodule
```

3. Converting Electrical Signal to Logic Signal

```
//Verilog-AMS HDL for "usrlib", "electrical_2_logic" "verilogams"

`include "constants.vams"
`include "disciplines.vams"

module electrical_2_logic (bit3,bit2,bit1,bit0,clk_in, out3,out2,out1,out0,clk_out);

    input bit3, bit2, bit1, bit0, clk_in;
    output out3, out2, out1, out0;
    output clk_out;
    logic out3, out2, out1, out0;
    logic clk_out;

    electrical bit3, bit2, bit1, bit0, clk_in;

    integer temp3,temp2,temp1,temp0;
    integer tmp_clk;

    assign out3 = temp3;
    assign out2 = temp2;
    assign out1 = temp1;
    assign out0 = temp0;
    assign clk_out = tmp_clk;
```

```
	always @ ( above( V(bit3) - 0.6))
		temp3 = 1'b1;
	always @ ( above( 0.6 - V(bit3)))
		temp3 = 1'b0;

	always @ ( above( V(bit2) - 0.6))
		temp2 = 1'b1;
	always @ ( above( 0.6 - V(bit2)))
		temp2 = 1'b0;

	always @ ( above( V(bit1) - 0.6))
		temp1 = 1'b1;
	always @ ( above( 0.6 - V(bit1)))
		temp1 = 1'b0;

	always @ ( above( V(bit0) - 0.6))
		temp0 = 1'b1;
	always @ ( above( 0.6 - V(bit0)))
		temp0 = 1'b0;

	always @ ( above( V(clk_in) - 0.6))
		tmp_clk = 1'b1;
	always @ ( above( 0.6 - V(clk_in)))
		tmp_clk = 1'b0;
endmodule
```

C. DEMUX in VHDL

```
library ieee,work;
	use ieee.std_logic_1164.all;
	use IEEE.STD_LOGIC_TEXTIO.ALL;
	use STD.TEXTIO.ALL;
	use WORK.CONV.ALL;

package demux_type is

	type state_type is (s0,	s1,	s2,	s3,	s4,	s5,	s6,	s7,
				s8,	s9,	s10, s11, s12, s13, s14, s15, s_initial);
	type TYPE_DEMUX_ARRAY_16x4 is array(0 to 15) of std_logic_vector(3 downto
0);
	type TYPE_DEMUX_ARRAY_256x4 is array(0 to 255) of std_logic_vector(3
downto 0);
	type TYPE_DEMUX_ARRAY_16x64 is array(0 to 15) of
TYPE_DEMUX_ARRAY_16x4;
```

```
end demux_type;

library ieee,work;
	use ieee.std_logic_1164.all;
	use IEEE.STD_LOGIC_TEXTIO.ALL;
	use STD.TEXTIO.ALL;
	use work.demux_type.all;
	use WORK.CONV.ALL;

entity clock_generator is
	Port (
			clk : IN std_logic;
			reset : IN std_logic;
			output : OUT std_logic_vector(15 downto 0);
		clk_fft : OUT std_logic;
			out_clk : OUT std_logic;
			out_clk1 : OUT std_logic;
			out_clk2 : OUT std_logic;
			out_clk3 : OUT std_logic;
			out_clk4 : OUT std_logic;
			out_clk5 : OUT std_logic;
			temp_clk : OUT std_logic
		);
end clock_generator;

architecture behavior of clock_generator is

----- signal declaration -----
signal current_state, next_state : state_type;
signal clk_16 : std_logic;

----- start -----

begin

   clk16: process(reset, clk)

     variable count : integer := 0;

   begin
	if (clk'event and clk='1') then
	    if (reset = '1') then

		current_state <= s_initial;
```

```
        else
        count := count + 1;
        if (count = 16 and reset /= '1') then
        count := 0;
        current_state <= next_state;
        end if;

       end if;
          end if;
    end process clk16;

   -- sync: process( clk_16 )
   -- begin
   --       if (clk_16'event and clk_16='1') then
   --               current_state <= next_state;
   --            end if;
   --       end if;
   -- end process sync;

    FSM: process(current_state)      -- mod => input is inserted
    begin
--          next_state <= s0;      -- default assignment;

          case current_state is
                when s_initial =>
                  next_state <= s0;
                  output <= "0000000000000000";
                when s0 =>
                        next_state <= s1;
                        output <= "0000000000000001";
                when s1 =>
                        next_state <= s2;
                        output <= "0000000000000010";
                when s2 =>
                        next_state <= s3;
                        output <= "0000000000000100";
                when s3 =>
                        next_state <= s4;
                        output <= "0000000000001000";
                when s4 =>
                        next_state <= s5;
                        output <= "0000000000010000";
                when s5 =>
                        next_state <= s6;
                        output <= "0000000000100000";
```

```
            when s6 =>
                next_state <= s7;
                output <= "0000000001000000";
            when s7 =>
                next_state <= s8;
                output <= "0000000010000000";
            when s8 =>
                next_state <= s9;
                output <= "0000000100000000";
            when s9 =>
                next_state <= s10;
                output <= "0000001000000000";
            when s10 =>
                next_state <= s11;
                output <= "0000010000000000";
            when s11 =>
                next_state <= s12;
                output <= "0000100000000000";
            when s12 =>
                next_state <= s13;
                output <= "0001000000000000";
            when s13 =>
                next_state <= s14;
                output <= "0010000000000000";
            when s14 =>
                next_state <= s15;
                output <= "0100000000000000";
            when s15 =>
                next_state <= s0;
                output <= "1000000000000000";
        end case;
    end process FSM;

    gen_out_clk: process(current_state)
    begin

        if ( current_state = S0 or current_state = S1 or current_state = S2 or current_state =
S3 or current_state = S4 or current_state = S5 or current_state = S6 or current_state = S7)
then
            clk_fft <= '1';
        else
            clk_fft <= '0';
        end if;

        --- This clk will synchronize all the pipelines flip-flops ---
```

```
    if (current_state = S0 or current_state = S1) then
        out_clk <= '1';            -- high from 0 to 12.8ns
    else
        out_clk <= '0';
    end if;

    --- This clk will go to initial_sort subsystem ---
    --- and latch the sel_high_low into their flip-flops ---
    if (current_state = S3 or current_state = S4) then
        out_clk1 <= '1';          -- high from 19.2 to 32ns
    else
        out_clk1 <= '0';
    end if;

    if (current_state = S7 or current_state = S8) then
        out_clk2 <= '1';          -- high from 44.8 to 57.6ns
    else
        out_clk2 <= '0';
    end if;

    if (current_state = S8 or current_state = S9) then
        out_clk3 <= '1';        -- high from 51.2 to 64ns
    else
        out_clk3 <= '0';
    end if;

    if (current_state = S12 or current_state = S13) then
        out_clk4 <= '1';        -- high from 76.8 to 89.6ns
    else
        out_clk4 <= '0';
    end if;

    if (current_state = S14 or current_state = S15) then
        out_clk5 <= '1';        -- high from 89.6 to 102.4ns
    else
        out_clk5 <= '0';
    end if;


    if (current_state = S13 or current_state = S14) then
        temp_clk <= '1';      -- high from 83.2 to 96ns
    else
        temp_clk <= '0';
    end if;
end process gen_out_clk;
```

```
end behavior;
--------------------------------------------------------
--------------------------------------------------------
library ieee,work;
       use ieee.std_logic_1164.all;
       use IEEE.STD_LOGIC_TEXTIO.ALL;
       use STD.TEXTIO.ALL;
       use work.demux_type.all;
       use WORK.CONV.ALL;

entity latch_v1 is
      port ( clk : IN std_logic;
                  input : IN std_logic;
                  output : OUT std_logic
                );
end latch_v1;

architecture behavior of latch_v1 is

begin

      test : process(clk)

      begin
           if ( clk'event and clk='1') then
            output <= input;
           end if;
      end process test;
end behavior;
--------------------------------------------------------
--------------------------------------------------------
library ieee,work;
       use ieee.std_logic_1164.all;
       use IEEE.STD_LOGIC_TEXTIO.ALL;
       use STD.TEXTIO.ALL;
       use work.demux_type.all;
       use WORK.CONV.ALL;

entity latch_v1_4_p is
      port ( clk : IN std_logic;
                  input : IN std_logic_vector(3 downto 0);
                  output : OUT std_logic_vector(3 downto 0)
               );
end latch_v1_4_p;
```

```
architecture structure of latch_v1_4_p is

component latch_v1
      port ( clk : IN std_logic;
                  input : IN std_logic;
                  output : OUT std_logic
               );
end component;

begin

      latch0: latch_v1 port map (clk,input(0),output(0));
      latch1: latch_v1 port map (clk,input(1),output(1));
      latch2: latch_v1 port map (clk,input(2),output(2));
      latch3: latch_v1 port map (clk,input(3),output(3));
end structure;
----------------------------------------------------
----------------------------------------------------
library ieee,work;
       use ieee.std_logic_1164.all;
       use IEEE.STD_LOGIC_TEXTIO.ALL;
       use STD.TEXTIO.ALL;
       use work.demux_type.all;
       use WORK.CONV.ALL;

entity latch_v1_4_p_16 is
      port ( clk : IN std_logic;
                  input : IN std_logic_vector(3 downto 0);
                  output : OUT TYPE_DEMUX_ARRAY_16x4
               );
end latch_v1_4_p_16;

architecture structure of latch_v1_4_p_16 is

component latch_v1_4_p
      port ( clk : IN std_logic;
                  input : IN std_logic_vector(3 downto 0);
                  output : OUT std_logic_vector(3 downto 0)
               );

end component;

      signal temp_output : TYPE_DEMUX_ARRAY_16x4;
```

```
begin

      latch0: latch_v1_4_p port map (clk,input,temp_output(15));
      latch1: latch_v1_4_p port map (clk,temp_output(15),temp_output(14));
      latch2: latch_v1_4_p port map (clk,temp_output(14),temp_output(13));
      latch3: latch_v1_4_p port map (clk,temp_output(13),temp_output(12));
      latch4: latch_v1_4_p port map (clk,temp_output(12),temp_output(11));
      latch5: latch_v1_4_p port map (clk,temp_output(11),temp_output(10));
      latch6: latch_v1_4_p port map (clk,temp_output(10),temp_output(9));
      latch7: latch_v1_4_p port map (clk,temp_output(9),temp_output(8));
      latch8: latch_v1_4_p port map (clk,temp_output(8),temp_output(7));
      latch9: latch_v1_4_p port map (clk,temp_output(7),temp_output(6));
      latch10: latch_v1_4_p port map (clk,temp_output(6),temp_output(5));
      latch11: latch_v1_4_p port map (clk,temp_output(5),temp_output(4));
      latch12: latch_v1_4_p port map (clk,temp_output(4),temp_output(3));
      latch13: latch_v1_4_p port map (clk,temp_output(3),temp_output(2));
      latch14: latch_v1_4_p port map (clk,temp_output(2),temp_output(1));
      latch15: latch_v1_4_p port map (clk,temp_output(1),temp_output(0));
      output <= temp_output;
end structure;
----------------------------------------------------
----------------------------------------------------
library ieee,work;
       use ieee.std_logic_1164.all;
       use IEEE.STD_LOGIC_TEXTIO.ALL;
       use STD.TEXTIO.ALL;
       use work.demux_type.all;
       use WORK.CONV.ALL;

entity latch_v1_4_p_16_p is
      port ( clk : IN std_logic;
                  input : IN TYPE_DEMUX_ARRAY_16x4;
                  output : OUT TYPE_DEMUX_ARRAY_16x4
               );
end latch_v1_4_p_16_p;

architecture structure of latch_v1_4_p_16_p is

component latch_v1_4_p
      port ( clk : IN std_logic;
                  input : IN std_logic_vector(3 downto 0);
                  output : OUT std_logic_vector(3 downto 0)
               );
end component;
```

```
begin

      block_16_latch0: latch_v1_4_p port map (clk, input(15), output(15));
      block_16_latch1: latch_v1_4_p port map (clk, input(14), output(14));
      block_16_latch2: latch_v1_4_p port map (clk, input(13), output(13));
      block_16_latch3: latch_v1_4_p port map (clk, input(12), output(12));
      block_16_latch4: latch_v1_4_p port map (clk, input(11), output(11));
      block_16_latch5: latch_v1_4_p port map (clk, input(10), output(10));
      block_16_latch6: latch_v1_4_p port map (clk, input(9), output(9));
      block_16_latch7: latch_v1_4_p port map (clk, input(8), output(8));
      block_16_latch8: latch_v1_4_p port map (clk, input(7), output(7));
      block_16_latch9: latch_v1_4_p port map (clk, input(6), output(6));
      block_16_latch10: latch_v1_4_p port map (clk, input(5), output(5));
      block_16_latch11: latch_v1_4_p port map (clk, input(4), output(4));
      block_16_latch12: latch_v1_4_p port map (clk, input(3), output(3));
      block_16_latch13: latch_v1_4_p port map (clk, input(2), output(2));
      block_16_latch14: latch_v1_4_p port map (clk, input(1), output(1));
      block_16_latch15: latch_v1_4_p port map (clk, input(0), output(0));
end structure;
------------------------------------------------------
------------------------------------------------------
library ieee,work;
       use ieee.std_logic_1164.all;
       use IEEE.STD_LOGIC_TEXTIO.ALL;
       use STD.TEXTIO.ALL;
       use work.demux_type.all;
       use WORK.CONV.ALL;

entity latch_v1_4_p_16_p_16 is
      port ( clk : IN std_logic_vector(15 downto 0);
                        input : IN TYPE_DEMUX_ARRAY_16x4;
                  output : OUT TYPE_INPUT_FFT
                );
end latch_v1_4_p_16_p_16;

architecture structure of latch_v1_4_p_16_p_16 is

component latch_v1_4_p_16_p
      port ( clk : IN std_logic;
                  input : IN TYPE_DEMUX_ARRAY_16x4;
                  output : OUT TYPE_DEMUX_ARRAY_16x4
              );
end component;

signal tmp_output : TYPE_DEMUX_ARRAY_16x64;
```

```
begin

    block_64_latch0: latch_v1_4_p_16_p port map (clk(0),input,tmp_output(0 ));
    output(0) <= tmp_output(0)(0);
    output(1) <= tmp_output(0)(1);
    output(2) <= tmp_output(0)(2);
    output(3) <= tmp_output(0)(3);
    output(4) <= tmp_output(0)(4);
    output(5) <= tmp_output(0)(5);
    output(6) <= tmp_output(0)(6);
    output(7) <= tmp_output(0)(7);
    output(8) <= tmp_output(0)(8);
    output(9) <= tmp_output(0)(9);
    output(10) <= tmp_output(0)(10);
    output(11) <= tmp_output(0)(11);
    output(12) <= tmp_output(0)(12);
    output(13) <= tmp_output(0)(13);
    output(14) <= tmp_output(0)(14);
    output(15) <= tmp_output(0)(15);

    block_64_latch1: latch_v1_4_p_16_p port map (clk(1),input,tmp_output(1));
    output(16) <= tmp_output(1)(0);
    output(17) <= tmp_output(1)(1);
    output(18) <= tmp_output(1)(2);
    output(19) <= tmp_output(1)(3);
    output(20) <= tmp_output(1)(4);
    output(21) <= tmp_output(1)(5);
    output(22) <= tmp_output(1)(6);
    output(23) <= tmp_output(1)(7);
    output(24) <= tmp_output(1)(8);
    output(25) <= tmp_output(1)(9);
    output(26) <= tmp_output(1)(10);
    output(27) <= tmp_output(1)(11);
    output(28) <= tmp_output(1)(12);
    output(29) <= tmp_output(1)(13);
    output(30) <= tmp_output(1)(14);
    output(31) <= tmp_output(1)(15);

    block_64_latch2: latch_v1_4_p_16_p port map (clk(2),input,tmp_output(2));
    output(32) <= tmp_output(2)(0);
    output(33) <= tmp_output(2)(1);
    output(34) <= tmp_output(2)(2);
    output(35) <= tmp_output(2)(3);
    output(36) <= tmp_output(2)(4);
```

```
output(37) <= tmp_output(2)(5);
output(38) <= tmp_output(2)(6);
output(39) <= tmp_output(2)(7);
output(40) <= tmp_output(2)(8);
output(41) <= tmp_output(2)(9);
output(42) <= tmp_output(2)(10);
output(43) <= tmp_output(2)(11);
output(44) <= tmp_output(2)(12);
output(45) <= tmp_output(2)(13);
output(46) <= tmp_output(2)(14);
output(47) <= tmp_output(2)(15);

block_64_latch3: latch_v1_4_p_16_p port map (clk(3),input,tmp_output(3));
output(48) <= tmp_output(3)(0);
output(49) <= tmp_output(3)(1);
output(50) <= tmp_output(3)(2);
output(51) <= tmp_output(3)(3);
output(52) <= tmp_output(3)(4);
output(53) <= tmp_output(3)(5);
output(54) <= tmp_output(3)(6);
output(55) <= tmp_output(3)(7);
output(56) <= tmp_output(3)(8);
output(57) <= tmp_output(3)(9);
output(58) <= tmp_output(3)(10);
output(59) <= tmp_output(3)(11);
output(60) <= tmp_output(3)(12);
output(61) <= tmp_output(3)(13);
output(62) <= tmp_output(3)(14);
output(63) <= tmp_output(3)(15);

block_64_latch4: latch_v1_4_p_16_p port map (clk(4),input,tmp_output(4));
output(64) <= tmp_output(4)(0);
output(65) <= tmp_output(4)(1);
output(66) <= tmp_output(4)(2);
output(67) <= tmp_output(4)(3);
output(68) <= tmp_output(4)(4);
output(69) <= tmp_output(4)(5);
output(70) <= tmp_output(4)(6);
output(71) <= tmp_output(4)(7);
output(72) <= tmp_output(4)(8);
output(73) <= tmp_output(4)(9);
output(74) <= tmp_output(4)(10);
output(75) <= tmp_output(4)(11);
output(76) <= tmp_output(4)(12);
output(77) <= tmp_output(4)(13);
```

```
output(78) <= tmp_output(4)(14);
output(79) <= tmp_output(4)(15);

block_64_latch5: latch_v1_4_p_16_p port map (clk(5),input,tmp_output(5));
output(80) <= tmp_output(5)(0);
output(81) <= tmp_output(5)(1);
output(82) <= tmp_output(5)(2);
output(83) <= tmp_output(5)(3);
output(84) <= tmp_output(5)(4);
output(85) <= tmp_output(5)(5);
output(86) <= tmp_output(5)(6);
output(87) <= tmp_output(5)(7);
output(88) <= tmp_output(5)(8);
output(89) <= tmp_output(5)(9);
output(90) <= tmp_output(5)(10);
output(91) <= tmp_output(5)(11);
output(92) <= tmp_output(5)(12);
output(93) <= tmp_output(5)(13);
output(94) <= tmp_output(5)(14);
output(95) <= tmp_output(5)(15);

block_64_latch6: latch_v1_4_p_16_p port map (clk(6),input,tmp_output(6));
output(96) <= tmp_output(6)(0);
output(97) <= tmp_output(6)(1);
output(98) <= tmp_output(6)(2);
output(99) <= tmp_output(6)(3);
output(100) <= tmp_output(6)(4);
output(101) <= tmp_output(6)(5);
output(102) <= tmp_output(6)(6);
output(103) <= tmp_output(6)(7);
output(104) <= tmp_output(6)(8);
output(105) <= tmp_output(6)(9);
output(106) <= tmp_output(6)(10);
output(107) <= tmp_output(6)(11);
output(108) <= tmp_output(6)(12);
output(109) <= tmp_output(6)(13);
output(110) <= tmp_output(6)(14);
output(111) <= tmp_output(6)(15);

block_64_latch7: latch_v1_4_p_16_p port map (clk(7),input,tmp_output(7));
output(112) <= tmp_output(7)(0);
output(113) <= tmp_output(7)(1);
output(114) <= tmp_output(7)(2);
output(115) <= tmp_output(7)(3);
output(116) <= tmp_output(7)(4);
```

```
output(117) <= tmp_output(7)(5);
output(118) <= tmp_output(7)(6);
output(119) <= tmp_output(7)(7);
output(120) <= tmp_output(7)(8);
output(121) <= tmp_output(7)(9);
output(122) <= tmp_output(7)(10);
output(123) <= tmp_output(7)(11);
output(124) <= tmp_output(7)(12);
output(125) <= tmp_output(7)(13);
output(126) <= tmp_output(7)(14);
output(127) <= tmp_output(7)(15);

block_64_latch8: latch_v1_4_p_16_p port map (clk(8),input,tmp_output(8));
output(128) <= tmp_output(8)(0);
output(129) <= tmp_output(8)(1);
output(130) <= tmp_output(8)(2);
output(131) <= tmp_output(8)(3);
output(132) <= tmp_output(8)(4);
output(133) <= tmp_output(8)(5);
output(134) <= tmp_output(8)(6);
output(135) <= tmp_output(8)(7);
output(136) <= tmp_output(8)(8);
output(137) <= tmp_output(8)(9);
output(138) <= tmp_output(8)(10);
output(139) <= tmp_output(8)(11);
output(140) <= tmp_output(8)(12);
output(141) <= tmp_output(8)(13);
output(142) <= tmp_output(8)(14);
output(143) <= tmp_output(8)(15);

block_64_latch9: latch_v1_4_p_16_p port map (clk(9),input,tmp_output(9));
output(144) <= tmp_output(9)(0);
output(145) <= tmp_output(9)(1);
output(146) <= tmp_output(9)(2);
output(147) <= tmp_output(9)(3);
output(148) <= tmp_output(9)(4);
output(149) <= tmp_output(9)(5);
output(150) <= tmp_output(9)(6);
output(151) <= tmp_output(9)(7);
output(152) <= tmp_output(9)(8);
output(153) <= tmp_output(9)(9);
output(154) <= tmp_output(9)(10);
output(155) <= tmp_output(9)(11);
output(156) <= tmp_output(9)(12);
output(157) <= tmp_output(9)(13);
```

```
output(158) <= tmp_output(9)(14);
output(159) <= tmp_output(9)(15);

block_64_latch10: latch_v1_4_p_16_p port map (clk(10),input,tmp_output(10));
output(160) <= tmp_output(10)(0);
output(161) <= tmp_output(10)(1);
output(162) <= tmp_output(10)(2);
output(163) <= tmp_output(10)(3);
output(164) <= tmp_output(10)(4);
output(165) <= tmp_output(10)(5);
output(166) <= tmp_output(10)(6);
output(167) <= tmp_output(10)(7);
output(168) <= tmp_output(10)(8);
output(169) <= tmp_output(10)(9);
output(170) <= tmp_output(10)(10);
output(171) <= tmp_output(10)(11);
output(172) <= tmp_output(10)(12);
output(173) <= tmp_output(10)(13);
output(174) <= tmp_output(10)(14);
output(175) <= tmp_output(10)(15);

block_64_latch11: latch_v1_4_p_16_p port map (clk(11),input,tmp_output(11));
output(176) <= tmp_output(11)(0);
output(177) <= tmp_output(11)(1);
output(178) <= tmp_output(11)(2);
output(179) <= tmp_output(11)(3);
output(180) <= tmp_output(11)(4);
output(181) <= tmp_output(11)(5);
output(182) <= tmp_output(11)(6);
output(183) <= tmp_output(11)(7);
output(184) <= tmp_output(11)(8);
output(185) <= tmp_output(11)(9);
output(186) <= tmp_output(11)(10);
output(187) <= tmp_output(11)(11);
output(188) <= tmp_output(11)(12);
output(189) <= tmp_output(11)(13);
output(190) <= tmp_output(11)(14);
output(191) <= tmp_output(11)(15);

block_64_latch12: latch_v1_4_p_16_p port map (clk(12),input,tmp_output(12));
output(192) <= tmp_output(12)(0);
output(193) <= tmp_output(12)(1);
output(194) <= tmp_output(12)(2);
output(195) <= tmp_output(12)(3);
output(196) <= tmp_output(12)(4);
```

```
output(197) <= tmp_output(12)(5);
output(198) <= tmp_output(12)(6);
output(199) <= tmp_output(12)(7);
output(200) <= tmp_output(12)(8);
output(201) <= tmp_output(12)(9);
output(202) <= tmp_output(12)(10);
output(203) <= tmp_output(12)(11);
output(204) <= tmp_output(12)(12);
output(205) <= tmp_output(12)(13);
output(206) <= tmp_output(12)(14);
output(207) <= tmp_output(12)(15);

block_64_latch13: latch_v1_4_p_16_p port map (clk(13),input,tmp_output(13));
output(208) <= tmp_output(13)(0);
output(209) <= tmp_output(13)(1);
output(210) <= tmp_output(13)(2);
output(211) <= tmp_output(13)(3);
output(212) <= tmp_output(13)(4);
output(213) <= tmp_output(13)(5);
output(214) <= tmp_output(13)(6);
output(215) <= tmp_output(13)(7);
output(216) <= tmp_output(13)(8);
output(217) <= tmp_output(13)(9);
output(218) <= tmp_output(13)(10);
output(219) <= tmp_output(13)(11);
output(220) <= tmp_output(13)(12);
output(221) <= tmp_output(13)(13);
output(222) <= tmp_output(13)(14);
output(223) <= tmp_output(13)(15);

block_64_latch14: latch_v1_4_p_16_p port map (clk(14),input,tmp_output(14));
output(224) <= tmp_output(14)(0);
output(225) <= tmp_output(14)(1);
output(226) <= tmp_output(14)(2);
output(227) <= tmp_output(14)(3);
output(228) <= tmp_output(14)(4);
output(229) <= tmp_output(14)(5);
output(230) <= tmp_output(14)(6);
output(231) <= tmp_output(14)(7);
output(232) <= tmp_output(14)(8);
output(233) <= tmp_output(14)(9);
output(234) <= tmp_output(14)(10);
output(235) <= tmp_output(14)(11);
output(236) <= tmp_output(14)(12);
output(237) <= tmp_output(14)(13);
```

```
	output(238) <= tmp_output(14)(14);
	output(239) <= tmp_output(14)(15);

	block_64_latch15: latch_v1_4_p_16_p port map (clk(15),input,tmp_output(15));
	output(240) <= tmp_output(15)(0);
	output(241) <= tmp_output(15)(1);
	output(242) <= tmp_output(15)(2);
	output(243) <= tmp_output(15)(3);
	output(244) <= tmp_output(15)(4);
	output(245) <= tmp_output(15)(5);
	output(246) <= tmp_output(15)(6);
	output(247) <= tmp_output(15)(7);
	output(248) <= tmp_output(15)(8);
	output(249) <= tmp_output(15)(9);
	output(250) <= tmp_output(15)(10);
	output(251) <= tmp_output(15)(11);
	output(252) <= tmp_output(15)(12);
	output(253) <= tmp_output(15)(13);
	output(254) <= tmp_output(15)(14);
	output(255) <= tmp_output(15)(15);
end structure;

library ieee,work;
	use ieee.std_logic_1164.all;
	use IEEE.STD_LOGIC_TEXTIO.ALL;
	use STD.TEXTIO.ALL;
	use work.demux_type.all;
	use WORK.CONV.ALL;

entity demux is
	port ( clk,reset : IN std_logic;
		input : IN std_logic_vector(3 downto 0);
		clk_fft : OUT std_logic;
		output: OUT TYPE_INPUT_FFT
	     );
end demux;

architecture structure of demux is

component latch_v1_4_p_16
	port ( clk : IN std_logic;
		input : IN std_logic_vector(3 downto 0);
		output : OUT TYPE_DEMUX_ARRAY_16x4
	     );
end component;
```

```
component latch_V1_4_p_16_p_16
      port ( clk : IN std_logic_vector(15 downto 0);
                  input : IN TYPE_DEMUX_ARRAY_16x4;
                  output : OUT TYPE_INPUT_FFT
               );
end component;

component clock_generator
      port ( clk : IN std_logic;
                        reset : IN std_logic;
                        output : OUT std_logic_vector(15 downto 0);
                  clk_fft : OUT std_logic;
                        out_clk : OUT std_logic;
                        out_clk1 : OUT std_logic;
                        out_clk2 : OUT std_logic;
                        out_clk3 : OUT std_logic;
                        out_clk4 : OUT std_logic;
                        out_clk5 : OUT std_logic;
                        temp_clk : OUT std_logic
                      );
end component;

      signal bus_64 : TYPE_DEMUX_ARRAY_16x4;
      signal clk_state : std_logic_vector(15 downto 0);
      --signal clk_fft : std_logic;
      signal out_clk, out_clk1, out_clk2 : std_logic;
      signal out_clk3, out_clk4, out_clk5, temp_clk : std_logic;
begin

      u0: clock_generator port map
(clk,reset,clk_state,clk_fft,out_clk,out_clk1,out_clk2,out_clk3,out_clk4,out_clk5,temp_cl
k);
--    clk_fft <= clk_fft;
      u1: latch_v1_4_p_16 port map (clk,input,bus_64);
      u2: latch_v1_4_p_16_p_16 port map (clk_state, bus_64, output);
end structure;
```

Bibliography

[1] M. Choi and Asad A. Abidi, " A 6-b 1.3 Gsample /s A/D Converter in 0.35µm CMOS ", *IEEE Journal of Solid-State Circuits*, Vol. 36, pp.1847-1858, Dec. 2001.

[2] X. Jiang, Z. Wang and M. F. Chang, "A 2GS/s 6-bit ADC in 0.18µm CMOS ", *ISSCC'03* Digest of Technical Papers, 2003.

[3] P.C.S. Scholtens and M. Bertregt, "A 6-b 1.6-Gsamples/s Flash ADC in 0.18-µm CMOS Using Averaging Termination", *IEEE J. of Solid-State Circuits*, Vol. 37 No. 12, Dec. 2002.

[4] J.-E. Eklund, "Robust ADC concept for integrated CMOS systems", *IEE Third International Conference on Advanced A/D and D/A Conversion Techniques and their Applications*, pp. 126-129, Glasgow, UK, July 27-28, 1999.

[5] Chen, C.-I. H., George K., Wang M., McCormick W. and Tsui, J., "2.5 GSPS/1 GHz Wide Band Digital Receiver," *Proceedings of* 29 *Annual International Conference of the IEEE Industrial Electronics*, pp. 1888-1893, Virginia, November, 2003.

[6] K. George, C.-I. H. Chen, and J. B. Y. Tsui "Extension of Two Signal Dynamic Range of Wideband Digital Receivers using Kaiser Window and Compensation Method," *IEEE Trans. Microwave Theory and Techniques*, vol. 55, no. 4, pp. 788–794, April, 2007.

[7] R. Martin and D. J. Secor, "High speed analog-to-digital converters in communication systems: Terminology, architecture, theory and performance," TRW Notes, Nov. 1981.

[8] H. Nyquist, "Certain topics in telegraph transmission theory," *Trans. Of the AIEE,* pp. 617-644, February 1928, 7.

[9] Behzad Razavi, Principles of Data Conversion System Design, IEEE Press, Piscataway, New Jersey, 1995.

[10] P. E. Allen and D. R. Holberg, CMOS Analog Circuit Design, Oxford University Press, Second Edition, Oxford, New York, 2002.

[11] B. C. Baker, Fast Fourier Transforms: A Good Tool for Mixed-Signal System Analysis, Technical Report, Microchip Technology, Inc., 1998, http://www.chipcenter.com/analog/tn014.htm.

[12] A. Moscovici, High Speed A/D Converter: Understanding Data Converter through SPICE, Kluwer Academic Publishers, 2001.

[13] W. T. Colleran, "A 10-bit 100-MS/s A/D converter using folding, interpolating, and analog encoding," Ph.D. Dissertation, Univ. California, Los, Angeles, CA, Dec. 1993.

[14] H. Pan, "A 3.3-V 12-b 50-MS/s A/D converter in 0.6-μm CMOS with over 80-dB SFDR," Ph.D. Dissertation, UCLA, Dec. 1999.

[15] Maxim Integrated Products, ADC and DAC Glossary, 2000, http://www.maximic.com/apponotes.cfm/appnote_number/641.

[16] J. Yoo, "A TIQ Based CMOS Flash A/D Converter for System-on-Chip Applications," Ph.D. Dissertation, The Pennsylvania State University, May, 2003.

[17] A. Pierazzi and A. Boni, "Design Issues for a High Frequency, 0.35μm, 3.3V CMOS Folding A/D Converter," *Advanced A/D and D/A Conversion Techniques and their Applications*, 27-28 July 1999, pp. 115-117.

[18] K. Sushihara, H. Klmura, Y. Okamoto, K. Nishimura, and A. Matsuzawa, "A 6-b 800MSample/s CMOS A/D Converter," *IEEE International Solid-Sate Circuits Conference*, pp. 428-429, 2000.

[19] G. Geelen, "A 6b 1.1 GSample/s CMOS A/D Converter," I*EEE International Solid-Sate Circuits Conference*, 2001, pp. 436-438.

[20] J. Vandenbussche, K. Uyttenhove, E. Lauwers, M. Steyaert and G. Gielen, "A 8-bit 200 MS/s Interpolation/Averaging CMOS A/D Converter," *IEEE Custom Integrated Circuits Conference*, 2002, pp. 445-448.

[21] K. Nagaraj, D. A. Martin, M. Wolfe, R. Chattopadhyay, S. Pavan, J. Cancio, T. R. Viswanathan, "A 700MSamples/s 6b Read Channel A/D Converter with 7b Servo Mode," *IEEE International Solid-State Circuits Conference*, 2000, pp. 426-428.

[22] Y. Li, and E. S. Sinencio, "A Wide Input Bandwidth 7-bit 300-MSample/s Folding and Current-Mode Interpolating ADC," *IEEE Journal of Solid-State Circuit*, Vol.38, No. 8, August 2003, pp. 1405-1410.

[23] C. Sander, M. Clara, A. Santner, T. Hartig and F. Kuttner, " A 6-bit 1.2-GS/s Low-Power Flash-ADC in 0.13-μm Digital CMOS," *IEEE Journal of Solid-Sate*

Circuits, Vol. 40, No. 7, July 2005, pp. 1499-1505.

[24] H. Pan and A. A. Abidi, "Spectral Spurs due to Quantization in Nyquist ADCs," *IEEE Transactions on Circuits and Systems*, Vol. 51, No. 8, August 2004, pp. 1422-1439.

[25] K. Kattmann and J. Barrow, "A Technique for Reducing Differential Non-linearity errors in Flash A/D Converters," in *Proc. IEEE Int. Solid-State Circuits Conf.* Feb. 1991, pp. 170-171.

[26] G. Feygin, K. Nagaraj, and S. Pavan, "A 165 MS/s 8-bit CMOS A/D Converter with Background Offset Cancellation," *IEEE Custom Integrated Circuits Conference*, 2001, pp. 153-157.

[27] H. Okada, Y. Hashimoto, and K. Ishibashi, "Offset Calibrating Comparator Array for 1.2-V, 6-bit, 4-Gsample/s Flash ADCs using 0.13-µm Generic CMOS Technology," *33rd European Solid-State Device Research Conference*, Estoril, Portugal, 16-18 Sept. 2003.

[28] P. M. Figueiredo and J. C. Vital, "Averaging Technique in Flash Analog-to-Digital Converters," *IEEE Transactions on Circuits and Systems*, Vol. 51, No. 2. Feb. 2004, pp. 233-253.

[29] S. Sheikhaei, S. Mirabbasi, and A. Ivanov, "A 4-Bit 5GS/s Flash A/D Converter in 0.18µm CMOS," *IEEE International Symposium on Circuits and Systems* (ISCAS), pp. 6138–6141, May 2005.

[30] K. Poulton, R. Neff, and M. Heshami, "A 4 GSample/s 8b ADC in 0.35-µm CMOS," *reported by Agilent Technologies*, Palo, Alto, CA, 2003.

[31] C. Azzolini, A. Boni and D. Vecchi, "Design of a 2-GS/s 8b Self-Calibrateing ADC in 0.18µm CMOS technology," *2005 IEEE International Symposium on Circuits and Systems*, Kobe, Japan, May 2005.

[32] X. Jiang, M. F. Chang, "A 1-GHz Signal Bandwidth 6-bit CMOS ADC with Power-Efficient Averaging," *IEEE Journal of Solid-State Circu*its, Vol. 40, No. 2, Feb. 2005, pp. 532-535.

[33] K. Uyttenhove and M. S. J. steyaert, " A 1.2-V 6-bit 1.2-GHz Flash ADC in 0.25-µm CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 38, No.7, July 2003, pp. 1115-1122.

[34] R. Taft et al., "A 1.8V 1.6GS/s 8b Self-Calibrating Folding ADC with 7.26 ENOB at Nyquist Frequency," *Proc. of ISSCC*, 2004.

[35] J. Segura, J. L. Rossello, J.Morra, and H. Sigg. A Variable Threshold Voltage Inverter for CMOS Programmable Logic Circuits. *IEEE Journal of Solid-State Circuits*, 33(8):1262-1265, August 1998.

[36] A, Tangel, "VLSI Implementation of the Threshold Inverter Quantization (TIQ) Technique for CMOS Flash A/D Converter Applications" PhD Dissertation, The Pennsylvania State University, Aug. 1999.

[37] J. Yoo, K. Choi, and A. Tangel, " A 1-GSPS CMOS Flash A/D Converter for System-on-Chip Applications", *IEEE CS Annual Workshop on VLSI*, pp.135-139, 2001.

[38] J. Yoo, K. Choi, and D. Lee, "Comparator Generation and Selection for Highly Linear CMOS Flash Analog-to-Digital Converter," *2002 Kluwer Academic Publishier, Printed in the Netherlands.*

[39] J. Yoo, K. Choi, and J. Ghaznavi, "Quantum Voltage Comparator for 0.07 μm CMOS Flash A/D Converters," IEEE Computer Society Annual Symposium on VLSI, pp. 135-139, April 2003.

[40] M. Wang, C. H. Chen, and S. Radhakrishnan, "Low-power 4-b 2.5 GSPS Pipelined Flash Analog-to-Digital Converter in 130 nanometer CMOS", *IEEE Transactions on Instrumentation and Measurement,* vol. 56, no. 3. pp 1064-1073, June 2007.

[41] M. Wang and C. H. Chen, "A High Spurious-Free Dynamic Range 4-bit ADC with Nyquist Signal Bandwidth for Wideband Communication", *Proceedings of IEEE Instrumentation and Measurement Technology Conference*, Warsaw, Porland, May 1-3, 2007.

[42] M. Wang and C. H. Chen, "Architecture and Design Synthesis of 2.5 Gsamples/s 4-b Pipelined Flash ADC in SoC Application", *Proceedings of the 31st Annual Conference of IEEE Industrial Electronics Society*, pp. 2224-2230, Raleigh, North Carolina, USA, Nov. 6-10, 2005.

[43] M. Wang and C. H. Chen, "Design Synthesis and Performance Measurement of Pipeline Flash ADC for SoC Applications", *Proceedings of the 2005 IEEE Instrumentation and Measurement Technology Conference*, pp. 19-24, Ottawa, Ontario, Canada, USA, May 17-19, 2005.

[44] H. Pan and A. A. Abidi, "Spatial Filtering in Flash A/D Converters," *IEEE Trans. Circuit Syst. II,* vol. 50, pp. 424-436, Aug. 2003.

[45] Y. Gendai, Y. Komatsu, S. Hirase, and M. Kawata, "An 8b 500MHz ADC," *Proceedings of. IEEE International Solid-State Circuits Conference*, pp. 172-173, Feb. 1991.

[46] H. Pan, M. Segmai, M. Choi, J. Cao, and A. A. Abidi, “A 3.3-V 12-b 50-MS/s A/D Converter in 0.6-μm CMOS with over 80-dB SFDR,” *IEEE Journal of Solid-State Circuits*, Vol. 35, No. 12, Dec. 2000, pp.1769-1780.

[47] M. Shinagawa, Y. Akazawa, and T. Wakimoto, “Jitter analysis of high-speed sampling systems,” *IEEE Journal of Solid State Circuits,* vol. SC-25, pp. 220-224, Feb. 1990.

[48] T. Wakimoto, Y. Akazawa, and Y. S. Konaka, “Si bipolar 2-GHz 6-bit flash A/D conversion LSI,” *IEEE Journal of Solid-State Circuits*, vol. 23, pp. 1345-1350, Dec. 1988.

[49] R. J. Plassche, and P. Baltus, “An 8-b 100MHz full Nyquist A/D converter,” *IEEE Journal of Solid-State Circuits*, Vol. 23, pp. 1334-1344, Dec. 1988.

[50] L. E. Larson, “High-speed analog-to-digital conversion with GaAs technology: prospects, trends and obstacles,” in *IEEE International Symposium on Circuits and Systems*, pp. 2871-2878, 1988.

[51] M. J. Demler, High-Speed Analog-to-Digital Conversion, Academic Press, Inc. San, Diego, California, 1991.

[52] A. Dixit, Design of a High Speed Analog-to-Digital Converter for Data Storage Channels, Master Thesis, Carnegie Mellon University, 1998.

[53] C. W. Mangelsdorf et al., “A 400-MHz Input Flash Converter with Error Correction,” *IEEE Journal Solid-State Circuits,* Vol. SC-25, pp. 184-191, Feb. 1990.

[54] B. Peetz. B. D. Hamilton, and J. Kang, “An 8-Bit 250 Megasamples/sec A/D Converter,” IEEE Journal Solid-State Circuits, Vol. SC-21, pp. 997-1002, Dec. 1986.

[55] Jerrold Henry Krenz, Electronic Concept: An Introduction, Cambridge Unviersity Press, Technology & Inductrial Arts, ISBN 05216682, 2000.

[56] F. Lai and W. Hwang, “Design and implementation of differential cascode voltage switch with pass-gate (DCVSPG) logic for high-performance digital systems,” *IEEE Journal of Solid-State Circuits*, vol. 32, no. 4, pp. 563-573, April 1997.

[57] S. Sheikhaei, S. Mirabbasi and A. Ivanov, “A 4-Bit 5GS/s Flash A/D Converter in 0.18μm CMOS, “*Proc. of the 2005 IEEE Instrumentation and Measurement Technology Conference*, Ottawa, Ontario, Canada, May 17-19, 2005.

[58] W. Ellersick, C. K. Yang, M. Horowitz, and W. Dally, “GAD: A 12GS/s CMOS

4-bit A/D converter for an equalized multilevel link," *IEEE Symp. VLSI Circuits, Dig. Tech. Papers*, June 1999, pp. 49-52.

[59] S. Naraghi and D. Johns, "A 4-bit analog-to-digital converter for high-speed serial inks," *Micronet Annual Workshop,* April 26-27, 2004, Aylmer, Quebec, Canada, pp. 33-34.