A Low Cost Asynchronous Eye Diagram Reconstruction System for High Speed Links

by

Shijie Zheng

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Electrical Engineering and Computer Science
May 18, 2013

Certified by .....
Vladimir Stojanovic
Associate Professor
Thesis Supervisor

Certified by .....
Pablo Acosta
Staff Design Engineer, Analog Devices, Inc.
Thesis Supervisor

Accepted by .....
Prof. Dennis M. Freeman
Chairman, Master of Engineering Thesis Committee

A Low Cost Asynchronous Eye Diagram Reconstruction System for High Speed Links

by

Shijie Zheng

Submitted to the Department of Electrical Engineering and Computer Science
on May 18, 2013, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

As link communication data rate increases, there is an increasing need for a more cost effective way to test and monitor signal integrity in link communication systems. Specifically, eye diagrams are valuable visual aids to analyze and quantify digital signal quality. This thesis presents a novel low cost eye diagram reconstruction system using asynchronous undersampling technique, which solves a key problem in performance monitoring in systems where synchronous sampling is not available, such as video switches. Existing works are studied and compared to this work in performance and cost. The proposed system is designed as a system-on-chip (SOC) and contains an undersampling ADC, aliased frequency estimator and a simple reconstruction algorithm. Major building blocks are implemented and simulated in 65nm CMOS process. Extensive system level analysis and simulations demonstrate functionality and performance of the system working at 10Gb/s maximum data rate.

Thesis Supervisor: Vladimir Stojanovic
Title: Associate Professor

Thesis Supervisor: Pablo Acosta
Title: Staff Design Engineer, Analog Devices, Inc.

Acknowledgments

My thanks go to all of those who were influential for the completion of this thesis and my personal development during my 5-year career at MIT. I am limited by space and words to express my emotions, but my most sincere thanks and wishes go to my dearest family, friends and mentors for your being in my life.

Foremost, I would like to express my gratitude towards Prof. Vladimir Stojanovic, my MIT thesis advisor. He provided immediate and essential feedbacks when I needed them. He did not only point me in the right directions for my thesis work, but also spent time to discuss my plans for future, and I am grateful for his words and efforts to push me onto higher grounds.

Pablo Acosta, my advisor at ADI, offered important technical advice and tailored the thesis topic to be open ended and allowed me the freedom to explore. I extremely appreciates his honest and insightful evaluations of my work and me as an engineer.

I thank Kimo Tam and the ADI High Speed Signal Integrity Group for supporting me for two internships and VI-A fellowship. Fellow interns (Branislav Jovanovic, Tayyar Rzayev, Natasa Trkulja, Alex Jurkov, etc.) and friendly engineers (Jeremy Walker, Ben Walker, Andy Wang, Mike Germain, Yuki Handa and others) treated me as a member of the family and made my experience in ADI memorable.

I would like to thank all my friends in MIT and abroad. It was never easy to accomplish what we all have achieved, and to that I say good job. I am grateful for the life long friendships we have built and will sustain. My special thanks go to Jorge Simosa, Krishna Settaluri and Danny Bankman for your lasting supports and charming personalities.

Last but not least, my most sincere thanks belong to my beloved family. My parents, Yao Hua Zheng and Dong Mei Liu who instilled within me intelligence, curiosity and passion, provided me with the most intense care and love. My younger brother, Richard Zheng, has been my best friend all along and I will support and wish you best on your own journey. None of this will be possible without you. I love you.

Contents

1 Introduction **17**

1.1 Motivations 18

1.2 Past Works 18

1.2.1 Synchronous Reconstruction 19

1.2.2 Asynchronous Reconstruction 20

2 Proposed System **23**

2.1 System Architecture 23

2.2 Sample and Hold 25

2.2.1 Possible Implementations 25

2.2.2 Sampling Clock and Analog to Digital Converter 27

2.3 Lambda Estimation 27

2.3.1 Subrate Extractor 28

2.3.2 Aliased Frequency by Counting 29

2.3.3 Pseudorandom Binary Sequence 30

2.3.4 Transition Density in PRBS 31

2.3.5 Local Fluctuation of Transition Density 32

2.3.6 Counter Resolution and Divider Length 35

2.4 Reconstruction Method 37

3 Circuit Implementation **41**

3.1 Sample and Hold 41

3.1.1 Pass Gate Self Bandwidth 42

3.1.2 S&H Implementation 43
3.1.3 Results 47
3.2 Sampling Clock Jitter 51
3.2.1 Eye Match Rate 51
3.2.2 Sample Clock Jitter Effect on Reconstruction 52
3.2.3 Oscillator Jitter Analysis 53
3.2.4 Ring Oscillator Phase Noise 58
3.3 Lambda Estimation 60
3.3.1 Frequency Divider for High Speed Data 60
3.3.2 CML Latch and Frequency Divider 61
3.3.3 CML Divider Performance 64
3.3.4 Lambda Estimator Implementation 67
3.4 Digital Reconstruction 69
3.4.1 Memory and Memory Controller 71
3.4.2 Tau Calculator 71
3.4.3 Eye Opening Finder 72
3.4.4 Coarse Search Correct 75
3.4.5 Fine Correct 77
3.4.6 RTL Synthesis and Area 79

4 Simulation and Results **83**
4.1 Behavioral Models 83
4.2 Test Strategies 84
4.3 System Test 85
4.4 Results 87

5 Conclusion **89**
5.1 Summary 89
5.2 System Usage and Application 90
5.3 Future Improvements and Explorations 92
5.3.1 Fabrication 93

5.3.2 Area Saving 93
5.3.3 Other Input Signals 94
5.3.4 External Control Features 95

List of Figures

1-1 One version of synchronous reconstruction using comparator and distribution statistics 19

1-2 Comparison between ideal (left) and reconstructed eye (right) with method in [1] with 512 point FFT. The plot in the middle is the periodogram of the transformed signal, with distinct frequency spikes at $2\pi\lambda_b$ 21

2-1 System block diagram 24

2-2 Example switched emitter follower track and hold circuit before ADC 26

2-3 Example fast bootstrapped switch implementation 26

2-4 Power spectrum of original data and its subrate 28

2-5 Block diagram for PRBS4 generation 30

2-6 Shift register state for a 0 to 1 transition 31

2-7 PPM error w/ respect to 25% against N bit shift register 32

2-8 Local 0 to 1 transition density fluctuation for run 2 to 5 34

2-9 PRBS13 fluctuation v.s. Number of runs on Log-Log Plot 34

2-10 2D lambda estimation error plot for PRBS13 37

2-11 Iteratively corrected lambda algorithm at work 40

3-1 Passgate self bandwidth 42

3-2 Schematic of sample and hold 43

3-3 Schematic of input amplifier 44

3-4 Schematic of output amplifier 44

3-5 Small signal model of equalizing input amplifier 45

3-6 Schematic of pass gate 45
3-7 Simple model of S&H 46
3-8 Layout of sample and hold 47
3-9 Monte Carlo plots of sample and hold in slow corner 49
3-10 Monte Carlo plots of sample and hold in nominal corner 49
3-11 Monte Carlo plots of sample and hold in fast corner 50
3-12 Transient waveforms of internal nodes in sample and hold 50
3-13 Actual channel eye v.s. reconstructed eye from S&H samples 50
3-14 Comparison of reconstructed eye w/ and w/o jitter 52
3-15 Eye diagram evolution with respect to increasing jitter 53
3-16 Match rate v.s. RMS jitter 53
3-17 Unbounded jitter free running oscillator with respect to time 54
3-18 Phase noise plot of period to period jitter 59
3-19 Phase noise plot of 3072 point duration 59
3-20 CMOS frequency divider (divide by 2) 60
3-21 Circuit schematic of conventional CML latch 61
3-22 Circuit schematic of f_t doubling latch 62
3-23 Block diagram of a new frequency divider circuit 62
3-24 Circuit schematic of proposed CML latch 63
3-25 CML latch based frequency divider 64
3-26 Clock modulation when divider oscillates 64
3-27 Sensitivity curves of conventional and new divider 65
3-28 Count of edges of conventional divider v.s. channel length in slow corner 66
3-29 Count of edges of new divider v.s. channel length in slow corner . . . 66
3-30 Block diagram of lambda estimator 67
3-31 Place and route output of lambda estimator 68
3-32 Flow chart of reconstruction process 69
3-33 Block diagram of reconstruction system 70
3-34 Filtered and unfiltered eye opening waveforms 73
3-35 Frequency transfer function running average filter with 8 taps 73

3-36 Filter outputs of wrong (left) and correct (right) λ_b estimation 74
3-37 Coarse search correct FSM 75
3-38 Histogram of lambda estimation error 76
3-39 Fine correct FSM 78
3-40 Ambiguity in correction direction 78
3-41 PNR logic only output 80

4-1 Test bench example for fine correct module 85
4-2 Final system test schematic 86
4-3 Example output waveforms 86
4-4 Output of high data rate and large eye 87
4-5 Output of high data rate and small eye 87
4-6 Output of low data rate and large eye 88
4-7 Output of low data rate and small eye 88

5-1 Search step upper bound v.s. transition densities 91
5-2 Example usage of proposed system as signal quality monitor 92
5-3 Example usage of proposed system in equalizer adaptation loop . . . 92
5-4 Reconstruction results of 2048 samples 94

List of Tables

2.1 Transition Probabilities between Perfect Clock and Random Sequence 28

3.1 Qualitative comparison of filter methods 74

3.2 Parameter comparison for low and high resolution fine correct 79

Chapter 1

Introduction

As device size scales down, computing and communication systems require faster link I/Os in order to supporthigher data and computation throughput requirements. This poses great challenge for signal integrity between a transmitter (TX) and a receiver (RX). Equalizers and clock and data recovery loops (CDR) are developed to reduce bit error rate (BER) in received data. As an evaluation tool, eye diagrams give a comprehensive picture of signal conditions before and after receivers; they are also used to analyze jitter, estimate BER, characterize channels, etc. High quality eye diagrams are typically generated with synchronous sampling and such analyzers are often expensive.

In this chapter, motivations and past works on eye diagrams reconstruction techniques are discussed, which serve as the foundation for the proposed system.

Chapter two describes the architecture of the proposed system, the major blocks' principles of working and the motivations for the design decisions made.

Chapter three describes the design of specific blocks on circuit level, as well as theoretical analysis that leads to system level design choices and optimizations. The analog and digital blocks will be discussed separately as well.

Chapter four describes the simulation strategy due to the size and complexity of the system. Test cases are used to verify the functionality of major blocks and system level simulations are also run to ensure correctness. Some more simulation results are presented in this chapter as well.

Chapter five concludes by showing comparisons in several aspects of the system with past works. Possible future expansions to the project are also discussed.

1.1 Motivations

An eye diagram proves to be very valuable in quick assessment of the behavior of high speed link or receiver. However, it is very difficult to probe any trace on an evaluation board. One example would be in a high speed switch box, in which a signal travels through many nodes. It is very difficult to physically put a probe from an analyzer onto any trace of interest inside the switch box. Another example would be for debugging a packaged chip for signal quality on any metal trace on the die. Either case would require a careful characterization plan to make signals of accessible when testing the product.

If there is an on-chip system that would allow any incoming signal to be monitored and generates a corresponding eye diagram, this would provide much flexibility and better evaluations. Such a system should be compact enough and have easy interface with external signals and terminal computers. Thus, in addition to generating moderate quality eye diagram, this on-chip system should also have well defined power and area constraints in its specifications. The project in this thesis aims to provide a solution for simpler signal monitoring capabilities by designing, analyzing and implementing an on chip eye diagram reconstruction system in standard TSMC 65nm CMOS process.

1.2 Past Works

There are several different approaches to eye diagram reconstruction. Both sychronous and asynchronous techniques have been developed and studied. Most of the these approaches are circuit board level designs due to complexity and no further work is observed for improvements and optimizations.

The synchronous technique involves CDR loop to explicitly extract the data fre-

quency and phase shift the recovered clock to find out information about the eye diagram. The asynchronous technique relies heavily on digital signal processing (DSP) and requires an understanding of undersampling and its effect on the original signal. This directly factors into the reconstruction method most people use when doing undersampling.

1.2.1 Synchronous Reconstruction

Synchronous reconstruction means finding explicitly what the incoming data rate is using a CDR loop, and interpolate the clock phase to calculate signal distribution for a given phase location. With a data clock, one can either use a multi-bit analog to digital converter to directly sample points , or a comparator (with 1 bit information) to obtain the density of signal traces. Figure 1-1 shows how such technique works.

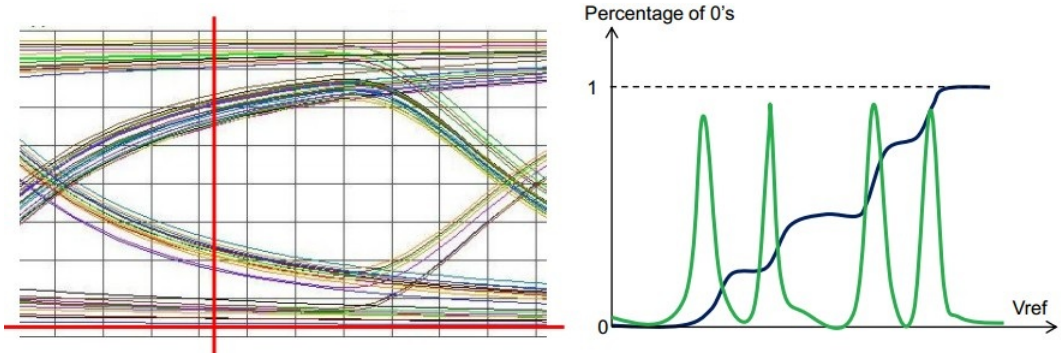


Figure 1-1: One version of synchronous reconstruction using comparator and distribution statistics

The vertical bar in the left subplot designates a specific phase location from the phase interpolated clock, i.e. every clock cycle the edge falls right at where the vertical bar is. The horizontal bar designates the voltage reference level into the comparator. By sweeping the reference voltage from bottom to top, we obtain a cumulative distribution of percentage of 0's shown as the blue curve in the right subplot. Taking the derivative of this curve gives the density curve at which the signal traces are concentrated. This information then can be stored, which reflects the signal distribution at that particular phase location. Subsequently we move clock phase to next location and repeat what we did. Eventually a 2 dimensional grid will

be formed (size is number of phase locations by number of reference voltage levels) and an eye diagram is formed by plotting the corresponding density into each grid.

One major advantage of this approach is the simplicity of using a single comparator with very descent reconstruction quality. On the other hand, high performance CDR loops are harder to design in a high speed condition and with a scaled down process. The comparator performance also becomes critical; it should have low offset, high bandwidth and very high gain. Moreover, the memory size is unnecessarily big since it contains many grids that have zero density. The disadvantages relate heavily to cost issues and researchers proposed asynchronous techniques to get around some of these issues.

1.2.2 Asynchronous Reconstruction

Signal reconstruction using undersampling (also known as sub-sampling) techniques has been studied by multiple groups, particularly in the optical communication area due to high data rate. [1] describes the theoretical foundation for a verified asynchronous undersampling technique for eye diagram reconstruction. This work focused on digital signal processing approach, in which a non-linear transformation of the input signal will make the data frequency manifest itself as a distinct spike in the frequency spectrum. After undersampling, this frequency spike will be aliased to a digital frequency given by

$$\Omega_{aliased} = \frac{mod(f_{data}, f_{sample})}{f_{sample}} \times 2\pi \tag{1.1}$$

where f_{data} is the data frequency and f_{sample} is the undersampling frequency, which is smaller than the nyquist frequency of the data. The factor in front of 2π (let's call it λ_b) is essentially the decimal part of f_{data}/f_{sample}, and it tells how many samples will be within a period. For example, a λ_b of 1/3 means there will be 3 samples within a period before the 4th sample wraps around to the beginning of the period again.

The reconstruction method, as described in [1], involves a pseudo Fourier Transform using $2\pi\lambda_b$ as the synchronous frequency. Then the time location of a specific

sample is the phase of this transform given a window of samples around it and a window sequence W_k. Equations 1.2 and 1.3 show the reconstruction algorithm.

$$\tau_n = \frac{angle(Y_n(\omega))}{2\pi} \tag{1.2}$$

$$Y_n(\omega) = \sum_{k=-K}^{K} y_{n+k} W_k e^{-2\pi\lambda_b nj} \tag{1.3}$$

Periodogram is used to find the power spectrum of the signal, and thus λ_b. Due to resolution error for λ_b, we can not simply increment the time location by adding λ_b to the previous time step, and that's why the above reconstruction method is proposed. Both [2, 3] have experimentally verified this reconstruction method by assembling a test board, which includes large DSP units. The result is satisfactory since it provides good approximation to the bit error rate, but some important features in the eye diagram, such as data dependent jitter, might be lost due to estimation errors. Besides, due to the computationally heavy nature of this method, large processing engines as FFT engines and complex math modules are used. Synthesizing such blocks into a system on chip would pose large constraints on power and area. Figure 1-2 shows a Matlab simulation, which compares an ideally simulated channel eye and reconstructed eye with this method.

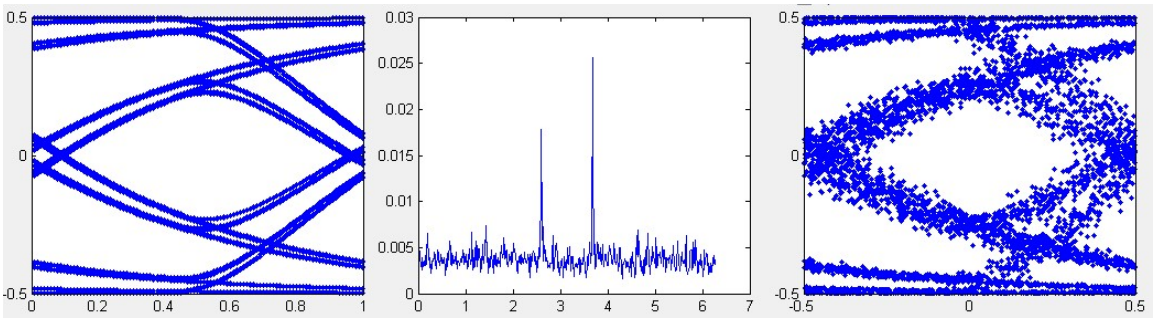


Figure 1-2: Comparison between ideal (left) and reconstructed eye (right) with method in [1] with 512 point FFT. The plot in the middle is the periodogram of the transformed signal, with distinct frequency spikes at $2\pi\lambda_b$

Such undersampling technique is not limited to eye diagram reconstruction but signal reconstruction in general. A recent paper [4] has used similar idea for periodic

signal acquisition. The same λ_b is used, but the reconstruction method is simply to increment the previous time location by λ_b, and wraps around by an modulus operation, given by equation 1.4.

$$\tau_n = mod(\tau_{n-1} + \lambda_b, 1) \tag{1.4}$$

The precision of λ_b is achieved by more points in FFT and pre and post conditioning of the signals, such as reducing spectral leakage and interpolation. Sampling and data jitter are compensated by updating λ_b constantly and using the new estimate for reconstruction. Due to the simplicity of the reconstruction method, when λ_b is very close to the true value, the reconstructed signal quality is really high, so there will be no error induced by calculations as in [1]. However, more computation power is put into the better estimation of λ_b, which means the bottleneck for area and power still exists.

An important point to note is that λ_b has to be an irrational number in order to have "random" phase locations within a period, i.e. if λ_b were a rational number like 1/3, then only 3 phase locations are sampled (which means it is more or less synchronous sampling), thus no complete eye diagram can be generated. This idea will be important throughout the project and it has implications to what the undersampling frequency can be.

Chapter 2

Proposed System

This chapter proposes an architecture for a new system that avoids the disadvantages of the works discussed previously. The system is divided into several blocks for ease of analysis, design and implementation. The specifications and theory behind each block will be discussed in more details in the following sections. Optimizations and design choices are also explained. The differentiating factors of this proposed system lie of simple aliased frequency estimation at the cost of estimation time and reduced computing power from self correcting reconstruction algorithm.

2.1 System Architecture

Based on these previous works, an asynchronous undersampling technique seems more advantageous, since no prior knowledge of the bit period is required. However, this brings constraints on power and area for the system due to heavy digital signal processing. The most important parameter here is λ_b: a good estimate would allow a simple reconstruction method as described in [4], while a crude approximation requires a more sophisticated reconstruction method using complex math.

The proposed system gets around some of these issues by explicitly finding a subrate of the data frequency using time division modules and a simple method for estimating lambda. One assumption has to be made; the input data sequence has to be random or close to random in the transition density sense, i.e. the 0 to 1 transition

has to be near 25%. It is an acceptable assumption since the same needs to be true for a periodogram based lambda estimation, and most testing data sequences, such as pseudorandom binary sequence, are designed to approximate random data input. This idea will be discussed more in the following sections. With a reasonable estimate of λ_b, the reconstruction method emulates the one used in [4], but iteratively corrects for the error of lambda by looking at the eye diagram drift over groups of samples. Figure 2-1 shows the block diagram of the proposed system.

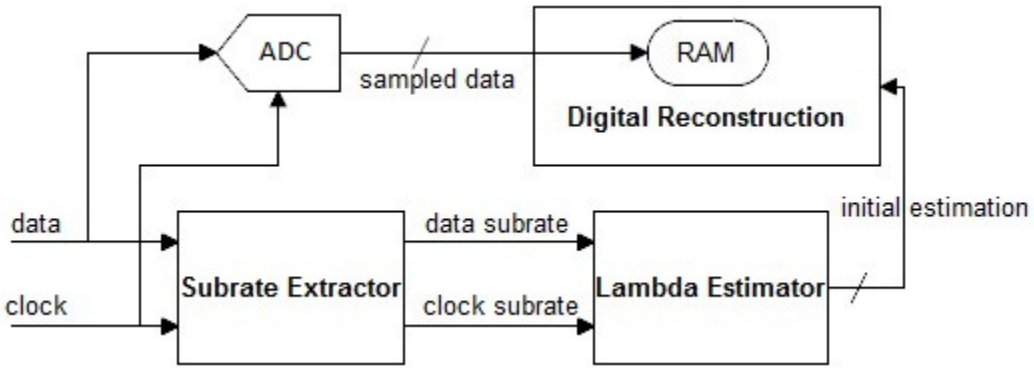


Figure 2-1: System block diagram

1. ***Subrate Extractor*** Takes both the input data and sampling clock and outputs two divided clocks at the subrates of data and sampling clock frequencies.

2. ***Lambda Estimator*** Takes both the input data and sampling clock subrate and outputs an estimated lambda by counting according to input clock frequencies.

3. ***Undersampling ADC*** Undersamples the input data analog waveform and stores 3K bytes of information on RAM (3072 data points).

4. ***Reconstruction*** Digital block of the system that takes the estimated lambda and calculates a time location for each point in the memory. Implemented in Verilog and will be synthesized.The embedded RAM will be 6K bytes, 2 bytes for each (τ, y) pair.

For monitoring purpose, heavy spectrum analysis is avoided by extracting the subrate of data clock directly, which can be easily achieved by a chain of dividers.

Then the lambda estimator block, which consists of two simple counters, will take the two subrate clocks and generate an estimate of λ_b with reasonable precision. Meanwhile, the 8-bit undersampling ADC will record 3072 points (3K bytes of data) and store in the on chip RAM. The reconstruction block, which will be implemented and synthesized in Verilog code, will take the estimated lambda and calculate a time location for each sample and iteratively correct for the estimation error in lambda. This system gets rid of complicated DSP module completely and only simple addition is required for reconstruction. Later sections will explain each sub block in more details.

2.2 Sample and Hold

The sample and hold circuit along with the ADC serves as the equivalent as the signal probe in an oscilloscope. The bandwidth of a signal probe is the determining factor for the largest data rate we can probe. The targeted maximum data rate in this application is 10Gbps. The analog "fundamental frequency" of a 10Gbps data stream is 5GHz. As we will see later, the actual data sequence contains all other frequency content other than multiples of 5GHz, but the signal processing thinking of Nyquist frequency still applies. In order to preserve enough high frequency content, a specification of at least 10GHz (twice the fundamental frequency) is decided. The sample and hold circuit would also require a high linearity for minimal distortion in the eye shape. Power isn't a major concern since monitoring happens for a short period of time. The sample and hold should also be able to drive ADC's that are sampling at about 200MHz.

2.2.1 Possible Implementations

For high speed sample and hold circuit, there are mainly three popular topologies: differential switched source follower, bootstrapped switches, and simple passgates. Switched source followers are often used in bipolar transistor (BJT) circuits in which devices are be turned off sharply, with high intrinsic gain, and a lot of headroom

is available with higher power supply. Figure 2-2 shows a schematic of a switched emitter follower track and hold in with BJTs. In this 65nm CMOS process due to limited power supply and headroom, stacking up transistors should be avoided. Leakage in small length devices would also be an issue in this topology.

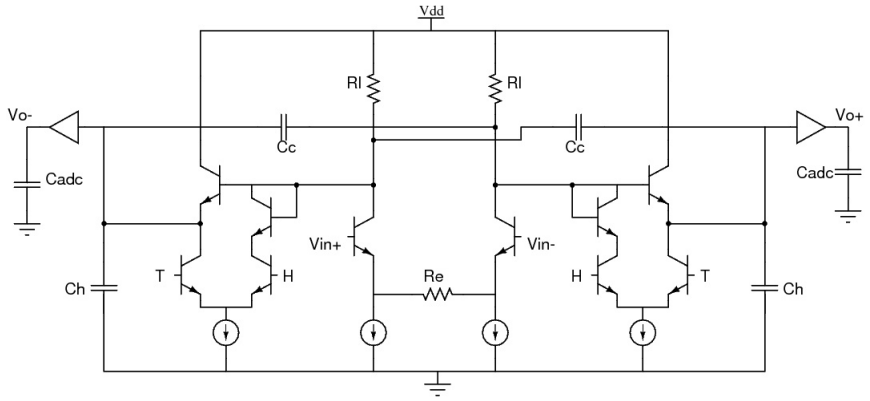


Figure 2-2: Example switched emitter follower track and hold circuit before ADC

Bootstrapped switches proved to have the best linearity and bandwidth.Figure 2-3 shows one example of a fast bootstrap switch implementation [5]. However, playing with high voltage (i.e. doubling supply voltage for bootstrapping) is "dangerous" in this process (the gate breakdown voltage is quoted to be around 1.7V) when nominal power supply is 1.2V.

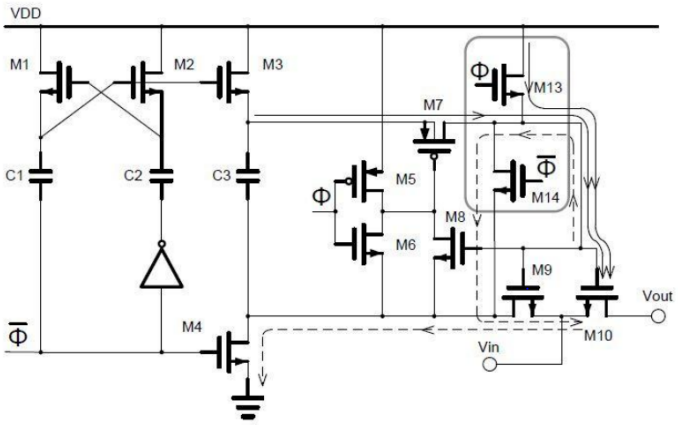


Figure 2-3: Example fast bootstrapped switch implementation

Simple passgates might provide the needed performance with moderate linearity and speed property, but its implementation simplicity, which is just PMOS and NMOS

back to back is extremely attractive. Several issues need to be addressed using pass gates, for example the directly coupling parasitic capacitance from drain to source in short channel devices. Detailed implementation of the sample and hold circuit will be presented in later chapter.

2.2.2 Sampling Clock and Analog to Digital Converter

For the scope of the project, no dedicated sampling clock or ADC will be designed from scratch. I will assume that such blocks will be provided given the required specification. As a result, part of the system level analysis will involve sampling clock jitter's effect on reconstructed eye quality. The undersampling clock is supposedly much slower than the data frequency, however still will be around 200MHz. In this application, the ADC needed would be relatively low resolution (8 bits) but high speed (200MHz) with the designed sample and hold as the front end. Topologies that can meet the specification could either be an 8 bit flash ADC with a sacrifice on area or a simple pipeline ADC with a bit more complexity. The reason that a successive approximation register (SAR) ADC comes in as last choice is due to more difficult clock management since the generated clock will have to faster than the actual undersampling frequency to run the ADC and area constraints coming from the capacitor array. After determining the required specifications, these two blocks will be simulated using behavioral models other than the full transistor level.

2.3 Lambda Estimation

Lambda estimation block eliminates the need for an FFT module to find a periodogram of the input signal. It uses a chain of frequency dividers to create near-perfect square waves with low enough frequencies so that simple counters can be used to estimate λ_b. This section will discuss the theory behind how subrate extractor, lambda estimation by counting and an analysis of Pseudorandom Binary Sequence (PRBS) without implementation details.

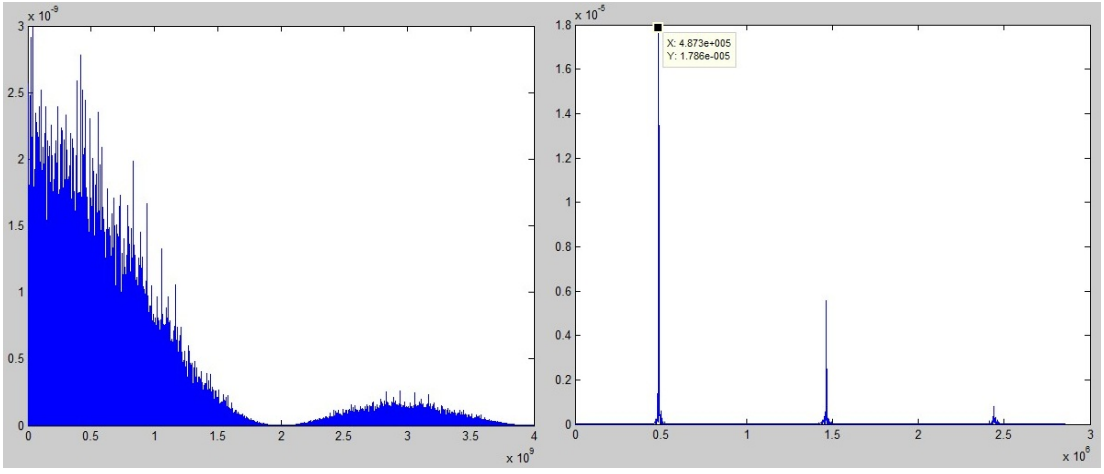
2.3.1 Subrate Extractor

A random input sequence is not only DC-balanced, but it also contains equal probabilities of transitions on average. Table 2.1 compares the transition densities of a perfect clock and a random sequence, both of which on average have the same number of 0's and 1's. This fact could be exploited to extract a subrate (or sub-harmonic) of the data rate [6]. With a chain of N divide by 2 blocks, the output waveform will converge to a clock output at the frequency of $0.25 \cdot f_{data}/2^N$.

Transition Type	$0 \rightarrow 0$	$0 \rightarrow 1$	$1 \rightarrow 0$	$1 \rightarrow 1$
Perfect Clock	0	0.5	0.5	0
Random Sequence	0.25	0.25	0.25	0.25

Table 2.1: Transition Probabilities between Perfect Clock and Random Sequence

For a non-return-zero (NRZ) data sequence, the power spectrum has frequency nulls at multiples of data clock frequency. Figure 2-4a shows the power spectrum of a random sequence of NRZ data of frequency 2GHz. After dividing the data frequency by 1024, we obtain a signal that's close to perfect clock at the frequency of $2GHz/4096 = 488.3kHz$, and Figure 2-4b shows the power spectrum of the divided output, which represents a near perfect square wave with small jitter noise.



(a) Power spectrum of random NRZ data (b) Power spectrum of subrate derived from dividing data by 1024

Figure 2-4: Power spectrum of original data and its subrate

A divide by two stage is simple to implement, conceptually a flip flop with inverted output fed back to its input. The front stages for data division need to be CML logic since it works with high speed, followed with CMOS stages after a level conversion. The clock division can be directly be CMOS type due to its much lower speed. There needs to be two more stages in the clock division chain due to the extra 0.25 factor in data transition density.

2.3.2 Aliased Frequency by Counting

After the subrate extractor, the two derived low frequency clock signals will be passed through the lambda estimator. The scaling property arithmetic guarantees that the two derived clocks will give the same λ_b as the original ones, namely

$$\lambda_b = \frac{mod(f_{data}, f_{sample})}{f_{sample}} \tag{2.1}$$

$$= \frac{mod(f_{data}/2^D, f_{sample}/2^D)}{f_{sample}/2^D} \tag{2.2}$$

$$= \frac{mod(f'_{data}, f'_{sample})}{f'_{sample}} \tag{2.3}$$

where D is the number of division stages and f' denotes the subrate of the original data or sampling frequency. We then use the approximation, $N = floor(f \times T)$, for which T is an arbitrary time period and N is the number of clock edges for a given clock frequency within that period. When T gets large enough, we can use N/T to approximate frequency to a very high precision. Then

$$\lambda_b = \frac{mod(f'_{data}, f'_{sample})}{f'_{sample}} \tag{2.4}$$

$$\approx \frac{mod(N'_{data}/T, N'_{sample}/T)}{N'_{sample}/T} \tag{2.5}$$

$$= \frac{mod(N'_{data}, N'_{sample})}{N'_{sample}} \tag{2.6}$$

This allows us to use simple counters to estimate λ_b. For a given count of N'_{sample}, by looking at a separate counter counting the data subrate clock, we can obtain a

good estimate of lambda. For example, for a 10 bit counter, because the data rate is higher than the sampling rate, by the time the sampling clock counter reaches the full count (1024), the data clock counter would have wrapped around multiple times and the number remaining in the counter is conveniently $mod(N'_{data}, N'_{sample})$, and even better is the fact that now this 10 bit number could be reinterpreted as a fixed point number with 10 bit precision after decimal point, so also performing the dividing by N'_{sample} part.

One realization is that both subrate extractor and counters could be a chain of dividers, which means they both provide some averaging function but for different purposes. This view gives rise to an optimal problem on whether there is an sweet spot in number of division stages and counter resolution. The following sections analyzes the characteristics of PRBS and then moves onto the final design choices.

2.3.3 Pseudorandom Binary Sequence

Pseudorandom Binary Sequence (PRBS) is a deterministic bit sequence that has characteristic similar to a true random bit sequence. A typical way of generating a PRBS is linear feedback shift register, in which there is N bit shift register and several taps are used to generate the next LSB in the register by a simple xor and the MSB of the register becomes the current output. Maximum length PRBS generation is achieved by selecting the corresponding taps such that the register cycles through all possible states except the all zero state. As a result, there number of bits in a PRBS is $2^N - 1$. Figure 2-5 shows the block diagram of how a PRBS4 is generated. PRBS is very useful in testing channel signal integrity since it provides almost all possible data pattern scenarios for inter-symbol interference (ISI) due to its pseudorandomness, thus very relevant in this context.

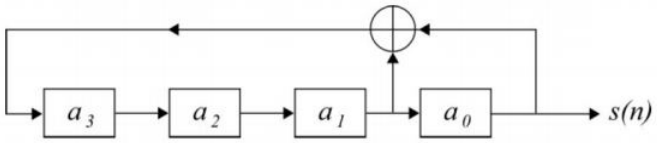


Figure 2-5: Block diagram for PRBS4 generation

2.3.4 Transition Density in PRBS

PRBS asymptotically has characteristics of a true random sequence, one of which is transition density. In a truly random binary sequence, the transition density for any of the four types, 0 to 0, 0 to 1, 1 to 0 and 1 to 1, approaches 25% on average. For a PRBS, an explicit form of transition density can be found and indeed it approaches 25% as shifter register size gets larger and larger. The PRBS generation method discussed above runs through all possible states in shift register other than all 0 states. The MSB is always used as the output.

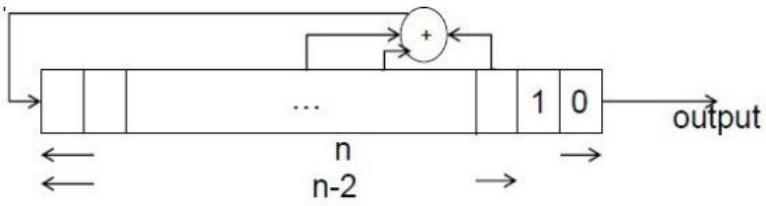


Figure 2-6: Shift register state for a 0 to 1 transition

As a result, the 0 to 1 transition happens when the two MSBs are 0 and 1. Figure 2-6 shows the possible states the shift register has for a 0 to 1 transition to occur. In this case, there are totally 2^{n-2} possible states since the other $n-2$ bits can be anything. As a result, the 0 to 1 transition density is given by

$$Pr(0 \to 1) = \frac{2^{n-2}}{2^n - 1} \tag{2.7}$$

The same argument applies for 1 to 1 and 1 to 0 transitions. The missing transition is for 0 to 0 since all zero state is invalid. The following plot shows the 0 to 1 transition density in ppm error with respect to 25% against n. The PPM error becomes neglible as n increases. When n is around 16, the error is already small enough, 15ppm. It is also important to note that these transition density applies for a complete cycle through the bit sequence. It also means that the the error gets smaller for a longer time due to more bits averaging effect.

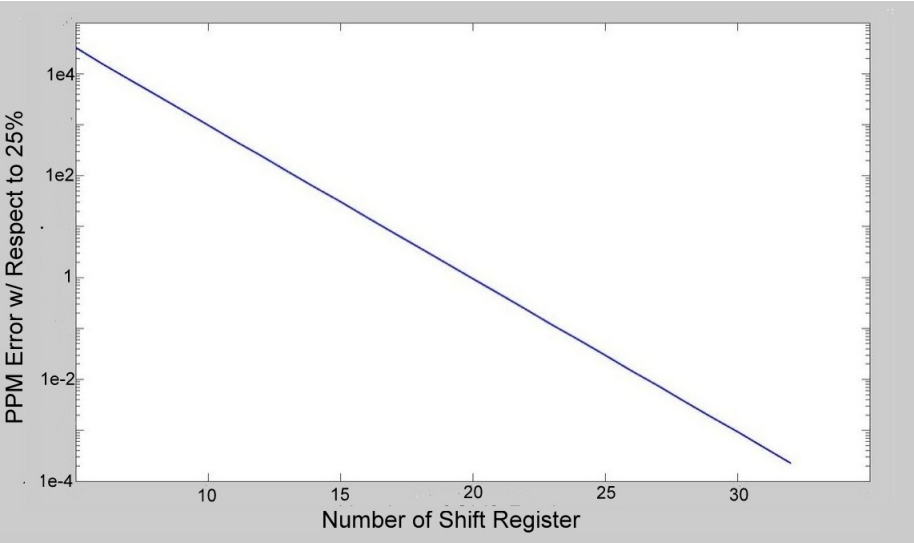


Figure 2-7: PPM error w/ respect to 25% against N bit shift register

2.3.5 Local Fluctuation of Transition Density

The closed form transition density from the previous section is for a full cycle of PRBS run. However, becasuse this application is asynchronous, it is not guaranteed to always have that density. Therefore, we need to look at the local transition density fluctuation. Figure 2-8 shows the instantaneous positive transition density of a PRBS13 from 2nd to 5th run. The shape within each run is fixed and the envelope of the fluctuation decreases with increasing number of runs. We can exploit this nature of a given PRBS, and find out how the range of fluctuation varies as a function of N, N being the number of runs.

For a given PRBS, there exists a point when the instantaneous 0 to1 transition density is minimum; let's call that point α_1, denoting the number of bits passed within a PRBS run, and the number of transition β_1. There is also another bit location such that the density is maximum, α_2, and number of transitions up to that point β_2. To make the definition clear, α_1, β_1, α_2, and β_2 are numbers such that

$$\frac{N2^{n-2}+\beta_1}{N(2^n-1)+\alpha_1} = \text{minimum within the last PRBS run} \tag{2.8}$$

$$\frac{N2^{n-2}+\beta_2}{N(2^n-1)+\alpha_2} = \text{maximum within the last PRBS run} \tag{2.9}$$

At the end of each complete run, the transition density is always equal to the characteristic density derived in the previous section. The fluctuation, $\delta(N)$ for the last run after N previous runs is then

$$\delta(N) = \frac{N2^{n-2} + \beta_2}{N(2^n - 1) + \alpha_2} - \frac{N2^{n-2} + \beta_1}{N(2^n - 1) + \alpha_1} \tag{2.10}$$

The interesting approximation happens when $N \to \infty$, and assuming $2^n >> 1$, which reduces the above equation to

$$\begin{aligned}
\delta(N) &\approx \frac{(N2^{n-2} + \beta_2)(N2^n + \alpha_1) - (N2^{n-2} + \beta_1)(N2^n + \alpha_2)}{(N2^n + \alpha_1)(N2^n + \alpha_2)} \\
&\approx \frac{(\beta_2 - \beta_1)N2^n + (\alpha_1 - \alpha_2)N2^{n-2} + \beta_2\alpha_1 - \beta_1\alpha_2}{(N2^n)^2} \\
&\approx \frac{(\beta_2 - \beta_1)N2^n + (\alpha_1 - \alpha_2)/4 \times N2^n}{(N2^n)^2} \\
&= \frac{\beta_2 - \beta_1 + (\alpha_1 - \alpha_2)/4}{N2^n} \\
&= \frac{(\beta_2 - \alpha_2/4) - (\beta_1 - \alpha_1/4)}{2^n}\frac{1}{N} \\
&= \frac{\gamma}{N}
\end{aligned} \tag{2.11}$$

Equation 2.11 shows that the local fluctuation asymptotically becomes inversely proportional to the number of runs N with some proportionality constant γ. It makes intuitive sense that eventually the average fluctuation will reduce to 0 and the transition density will converge to the characteristic density given by the closed form.

The proportionality constant γ is essentially the bit number fluctuation resulting in density fluctuation divided by the total number of bits, (or interpreted as the fluctuation of a single PRBS run) which also makes sense. γ can either be found by looking at the maximum and minimum density bit location or empirically for a given PRBS by running a regression given this fluctuation model. Figure 2-9 shows a linear relation between number of runs and fluctuation range on a log-log plot. After running a regression, the slope of the line is very close to 1 (1.016) and the offset is

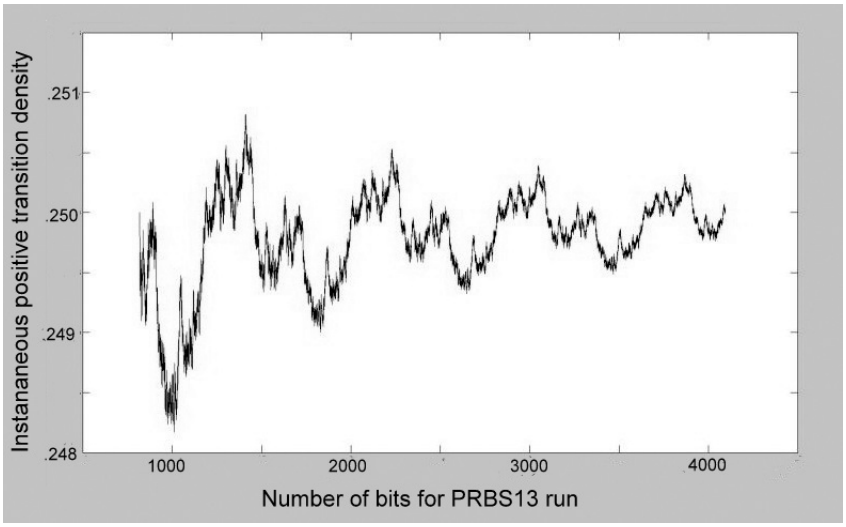


Figure 2-8: Local 0 to 1 transition density fluctuation for run 2 to 5

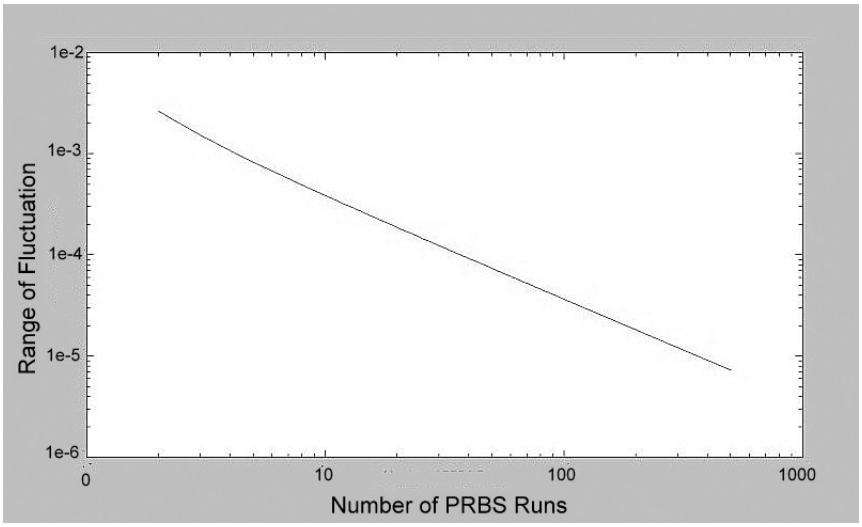


Figure 2-9: PRBS13 fluctuation v.s. Number of runs on Log-Log Plot

-7.97 (log based 2), which makes γ equal to $2^{-7.97} \approx 0.004$. To validate the model given in 2.11, α's and β's are explicitly found. $\alpha_2 = 5915$, $\beta_2 = 1490$, $\alpha_1 = 1919$, $\beta_1 = 461$, plugging into the closed form for γ with $n = 13$, the value comes out to be 0.0037, very close to the regression value.The implication of this finding becomes useful when we determine the amount of resource we need for lambda estimation, which will be discussed in the next section.

2.3.6 Counter Resolution and Divider Length

When doing lambda estimation using PRBS's, the estimation error is two-fold: the deterministic ppm error from transition density and estimation quantization error. However, the transition density is not really fixed due to local fluctuation, but is inversely proportional to number of runs. There is a subtle optimization that we are able to achieve: we have fixed number of bits in counters and can only achieve a certain resolution, then intuitively after a certain point in time the decreasing local fluctuation in transition density can not be resolved by our counters anymore. In other words any extra counting time would be a waste. This relates to the number of division stages needed for counting and the question remains whether there exists a point when any extra stage will not provide any more benefits.

Let's define f_d as the true data frequency, f_s as the sampling frequency. For the purpose of this analysis, let's ignore the deterministic ppm error given by equation 2.7, and we will only look at the local fluctuation error. For an m bit counter, the resolution it can achieve is then $1/2^m$. This quantization error can be translated to an effective ppm error,ϵ given by

$$\begin{aligned} \frac{1}{2^m} &= \frac{\epsilon f_d}{f_s} \\ \epsilon &= \frac{f_s}{f_d 2^m} \end{aligned} \tag{2.12}$$

ϵ is the minimum error an m bit counter can ever achieve due to its limited resolution. Also, given that there are D division stages, the total counting time is

$$T_{count} = \frac{2^m}{f_s/2^D} \tag{2.13}$$

Then the number of runs, N, the l-bit PRBS has gone through is given by

$$\begin{aligned} N &= T_{count} \times \frac{f_d}{2^l - 1} \\ &\approx 2^{m+D-l} \frac{f_d}{f_s} \end{aligned} \tag{2.14}$$

From the previous section, we found out the local fluctuation of transition density is bounded by γ/N. Now we want to find the D such that after certain N, the error is not dominated by local fluctuation but the counter resolution. So we set the two bounds equal, and obtain

$$\begin{aligned} \frac{\gamma}{N} &= 2 \times 0.25 \times \epsilon \\ \frac{\gamma f_s}{2^{m+D-l} f_d} &= \frac{f_s}{2^{m+1} f_d} \\ 2^D &= \gamma 2^{l+1} \\ D &= \log_2 \gamma 2^{l+1} \end{aligned} \tag{2.15}$$

We also know what γ is from the previous derivations, namely $\frac{(\beta_2 - \alpha_2/4) - (\beta_1 - \alpha_1/4)}{2^l}$. Plugging in this into 2.15 we have

$$D = \log_2 2 \left((\beta_2 - \frac{\alpha_2}{4}) - (\beta_1 - \frac{\alpha_1}{4}) \right) \tag{2.16}$$

This is a very interesting result since there is no explicit dependency on m or$l$. However, α and β do have some dependency on l, but they have small effect on D due to the log function. Therefore D stays relatively constant and this is the maximum D we need for fluctuation to disappear for any given m bit counter. Plugging in the number for PRBS13, we approximate D to be about 6. This means that for any number of division stages larger than 6, we should expect a flat line for estimation error, which is solely given by quantization error. Figure 2-10 shows a 2-D plot of lambda estimation error (with deterministic error excluded) when PRBS13 is used. For increasing number of bit in counters, we see a monotonically decrease in error. In the number of division stages direction, we see large fluctuations for low division numbers and it starts to become flat at around D=6, which agrees with what we find analytically. Therefore, we choose the number of bits in counter based on the error we can tolerate, and we choose number of division stages based on local fluctuation attenuation. Leaving some margins, 10 bit counters and 7 division stages are used for lambda estimator. This guarantees $< 10^{-3}$ error in quantization error.

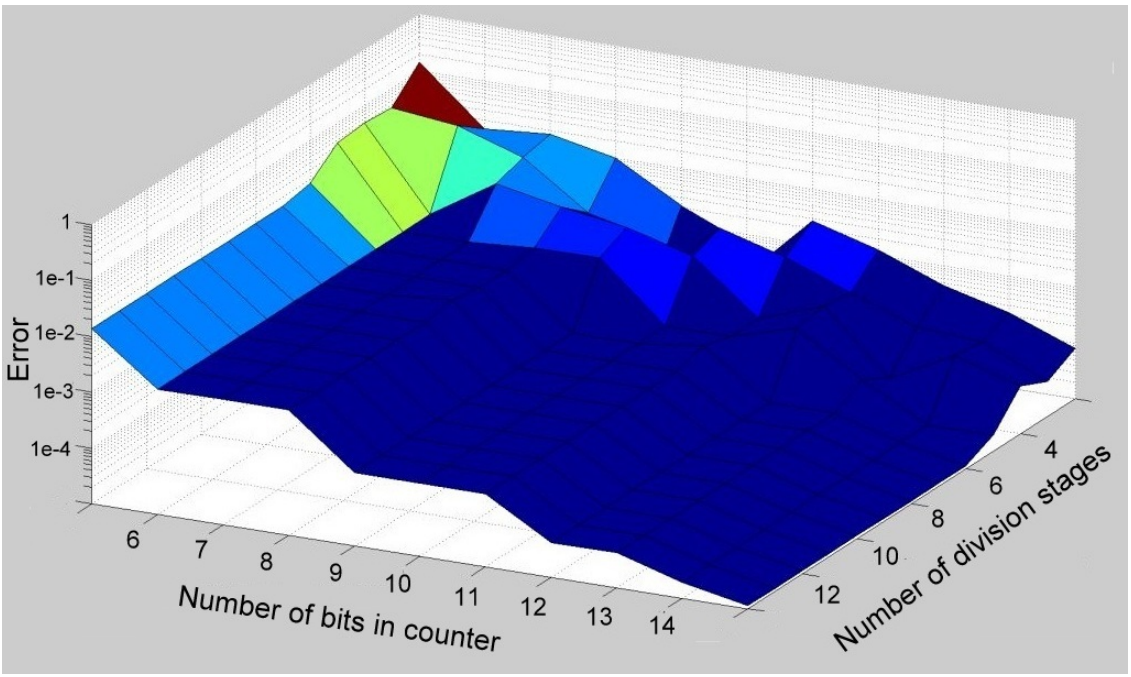


Figure 2-10: 2D lambda estimation error plot for PRBS13

This analysis results in a reasonable precision in estimation using the minimum amount of resources. Empirically, D increases a bit with respect to increasing l, but this effect is masked by the deterministic ppm error. Moderate fluctuation near a number that's closer to 25% should yield the same or even better final result than small fluctuation near a number away from 25%. The conclusion is that this fixed number design should suffice for any PRBS test sequences.

2.4 Reconstruction Method

As shown above, the lambda estimator can provide a good estimation of λ_b, however the error could still be on the order of 10^{-3}, which may be larger than needed. When the error is large, we will have no eye opening when we calculate the phase location of each point with the simple reconstruction method

$$\tau_n = mod(\tau_{n-1} + \lambda_b, 1) = mod(n \times \lambda_b, 1) \tag{2.17}$$

Therefore, the first phase of the reconstruction algorithm would be to do a crude search within the error range until we have a reasonable eye opening at the output.

1. For trial n, find time location for each sample for first 1024 sample group
2. Discretize these time locations into M bins, where M can be adjusted for precision (for first iteration, M=32 is chosen).
3. For each bin, find the minimum value above 0 and maximum value below 0, which then tells the opening of that bin
4. Pass the opening values through a circular running average filter.
5. Find the maximum value of the filter, that would be the eye opening.
6. If eye opening is larger than threshold, move onto second phase. If not, add or subtract ceiling(n/2)/1024 to the estimated lambda depending on the trial number. Then iterate to 1.

When finished, the error would be small enough for us to move onto the next phase of the reconstruction.The eye opening threshold level for the first phase estimation would be an adjustable number to achieve the desired result. Let's call the coarsely estimated lambda from the first phase $\hat{\lambda_b}$, and the true lambda λ_b, then they can be related as

$$\lambda_b = \hat{\lambda_b} + e_b \tag{2.18}$$

where e_b is the error between the two. When we use the simple reconstruction method, we see that the error term would also accumulate inside as $mod(n \times e_b, 1)$. The precision we obtained from the lambda estimator would guarantee that for 1024 points, the accumulated error would be less than a whole period, ie. $1024 \times e_b < 1$. However, the next 1024 points would start with the accumulated error, which means the eye diagram with respect to the first group of samples is shifted according to e_b. With this insight, we can analyze how much each small eye diagram has shifted with respect to the previous one and use that as a correction term for the estimated lambda. Let's

have c_l be the center of the eye opening for the lth data sample group, then ideally we will have

$$n \times \hat{e_b} = c_{l+1} - c_l \tag{2.19}$$

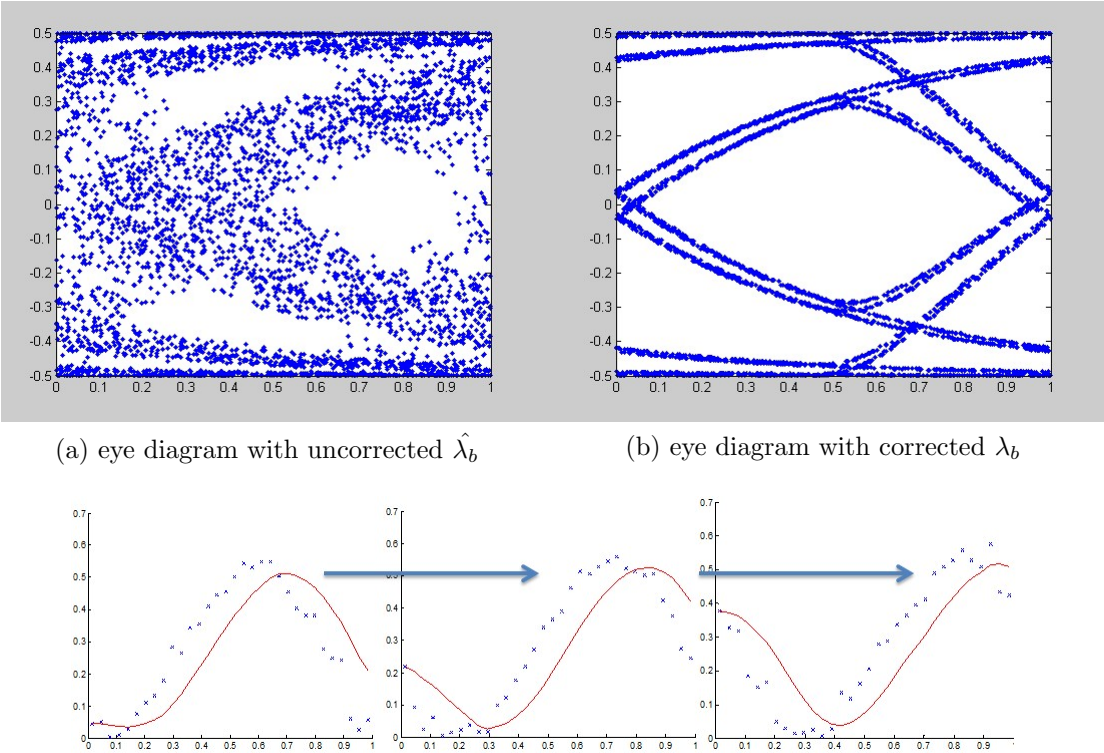
$$e_b = \frac{c_{l+1} - c_l}{n} \tag{2.20}$$

However, finding the exact center of the eye with only 1024 points is challenging, thus another estimation for the center of the eye is used as well. To formulate this, we have the following algorithm for correction:

1. Find time location for each sample for first 1024 sample group
2. Discretize these time locations into M bins, where M can be adjusted for precision (for first iteration, M=32 is chosen).
3. For each bin, find the minimum value above 0 and maximum value below 0, which then tells the opening of that bin
4. Pass the opening values through a circular running average filter.
5. Repeat step 1 to 4 for the second sample group.
6. The eye opening center could wrap around to other side of the period. Correct lambda in one direction first by subtracting two eye centers and divide by 1024.
7. If correction direction is wrong, it will cause the eye to close again. If so, try other direction.
8. Iterate back to one with the new lambda and higher precision M.

Figure 2-11 shows the iterative correction algorithm at work. Figure 2-11a is the resulting eye diagram with uncorrected $\hat{\lambda_b}$, in which case we can see a great deal of dispersion and the eye opening is nowhere close to the simulated eye. After the eye opening shift analysis in Figure 2-11c, we see the sub-eyes are moving constantly to the right, which is the result of the error in the lambda estimate. The blue crosses

are the eye opening for each discrete bin, and the red curves are the output after the circular running average filter.



(a) eye diagram with uncorrected $\hat{\lambda_b}$

(b) eye diagram with corrected λ_b

(c) eye opening plot for three consecutive sample groups

Figure 2-11: Iteratively corrected lambda algorithm at work

Using this information, we regenerate the eye diagram with the corrected lambda, which is shown in Figure 2-11b. We see that the result resembles what the ideal simulated eye diagram almost perfectly. After the corrections for lambda, the newest time locations for each point are calculated and finally stored in the next 3K locations. The precision is going to be downscaled to only 8 bit decimal numbers (represented by the memory byte), which means the final output to the computer is discretized to 256 by 256 grids both for horizontal and vertical axis.

Chapter 3

Circuit Implementation

This chapter reviews implementation details of several important building blocks in the system, including high speed sample and hold, analysis on bounded time sampling clock jitter, lambda estimator and the digital reconstruction block. On the analog side, the sample and hold circuit is designed and simulated on transistor level across process, voltage and temperature (PVT) corners with layout parasitics. Mathematical model is studied and presented on sampling clock jitter and a specification on the required RMS jitter is determined. A novel CML latch circuit is designed particularly for this application to be used in subrate extractor. On the digital side, the lambda estimator and digital reconstruction blocks are implemented in Verilog RTL code and synthesized to obtain area estimates.

3.1 Sample and Hold

As mentioned in the previous chapter, the sample and hold circuit is one of the hardest building blocks in this system. The system is built upon undersampling techniques; even though precision and linearity might not be a top priority, high bandwidth spec is the most difficult. The targeted specification for bandwidth is 10GHz, the Nyquist frequency of the maximum data rate. For reasons already explained, the proposed circuit is pass gate switch based. In order to make the S&H circuit as general as possible, no prior source impedance or output ADC capacitor load is assumed. As a

result, there needs to be input and output buffer amplifiers to fix the source impedance and output loads seen by the switches. The issue of direct C_{ds} is solved by cross coupling in differential configuration. Following subsections will discuss the design in more details.

3.1.1 Pass Gate Self Bandwidth

In order to see whether the 10GHz input bandwidth spec is feasible, the pass gate's intrinsic bandwidth, given by its on resistance and parasitic drain capacitance ($R_{ON}C_d$), must be sufficiently larger. Even though charge injection is less of an issue in differential configuration, by appropriately sizing the NMOS and PMOS, one can reduce the clock feedthrough while improving linearity. In a case when the input DC level is close to power supply, PMOS will turn on strongly and doing most of the work. The test circuit contains a passgate with NMOS and PMOS width ratio to be 2:3. Unit width is swept from 0.1um to 2.1um. Figure 3-1 shows the intrinsic bandwidth of passgate with respect to width.

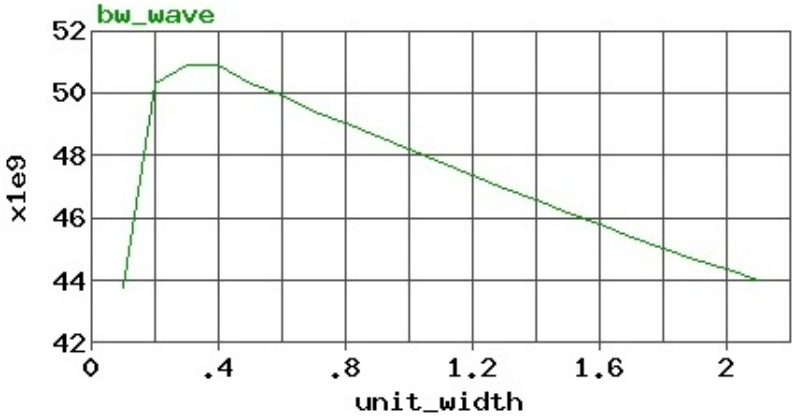


Figure 3-1: Passgate self bandwidth

Even though there exists an optimum bandwidth point at around 0.4um unit width, the final circuit's optimum point might move due to parasitic capacitance from the next stage and/or source resistance from the previous stage. However, the main takeaway is that the self bandwidth of simple passgate shows to be $> 40GHz$, which is way above the required spec. This means that the limiting factor in the

input signal bandwidth will come from the RC time constants in amplifiers.

3.1.2 S&H Implementation

Figure 3-2 shows the top level sample and hold circuit schematic. The signal path consists of a linear input amplifier that has equalization to push out input bandwidth and sets a low source impedance seen by the pass gates, the main pass gate switches, and a linear output amp that will drive the sampling ADC.

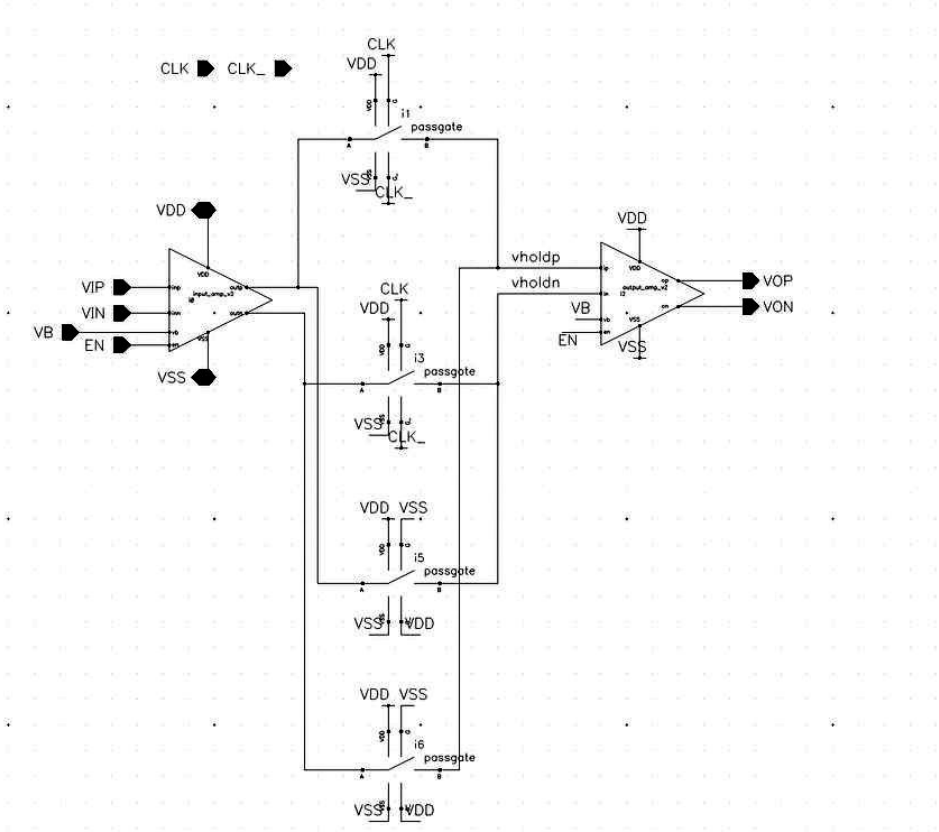


Figure 3-2: Schematic of sample and hold

Figure 3-3 and 3-4 show the schematics of the input and output amplifiers. The input amplifier is a source degenerated amplifier with high tail current to use smaller load resistor, thus higher speed. The capacitor across the degeneration resistor is to provide peaking in frequency response at high frequency. The small signal model of this amplifier is shown in figure 3-5, where r_e is half of the degeneration resistor, c_e is twice the degeneration capacitor, and R_l is the effective load resistance.

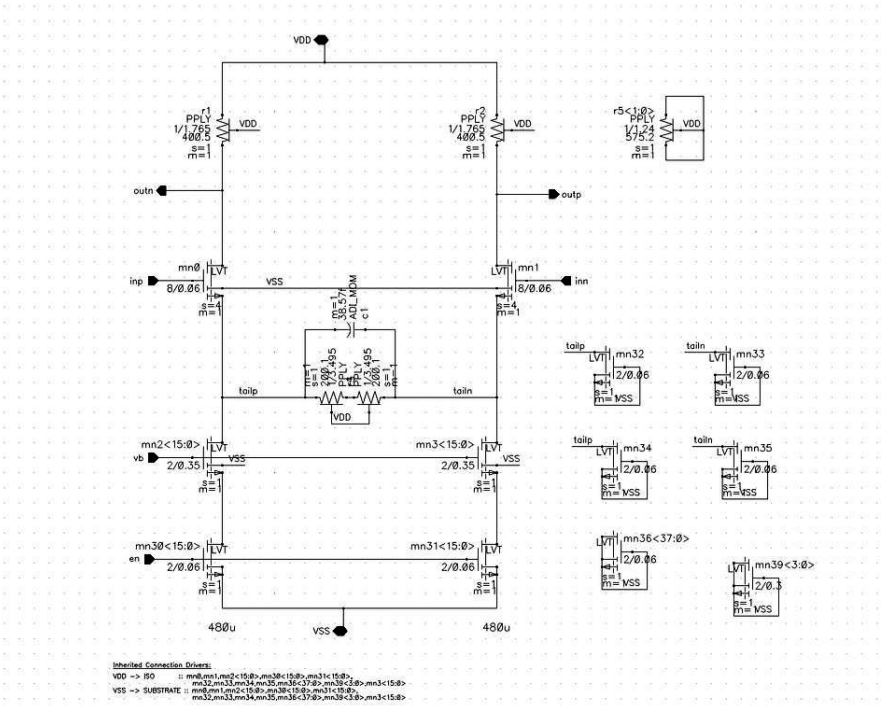


Figure 3-3: Schematic of input amplifier

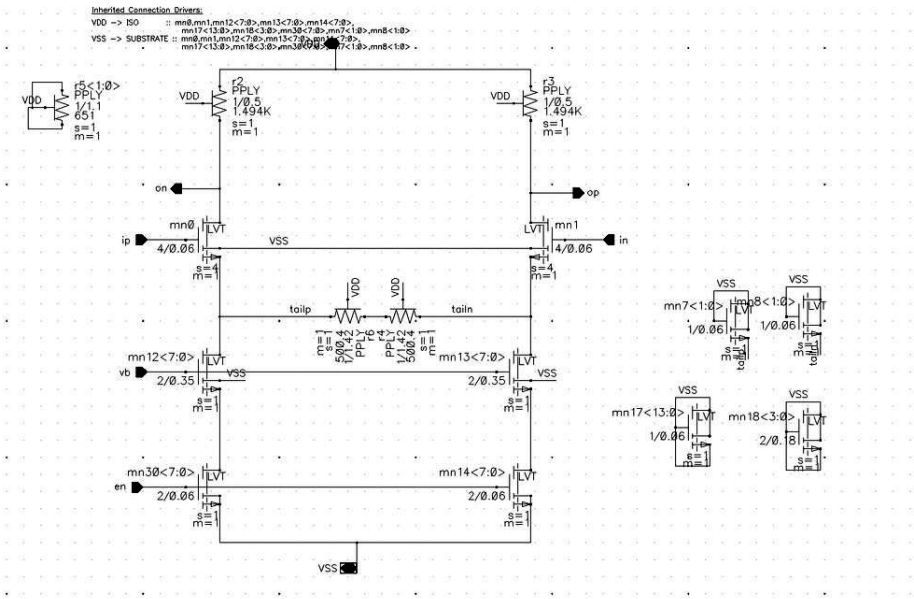


Figure 3-4: Schematic of output amplifier

If we make the approximation that C_{gs} and C_{gd} are much smaller than c_e, then

the gain transfer function can be approximated by

$$\frac{v_{out}}{v_{in}} \approx \frac{-g_m R_l}{1+g_m r_e} \frac{1+sr_e c_e}{1+sr_e c_e/(1+g_m r_e)} \tag{3.1}$$

The degeneration give a zero at $1/(r_e c_e)$ and a pole at $(1+g_m r_e)/(r_e c_e)$, which provides a bump in the gain curve with the zero occurring before the pole. The effect of C_{gs} and C_{gd} will be adding high frequency parasitic poles to eventually load down the gain and move the zero location a bit. By adjusting the degeneration capacitor, desired gain peaking to push out bandwidth and compensate for later stage poles can be achieved. The amplifier has a load resistance of 400Ω. Foot switches are added for enable function in this amplifier. The output amplifier is a simple source degenerated amplifier with lower current for much lower speed (driving ADC at 200MHz). Figure 3-6 shows the sizing the passgates used in this design. Parametric sweep on width shows the optimum bandwidth happens at these sizes.

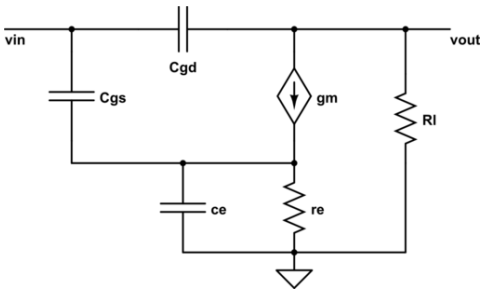


Figure 3-5: Small signal model of equalizing input amplifier

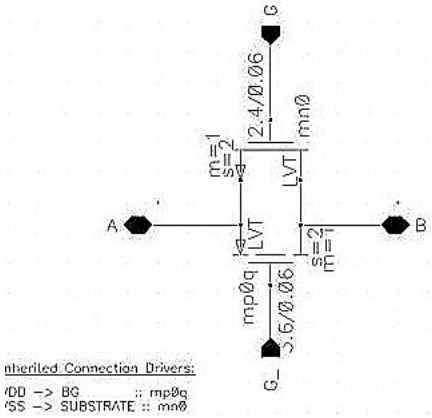
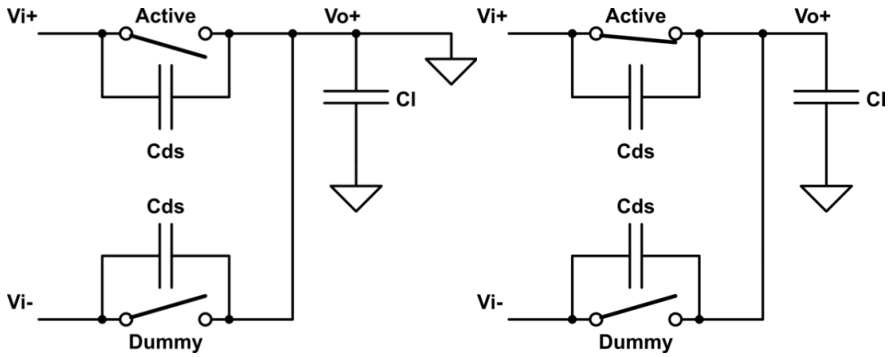


Figure 3-6: Schematic of pass gate

In order to obtain the maximum bandwidth possible, no explicit hold capacitors are used. The parasitic drain capacitance of the switches are used as the hold capacitors, and consequently the effect of C_{ds} coupling is not negligible. The cross coupled dummy switches serve to cancel the direct C_{ds} coupling while holding the sampled value. The trade off is a small bandwidth decrease due to Miller Effect. Figure 3-7 show simple models of the cross coupled switch network during off and on modes. While off, the fully differential configuration enforces that V_o^+ becomes incremental



(a) Off (b) On

Figure 3-7: Simple model of S&H

ground because equal amount of charge injected on one input side will be sinked by the other side. In this case, each V_i terminal will see twice C_{ds} since each will drive two off switches. While on, we can approximate V_o^+ to be V_i^+, which makes the C_{ds} seen by V_i^- double due to Miller Effect. Equivalently, in track mode each input terminal will also see $2C_{ds}$. Using a replica input amplifier to drive the dummy switches will not eliminate this Miller multiplication, but only reduces the source parasitic loading on the main input amp. As a result, considering power and area trade-off, a replica input amp is not used. The finished sample and hold circuit consumes 1.5mW at 1.2 nominal power supply. Figure 3-8 shows the layout of the circuit, which is about $15\mu m$ by $30\mu m$.

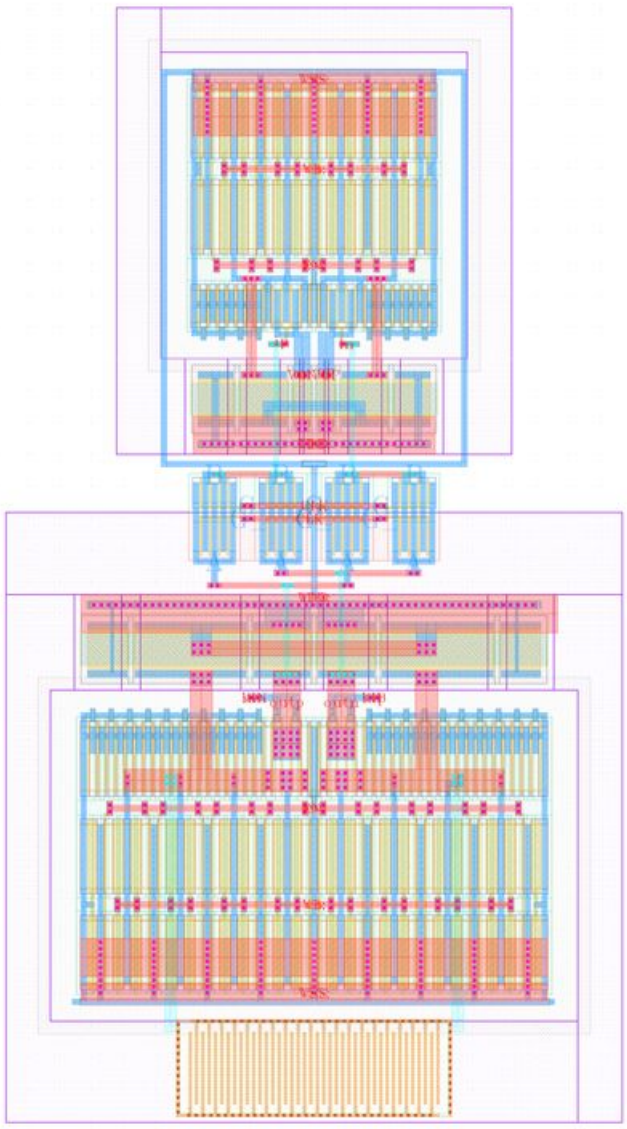


Figure 3-8: Layout of sample and hold

3.1.3 Results

Figures 3-9, 3-10 and 3-11 show Monte Carlo simulation results for varying temperature, voltage supply and process corners. Bandwidth, peaking and output gain are plotted. The worst case scenario is for low power supply, high temperature and slow corner, but bandwidth still maintains greater than 10GHz. Peaking is controlled under .12db across all conditions, which is still considered flat in gain transfer function.

Output gain is very strong function of temperature and process corner. From the simulation, it is really a gain bandwidth trade-off, i.e. in slow corner the output gain is closer to unity but bandwidth is smaller, and in fast corner bandwidth is higher for a lower output gain. Output gain also decreases almost linearly with temperature. However, gain error could be compensated for anywhere later in the system, either by the voltage reference on ADC or even in the digital domain. For purpose of the rest of the thesis, we will assume that the gain error is calibrated and the final sampled point is on the full scale of our interest, i.e. 0-255 in digital values.

The hold mode cross coupled gain is around -53db on average across all PVT conditions. Mismatch in layout is most likely the issue for the non complete cancellation. More investigation could be put in to optimize layout in that respect; however -53db gain during hold mode is enough to achieve 8 bit resolution with some margin for noise.

A post layout transient simulation is also run, and a reconstruction is performed in Matlab. Figure 3-12 shows the transient plot of the input, voltage after switches and final sample and hold output respectively. The sampling clock has relatively sharp edges to model finite switch turn on and turn off time. Hold values are sampled and Matlab reconstruction code are run to obtain a reconstructed eye. Figure 3-13 shows the real channel eye on the left and reconstructed eye from sample and hold circuit on the right. The limited bandwidth of the S&H circuit causes reconstructed eye to close a bit compared to the actual channel eye. This is to be expected since we argue that this circuit behaves as a limited bandwidth signal probe. The reconstructed eye does show acceptable result qualitatively in terms of showing estimates of eye opening and jitter performance.

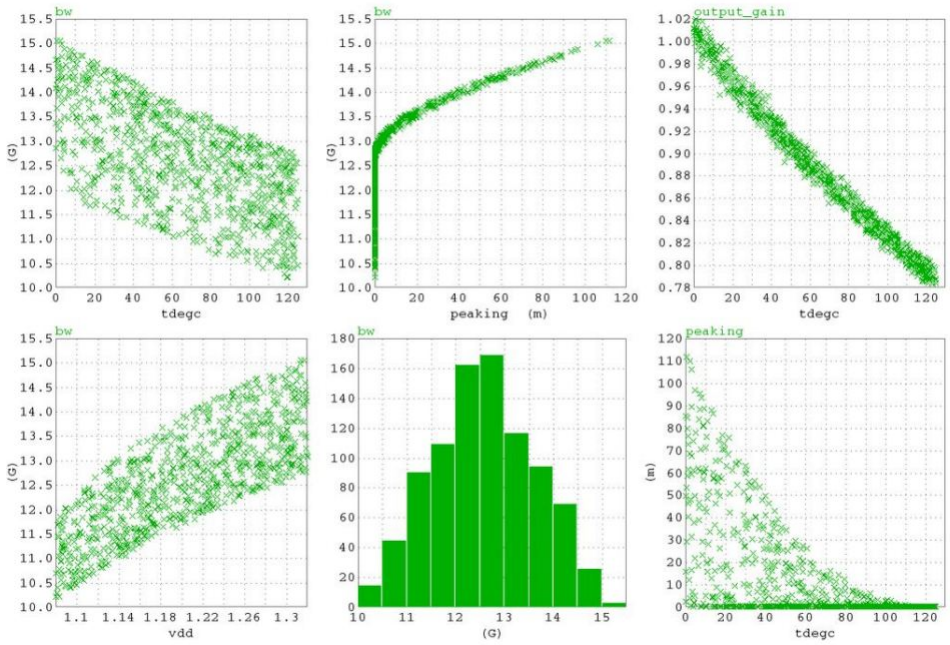


Figure 3-9: Monte Carlo plots of sample and hold in slow corner

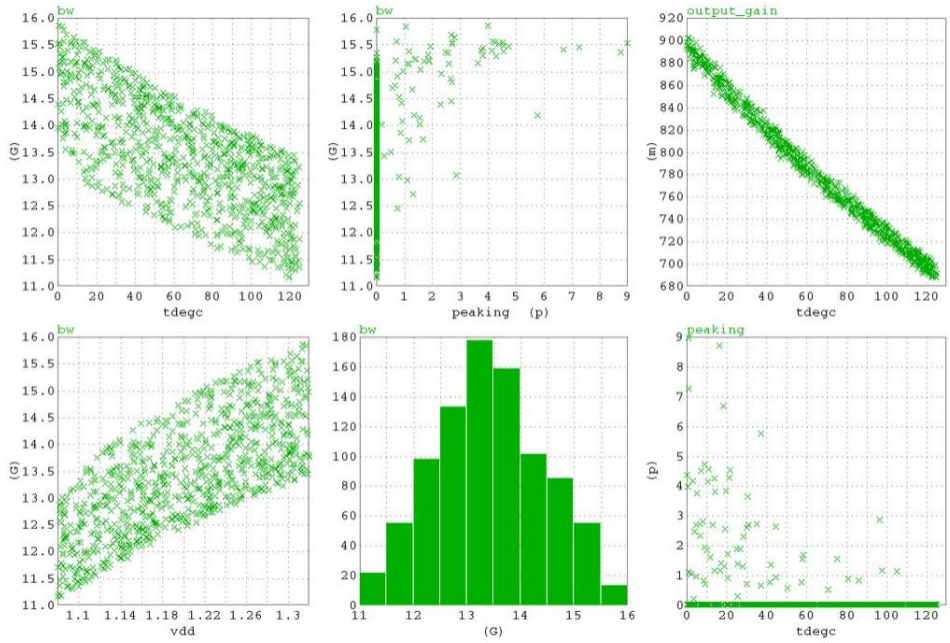


Figure 3-10: Monte Carlo plots of sample and hold in nominal corner

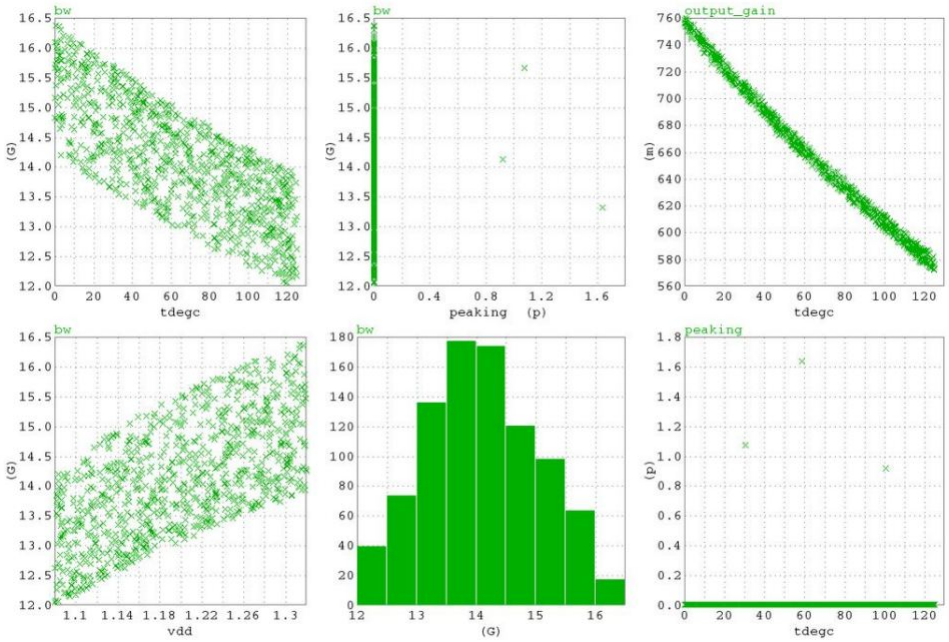


Figure 3-11: Monte Carlo plots of sample and hold in fast corner

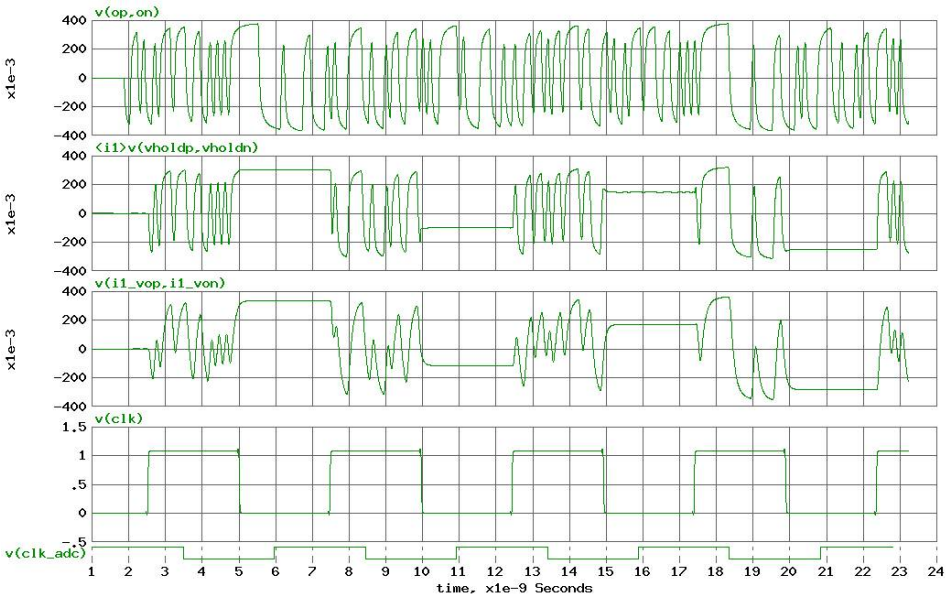


Figure 3-12: Transient waveforms of internal nodes in sample and hold

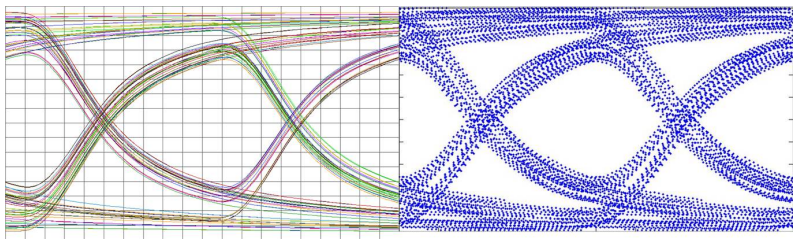


Figure 3-13: Actual channel eye v.s. reconstructed eye from S&H samples

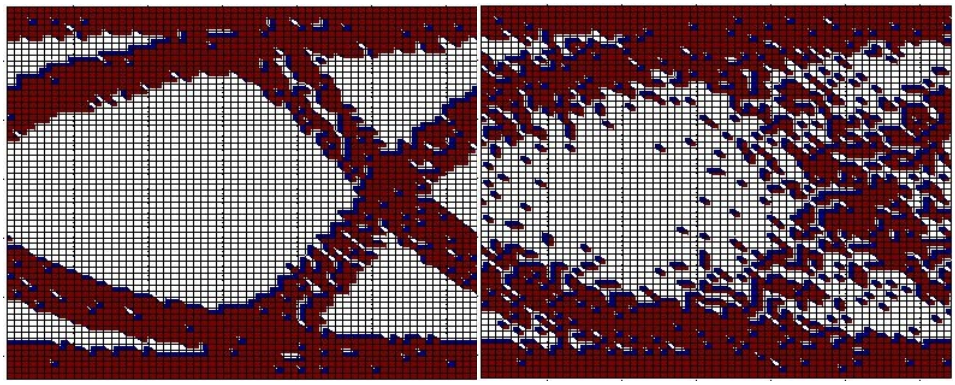
3.2 Sampling Clock Jitter

Sample and hold circuit determines the vertical resolution one has when reconstructing the eye diagram, the other aspect we have to consider then is the time domain resolution. Sampling clock jitter determines the effective number of time steps we can have within a bit period. The worst case is again when the bit period is the shortest (highest data rate), in which clock jitter would become a larger percentage of a bit period. It is difficult to evaluate the effect of clock jitter on the reconstructed eye diagram unless a measure of reconstruction quality is created.

3.2.1 Eye Match Rate

Treating the eye diagrams as images with limited amount of pixels help one create a measure for reconstruction quality. If the eye diagrams are seen as 64 by 64 2-D images, then each grid will either have or not have points in it. Instead looking at the specific number of points within each grid, we should lump it into a binary bit, i.e. we should only care about whether there are points inside a particular grid. This allows us to create a "simplified" eye diagram that still contains enough information in it for comparison. Figure 3-14a shows the lumped eye diagram for a bit period in a 64 by 64 mesh when there is no sampling clock jitter. Figure 3-14b on the other hand shows a lumped eye diagram when clock's rms jitter is 10ps. We easily see the 10ps jitter eye diagram becomes very smudged, and points fall in places that are far from where they should be.

By finding the absolute euclidean distance of the two images, we obtain the number of mismatched grids between the two images. Dividing by the total number of grids we then have a metric that can be interpreted as the mismatch rate of the two images. In this case a mismatch of 100% means an image that's inverted from the original. Assuming about half of the image is occupied on average, a random image (with points scattered around randomly) will yield a mismatch rate of close to 50%.



(a) No jitter (b) 10ps RMS jitter

Figure 3-14: Comparison of reconstructed eye w/ and w/o jitter

3.2.2 Sample Clock Jitter Effect on Reconstruction

Using the metric developed in the previous subsection, transient simulations with clock jitter varying from 0 to 15ps are run and eye diagrams are reconstructed with the sampled points. In this simulation since we want to look at effects of clock jitter alone, the reconstruction algorithm is NOT run, but the accurate lambda is explicitly given and points are repositioned accordingly. In some of the cases (very large jitter), the reconstruction method failed to reconstruct since the eye is essentially closed by several points.

Figure 3-15 shows the evolution of eye diagram with increasing RMS jitter. The reconstructed eye still has reasonable quality up to 5ps jitter. When jitter goes up to 10ps, which is about 10th of the highest data rate bit period, the eye started to get smudged. In figure 3-16, the match rate (1-mismatch rate) is plotted against RMS jitter used in sampling. The numbers are normalized with respect to the no jitter eye diagram, i.e. consequent eyes are compared to the no jitter eye diagram as reference. The match rate almost decreases linearly with increasing jitter. Match rate gets to as low as 70% when 15ps jitter is used. As a conclusion, I use this result to define the jitter specification for sampling clock. From this plot as well as the visual eye diagram outputs, a minimum of 85% match rate is desired, so the RMS jitter performance should be no larger than 4ps in clock performance.

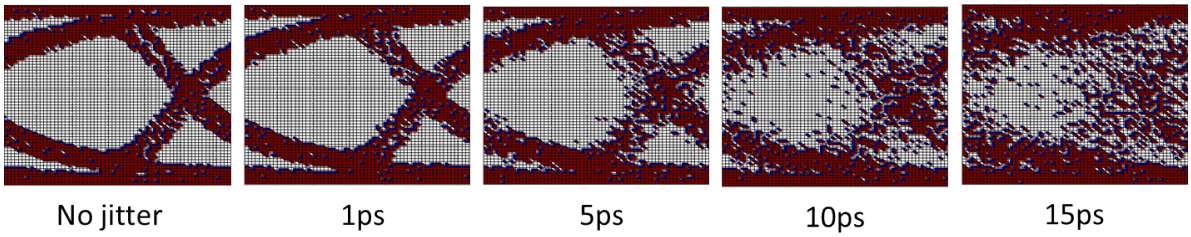


Figure 3-15: Eye diagram evolution with respect to increasing jitter

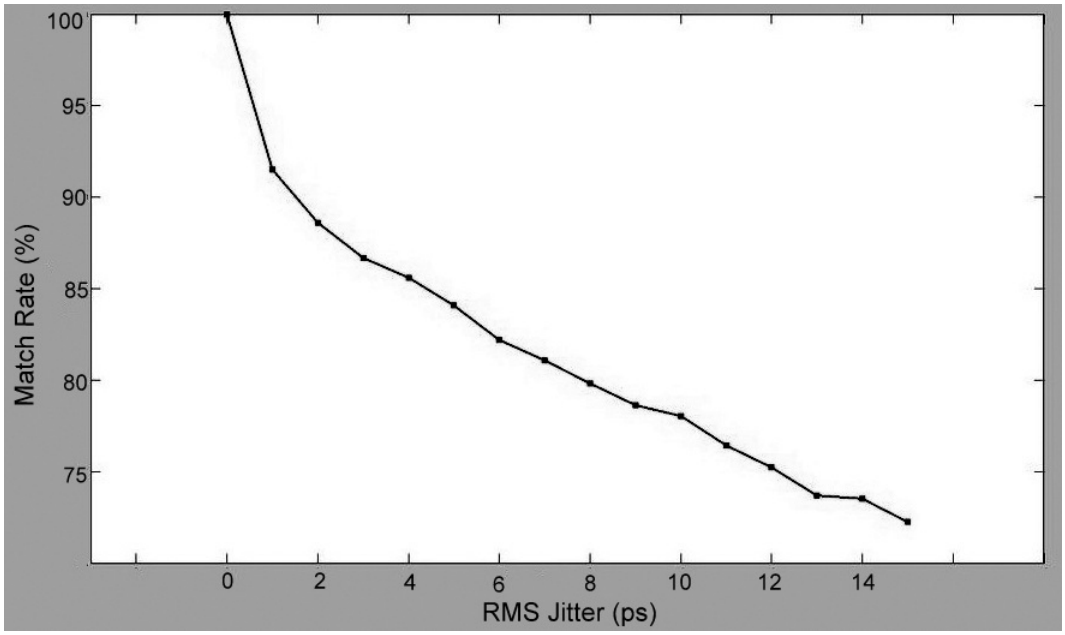


Figure 3-16: Match rate v.s. RMS jitter

3.2.3 Oscillator Jitter Analysis

The nature of the application only requires bounded time sampling, i.e. the sampling clock is only used for finite amount of time. This means phase noise from low frequency components won't manifest themselves as strongly, resulting in better RMS jitter performance. The implication is then simple oscillator such as ring oscillator might be able to meet the specification of $< 4ps$ RMS jitter in this scenario, which would be a big area and power saving. In this subsections the jitter model will be studied and applied to conclude despite bounded time sampling, simple oscillators jitter isn't small enough to be used as sampling clock in this case.

Jitter is the deviation of the zero crossing point away from its desired location for a given period. Jitter arises from noise sources affecting the voltage waveforms par-

ticularly around the zero crossing points in an oscillator, thus advancing or delaying the transition time. In the frequency domain jitter translates to a power leakage to the side bands of the oscillation frequency. The resultant phase noise can be plotted with respect to frequency offset from the center frequency. For a free running oscillator, the jitter is theoretically unbounded as time approaches infinity given a starting point. Figure 3-17 [7] shows the increasing jitter with time due to accumulation of jitter from period to period.

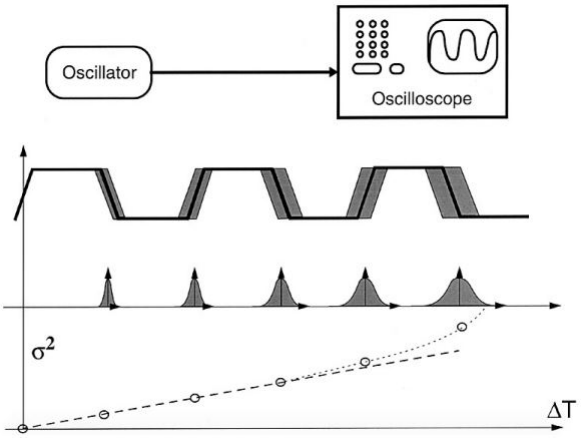


Figure 3-17: Unbounded jitter free running oscillator with respect to time

Since we only use the oscillator for a very short amount of time for sampling, so the jitter is actually bounded. However, its time increasing nature does not change. As a result, an analytical way of finding time domain RMS jitter from phase noise must be derived in order to determine whether a free running oscillator will meet the specification.

Let's define the oscillator's voltage output as a function of time, such that

$$V_{out}(t) = A(t) \cdot f(\omega_0 t + \phi(t)) \tag{3.2}$$

where $A(t)$ is the amplitude fluctuation due to noise and $\phi(t)$ is the phase fluctuation. We are interested in the phase term and how to convert its fluctuation to time for a given period. Assuming $\phi(t)$ is a stationary random process (caused mostly by white noise sources), which means the mean and standard deviation stays the same at any

given point in time, then the analysis becomes quite straight forward. The RMS jitter after N periods is derived in the following way:
Let t_1 be our origin of time, and it is the first zero crossing of V_{out} such that

$$\begin{aligned} V_{out}(t_1) &= 0 \\ \omega_0 t_1 + \phi(t_1) &= 0 \end{aligned} \tag{3.3}$$

Let t_2 be the N_{th} zero crossing (so N_{th} period) and t_2 must satisfy

$$\omega_0 t_2 + \phi(t_2) = 2\pi N \tag{3.4}$$

The accumulated jitter after N periods (deviation of N_{th} zero crossing away from ideal crossing location) is then

$$\Delta t = t_2 - t_1 - NT_0 \tag{3.5}$$

where T_0 is the ideal period, same as $1/f_0$ or $2\pi/\omega_0$. From 3.3 and 3.4, we obtain

$$\begin{aligned} \omega_0(t_2 - t_1) + \phi(t_2) - \phi(t_1) &= 2\pi N \\ \omega_0(\Delta t + NT_0) &= 2\pi N + (\phi(t_1) - \phi(t_2)) \\ \Delta t &= \frac{2\pi N}{\omega_0} + \frac{\phi(t_1) - \phi(t_2)}{\omega_0} - NT_0 \\ \Delta t &= \frac{\phi(t_1) - \phi(t_2)}{\omega_0} \end{aligned} \tag{3.6}$$

Δt is now expressed as a function of stationary random process $\phi(t)$, and its mean and variance will be expressed as following

$$E[\Delta t] = \frac{1}{\omega_0}(E[\phi(t_1)] - E[\phi(t_2)]) = 0 \tag{3.7}$$

$$Var[\Delta t] = \frac{1}{\omega_0^2}(E[\phi^2(t_1)] + E[\phi^2(t_2)] - 2E[\phi(t_1)\phi(t_2)]) \tag{3.8}$$

Our interest here is the variance of Δt, which we express as Δt_{rms}^2, and since $\phi(t)$ is

stationary, we simplify 3.8 as

$$\Delta t_{rms}^2 = \frac{2}{\omega_0^2}(E[\phi^2(t)] - E[\phi(t_1)\phi(t_2)]) \tag{3.9}$$

The term $E[\phi^2(t)]$ is simply the variance of phase noise given that the mean is 0. The later term $E[\phi(t_1)\phi(t_2)]$ is then the covariance of the two random processes given a time offset. Since $\phi(t_1)$ is stationary, this term is the same as $E[\phi(0)\phi(t_2 - t_1)] \approx E[\phi(0)\phi(NT_0)]$. Both of these terms can be expressed with $\phi(t)$'s auto-covariance function $R(\tau)$, where τ is the time offset. Therefore

$$\Delta t_{rms}^2 = \frac{2}{\omega_0^2}(R(0) - R(NT_0)) \tag{3.10}$$

Next, we explore the relationship between the auto-covariance function and the phase noise spectrum. The phase noise spectrum (more precisely it's the phase noise power spectral density) is namely the Fourier Transform of the random process's auto-covariance function. The relationship is then given by the following

$$S_\phi(f) = \int_{-\infty}^{\infty} R(\tau)e^{-j2\pi f\tau}d\tau \tag{3.11}$$

$$R(\tau) = \int_{-\infty}^{\infty} S_\phi(\omega)e^{j2\pi f\tau}df \tag{3.12}$$

Since both $R(\tau)$ and $S_\phi(\omega)$ are even functions, these terms can be reduced further

$$\begin{aligned} R(\tau) &= \int_{-\infty}^{\infty} S_\phi(f)e^{j2\pi f\tau}df \\ &= \int_{-\infty}^{\infty} S_\phi(f)(\cos(2\pi f\tau) + j\sin(2\pi f\tau))df \\ &= \int_{-\infty}^{\infty} S_\phi(f)\cos(\omega\tau)df + j\int_{-\infty}^{\infty} S_\phi(f)\sin(2\pi f\tau)df \end{aligned} \tag{3.13}$$

The second integral will become zero since it's an even function multiplying an odd

function. The transform then is reduced to Cosine Transform

$$\begin{aligned} R(\tau) &= \int_{-\infty}^{\infty} S_\phi(f)\cos(2\pi f\tau)df \\ &= 2\int_{0}^{\infty} S_\phi(f)\cos(2\pi f\tau)df \end{aligned} \tag{3.14}$$

We recognize that $R(0)$ is then just the area under S_ϕ,

$$R(0) = 2\int_{0}^{\infty} S_\phi(f)df \tag{3.15}$$

Now equation 3.10 can be expressed in terms of phase noise PSD,

$$\begin{aligned} \Delta t_{rms}^2 &= \frac{2}{\omega_0^2}(2\int_{0}^{\infty} S_\phi(f)df - 2\int_{0}^{\infty} S_\phi(f)cos(2\pi fNT_0)df) \\ &= \frac{4}{\omega_0^2}\int_{0}^{\infty} S_\phi(f)(1-\cos(2\pi fNT_0)df \qquad (3.16) \\ &= \frac{4}{\omega_0^2}\int_{0}^{\infty} S_\phi(f)\times 2\sin^2(\frac{2\pi f}{2}NT_0)df \\ \Delta t_{rms}^2 &= \frac{8}{\omega_0^2}\int_{0}^{\infty} S_\phi(f)\sin^2(\pi NT_0 f)df \qquad (3.17) \end{aligned}$$

This expression is very interesting and might not seem intuitive first with the $\sin^2$ term. This function acts as a weighting function depending on the observation period. One intuitive way to understand this expression is by looking at several special cases. When the observation period is very small, approaching zero, $NT_0 -> 0$ it is obvious that there is no observed jitter, so this expression does give zero RMS jitter as an answer. When the time runs till infinity, $NT_0 -> \infty$, it's easier to see from equation 3.17 that the cosine part the expression will approach 0 after integration. The proof

is the following by integration by parts

$$\begin{aligned}\Delta t_{rms}^2 &= \frac{4}{\omega_0^2}\int_0^\infty S_\phi(f)(1-\cos(2\pi f N T_0)df \\ &= \frac{4}{\omega_0^2}\int_0^\infty S_\phi(f)df - \frac{4}{\omega_0^2}\int_0^\infty S_\phi(f)\cos(2\pi f N T_0)df\Big|_{u=S_\phi(f),dv=\cos(2\pi f N T_0)df} \\ &= \frac{4}{\omega_0^2}\int_0^\infty S_\phi(f)df - \left(S_\phi(f)\frac{\sin(2\pi f N T_0)}{2\pi N T_0}\Big|_0^\infty - \int_0^\infty \frac{\sin(2\pi f N T_0)}{2\pi N T_0}\frac{dS_\phi(f)}{df}df\right)\end{aligned}$$

Since there is a $2\pi NT_0$ on the denominator for both of the later terms and the integrals will evaluate to a finite number, as NT_0 approaches ∞, these two terms will go away and only leaving the first term, thus

$$\Delta t_{rms}^2\big|_{NT_0->\infty} = \frac{4}{\omega_0^2}\int_0^\infty S_\phi(f)df \tag{3.18}$$

And this expression is exactly the area under the phase noise power spectrum converted to time, and it means this expression contains the noise contribution from all frequency phase noise, which is what we expect when time goes to infinity. Another way of looking at it is from the auto-covariance point of view. $R(0)$ is the full variance of the noise process, which is what we see when time runs to infinity, and $R(\tau)$ for very large τ should approach 0, which means the two noise samples that are far apart should not be correlated at all. This reasoning will give us the same answer if we take τ to infinity from equation 3.10.

3.2.4 Ring Oscillator Phase Noise

Figure 3-18 shows the phase noise plot of a simple ring oscillator of 218MHz (green) and the generated weighting function (yellow) when the observation time is only one period. This is known as the period to period phase noise of an oscillator. The weighting function acts as a +20dBc/Hz high pass filter with a cutoff at half the oscillation frequency. It is intuitive that such a high pass eliminates low frequency phase noise due to such a short observation time. Similarly, if we look at longer observation time, i.e. 3072 sampling points, then the high pass cutoff frequency shifts

left to include more low frequency phase noise, as shown in figure 3-19.

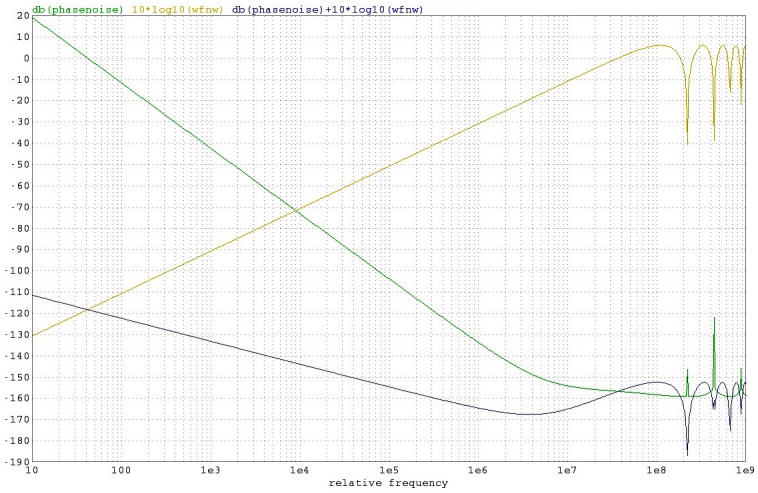


Figure 3-18: Phase noise plot of period to period jitter

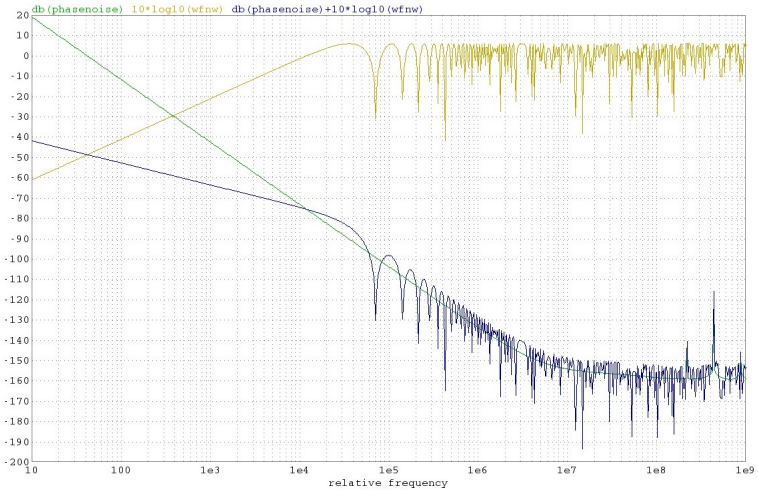


Figure 3-19: Phase noise plot of 3072 point duration

The RMS jitter is then obtained by integrating the weighted phase noise curve (blue). For this simple ring oscillator, the integrated RMS jitter is 60ps, which is larger than the required specification. As a result, free running oscillators in general can't be used as sampling clocks in this application due to relatively long observation period for low frequency noise to play a big role. Another possible solution could be a LC oscillator tank (whose phase noise is much smaller) and divider combination, which is not explored in this work. Otherwise, dedicated clock generators are required

but will be assumed to be given with the specified RMS jitter for later parts of the thesis.

3.3 Lambda Estimation

3.3.1 Frequency Divider for High Speed Data

The front stage of the lambda estimator is the subrate extractor, a chain of frequency dividers to slow down the input signal to a low frequency square wave that the counters can handle. The sampling clock (200MHz) is already considered low frequency for 65nm CMOS devices, so standard CMOS flip flop based frequency divider can be directly used. Figure 3-20 shows the simplified block diagram of a standard divider. Such CMOS D flip flops can be easily found in standard digital cells in the process library.

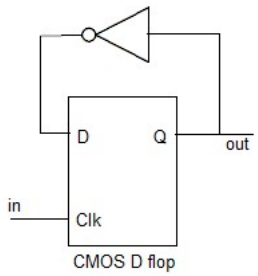


Figure 3-20: CMOS frequency divider (divide by 2)

The challenge lies in the frequency divider for data input, which comes in at 10Gbps, significantly faster than what CMOS logic can handle. As a result, current mode logic (CML) blocks are used as the first stages of data subrate extractor chain. This application exists another particular challenge: the input data might not have a wide open eye, which means a conventional CML frequency divider designed for a relatively large input amplitude might not work in this case. As a result, a new CML frequency divider with better input amplitude sensitivity is desired. The correctness of frequency division will also contribute to the initial estimation error in λ_b. The reconstruction algorithm will search for an eye opening by sweeping λ_b in the first

phase, so this error can also be treated as any other initial error; however, a correct division operation will only prove valuable for later stages.

3.3.2 CML Latch and Frequency Divider

Figure 3-21 shows a schematic of a conventional CML latch, in which the clock signal steers the bias current through either a differential pair (in track mode) or the cross coupled pair (in latch mode). If the clock signal is strong enough, the bias current will only go through one branch of the latch, thus guaranteeing correctness of operation. The major disadvantage of the conventional design is the fixed bias current, which limits the sizes of the cross pair and diff pair devices and they are coupled in design. Besides, clock feed-through might disturb the output during clock transitions if output amplitude is not big enough. The hard turn on and turn off of the cross pair is also an issue that slows down the operation.

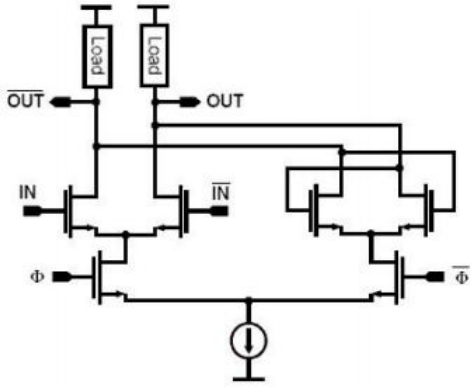


Figure 3-21: Circuit schematic of conventional CML latch

Authors in [8] discuss the disadvantages for the conventional latch and proposed alternative designs in which an always-on cross pair is present and clock steering differential pair brings the current to power supply. This achieves faster switching by a ft doubler when clock is compared to the common mode V_{ref}, shown in figure 3-22. However, during the track mode the swing will be reduced since only the cross coupled pair's tail current goes through the load and in a low power supply process, this is not desirable. However, this isolates the design of differential pair and the latch pair and does work better.

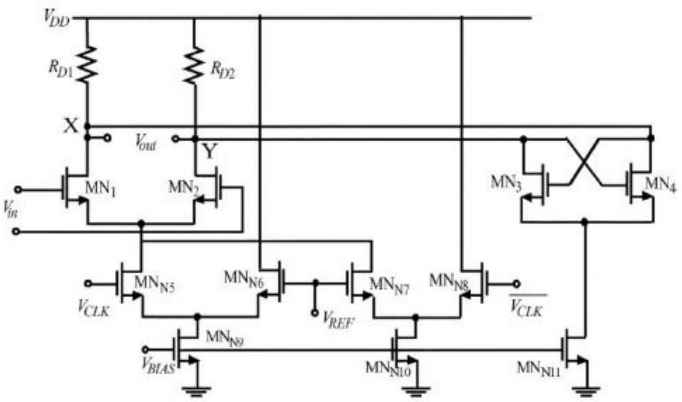


Figure 3-22: Circuit schematic of f_t doubling latch

Figure 3-23 [9] shows another attempt at a frequency divider circuit by combining the tail currents of the two latches and route them through either the master (diff pair) or slave (cross coupled pair) block according to clock signal. This design allows the sizing of the latch pair and diff pair determine the current through them when on. This allows a similar isolated design between the two blocks, but a fluctuation in swing and hard switching still cause problems.

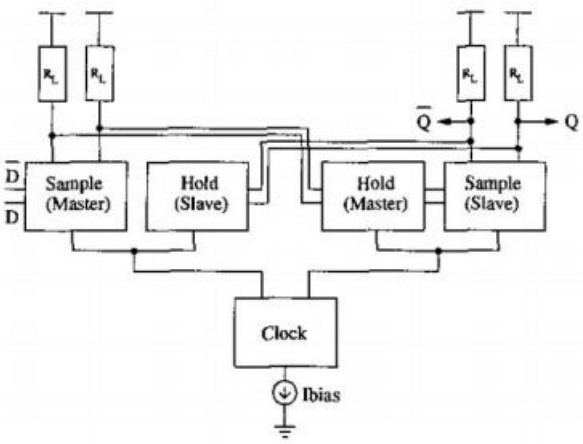


Figure 3-23: Block diagram of a new frequency divider circuit

A new topology is designed for this specific application. Figure 3-24 shows the circuit schematic of the proposed latch. Similar to conventional latch, there is a diff pair for sampling as well as a cross coupled pair for latching. The difference is the extra stand alone cross pair on the side. When sampling, the latch works as an amplifier with the stand alone cross pair and resistors on top as load. When transitioning to

latch mode, the cross pair is already able to start latching without any extra time for turn on. The secondary cross pair then starts helping the latch action when the bias current is steered to it. In this case, all the bias currents are saved, and the output swing will stay constant. Besides, since there are now essentially two latch cells doing work at the same time, only a small amount of current needs to go through the secondary latch in order for the output to hold correctly. This means a smaller clock amplitude is required in order for latching to happen.

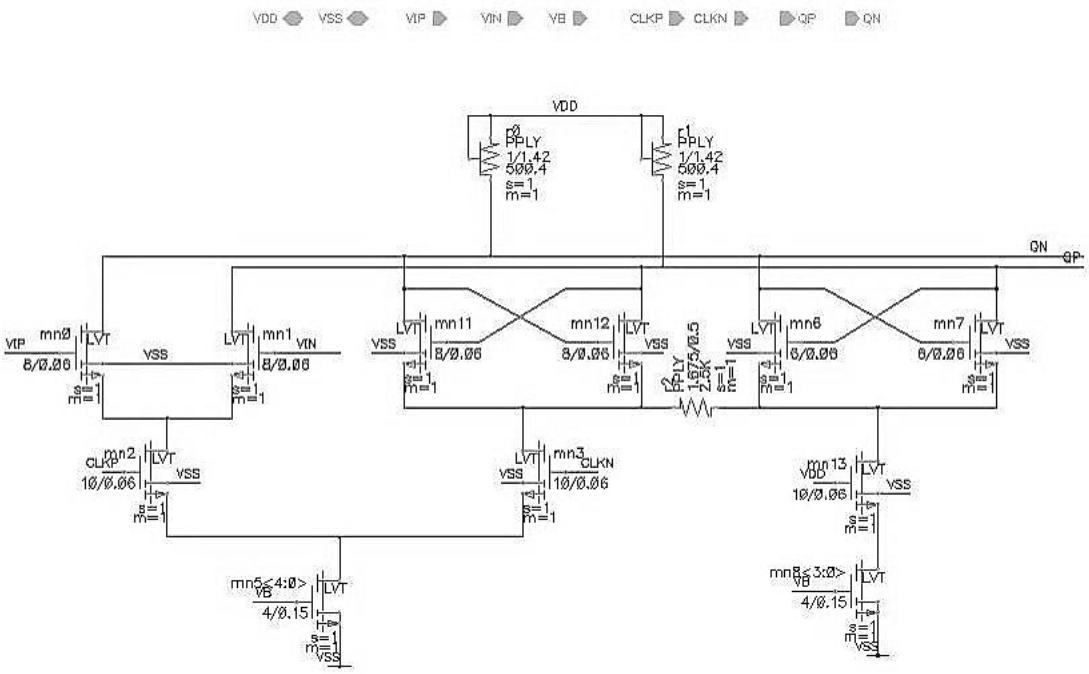


Figure 3-24: Circuit schematic of proposed CML latch

This topology also gives relative isolation in designing the latch and differential pair. There is no sacrifice for speed since the device sizes are comparable if not smaller than the conventional latch. This latch also uses comparable power compared to a conventional one. The resistor in the schematic connecting the two cross paired cells serves as a bleeding resistor. It provides a smaller bleeding current through the secondary latch while it's off, so that it wouldn't have a hard turn on transition and can start latching faster. The extreme cases are when there is no bleeding resistor in which case performance degrades due to slow turn on, and when the two are shorted in which case during sample mode the output sees more parasitic capacitance.

3.3.3 CML Divider Performance

A CML divider uses CML latch in the configuration shown in Figure 3-25. When the clock amplitude is small,the two latches behave as amplifiers and inverted feedback makes this a two-stage ring oscillator. This phenomenon is known as self oscillation in CML dividers [10], in which the divider has a preferred oscillation frequency when clock is differentially zero. The self oscillation frequency is approximately $|g_m R_L - 1|/(R_L C_L)$, where R_L and C_L are the load resistance and capacitance and g_m is the transconductance of the cross coupled pair. Figure 3-26 shows a transient waveform of divider output when clock amplitude is small, and it behaves as a mixer with clock modulated with the self oscillation frequency.

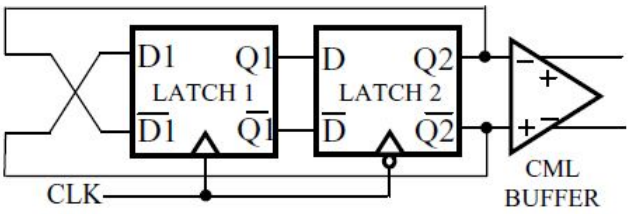


Figure 3-25: CML latch based frequency divider

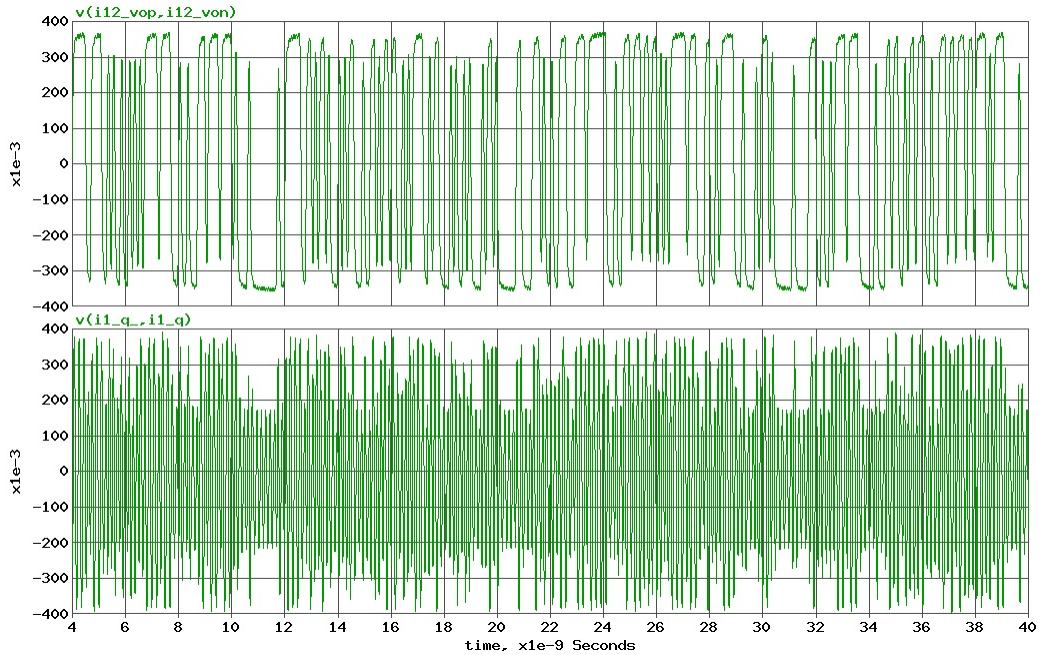


Figure 3-26: Clock modulation when divider oscillates

One can use self oscillation to divide even faster at the frequency of interest. When self oscillation is half of the incoming data fundamental frequency, the divider tend to operate faster. This gives rise of an input sensitivity curve for CML divider in which a minimum input amplitude required exists at divider's self oscillation frequency. Figure 3-27 shows sensitivity curves of conventional divider (red) and proposed divider (blue) when designed to have the same self oscillation frequency at same current level. Clearly the proposed divider has better input amplitude sensitivity (20mV).

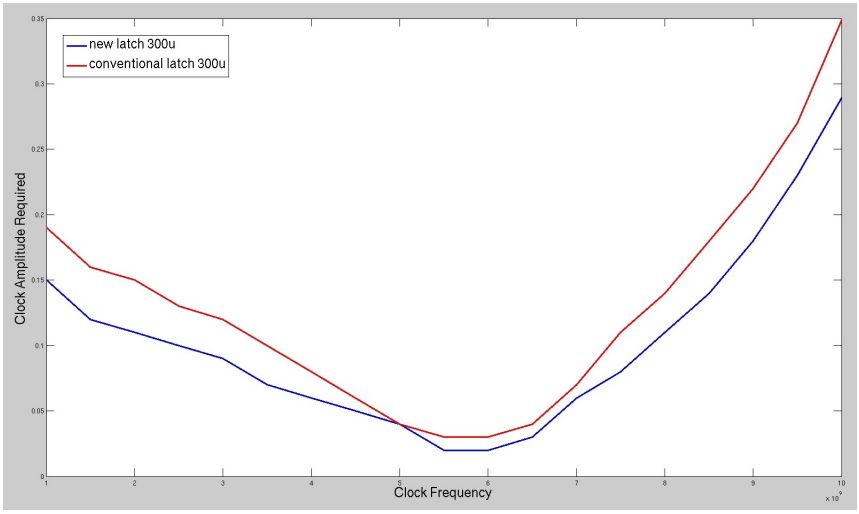


Figure 3-27: Sensitivity curves of conventional and new divider

Figure 3-28 and 3-29 shows the count of output waveform edges plotted against channel length for conventional and new dividers respectively in slow process corner. The conventional divider used is a standard cell that has a much higher self oscillation frequency. For long channels (smaller clock amplitude), the conventional latch's self oscillation contributes greatly to the erroneous output count, while the new divider starts to lose reasonable amount of count at about 17 inch hybrid transmission line modeled channel at 10Gbps (80mV eye opening). In conclusion, the proposed latch is used in the final system for its better clock sensitivity to input clock at comparable power and area consumption. Once the first stage divides down the input clock, a standard CML to CMOS converter is used for logic style conversion and following CMOS dividers will be fast enough to process further.

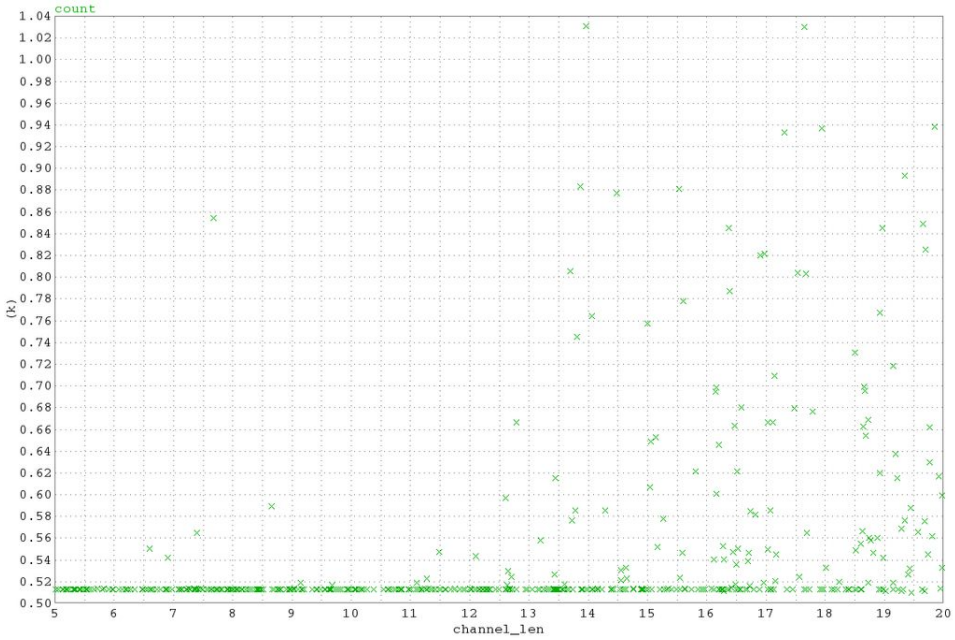


Figure 3-28: Count of edges of conventional divider v.s. channel length in slow corner

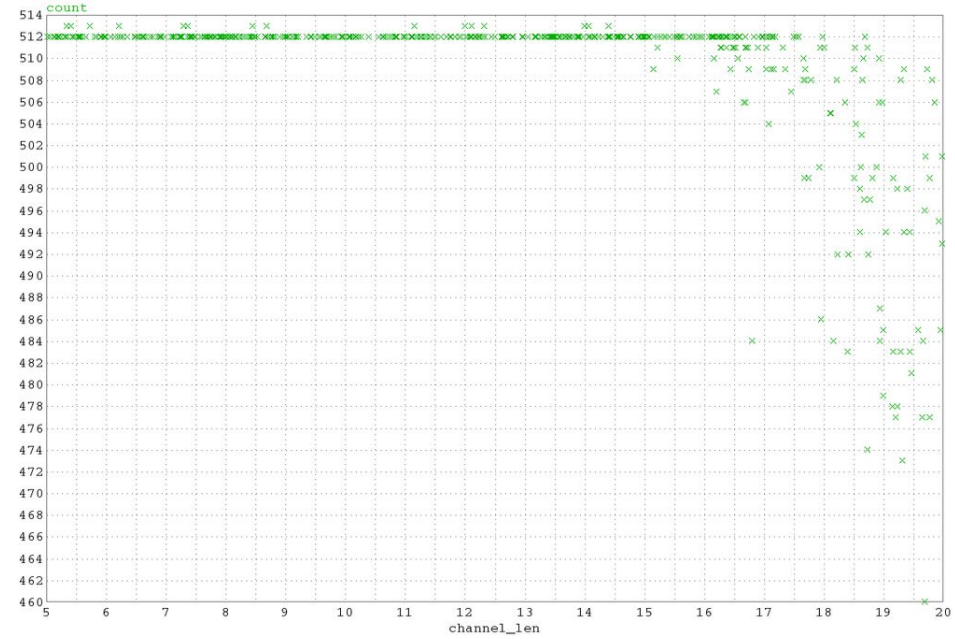


Figure 3-29: Count of edges of new divider v.s. channel length in slow corner

3.3.4 Lambda Estimator Implementation

Figure 3-30 shows the block diagram of lambda estimator. The two clocks in the schematics come from the division stages. Division chain for data clock involves a

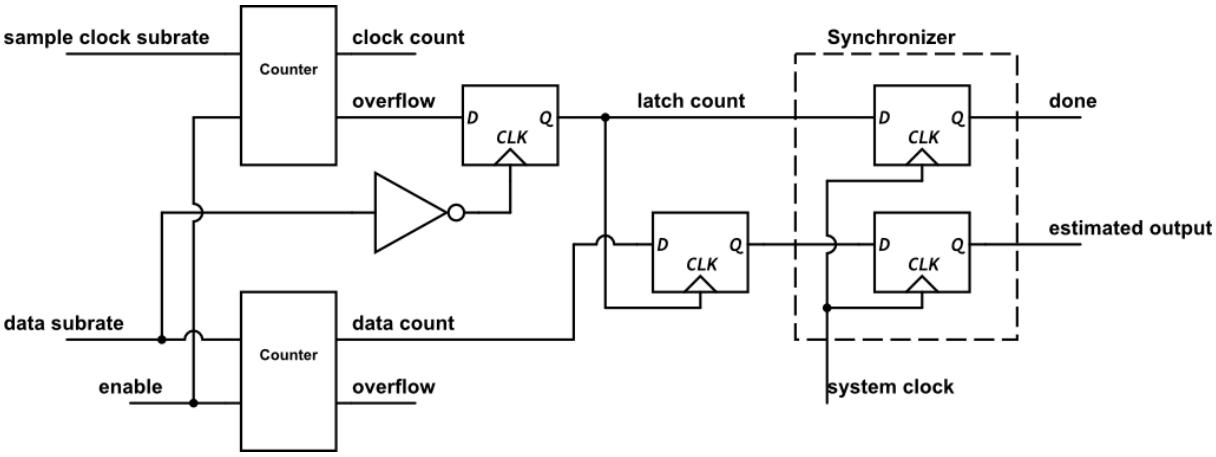


Figure 3-30: Block diagram of lambda estimator

CML stage as front end and CMOS afterwards. The division chain for sampling clock is purely CMOS. Due to the innate 25% transition density, the number of division needed for data is 2 fewer than sampling clock's.

This block is asynchronous, but involves some synchronization and metastability issue. If we use the overflow signal from the sampling clock counter to latch the count of the data clock counter directly, we might run into cases when the data count is still transitioning and we latch a bogus value. As a result, the sampling clock counter overflow signal is synchronized with the negative edge of data clock. The 10 bit counter should be fast enough for output to settle within half of a clock cycle considering our counting clock is 128 times smaller than the original (10Gbps/128 80MHz). This eliminates the possibility of a complete bogus value, but might be off by 1, which is negligible error. Both the latch signal and the latched count is then synchronized with the system clock to be used as outputs to the master control block. The native control block controls when to start counting (by enable signal) and when to reset depending on asynchronous system control signals.

Figure 3-31 shows the place and route output with up to MET2 layer. The shape of the block was chosen to be square and the pin placement was random due to lack

of floor plan of the entire chip; however this should provide a relatively accurate estimate. The block is about $45\mu m$ by $45\mu m$ ($1700\mu m^2$) at a P&R utilization of 75%.Transient simulations have been done to verify the functionality of the block.

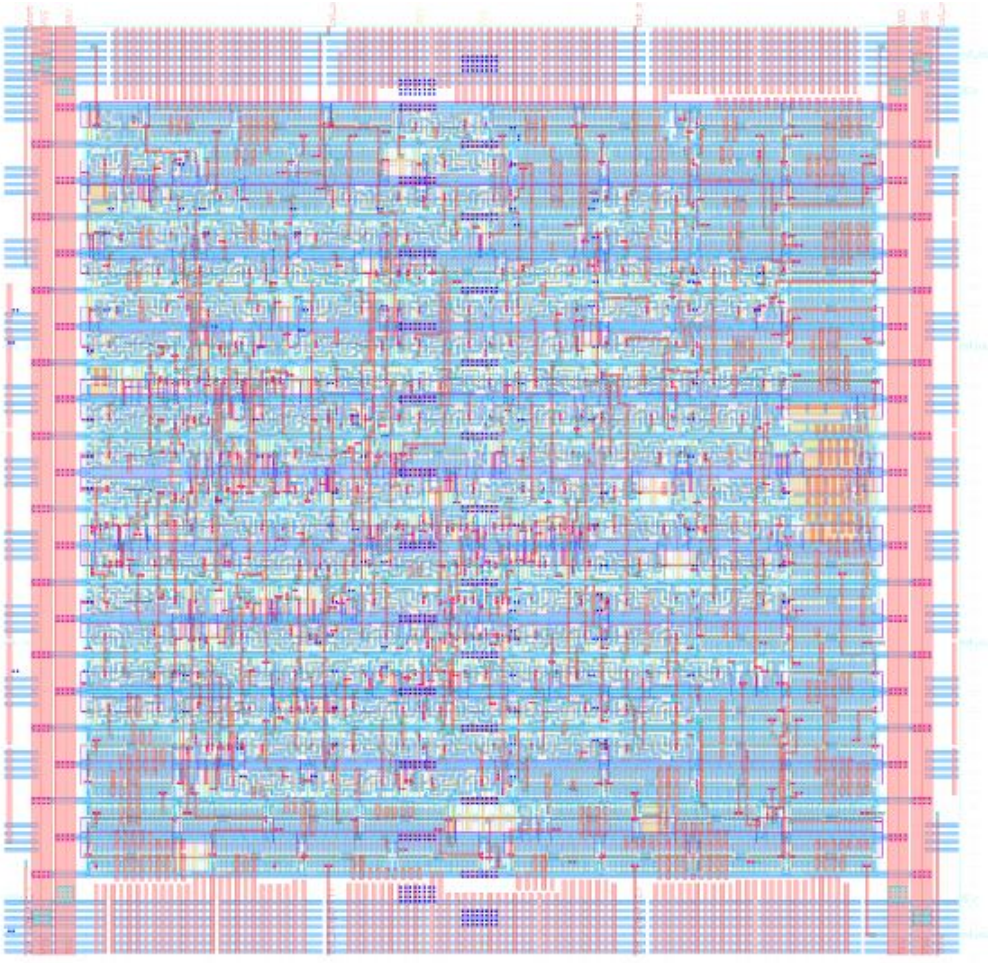


Figure 3-31: Place and route output of lambda estimator

3.4 Digital Reconstruction

The digital reconstruction takes the 3072 samples and initial λ_b estimation and assigns a phase location to each corresponding point while iteratively refining the accuracy of λ_b. The digital block also enables/disables the sample and hold circuit to allow sampling. It also sends master control signal to lambda estimator block. The reconstruction algorithm happens in three phases, coarse search, fine correct with low resolution and fine correct with high resolution. Figure 3-32 shows a flow chart of the reconstruction process.

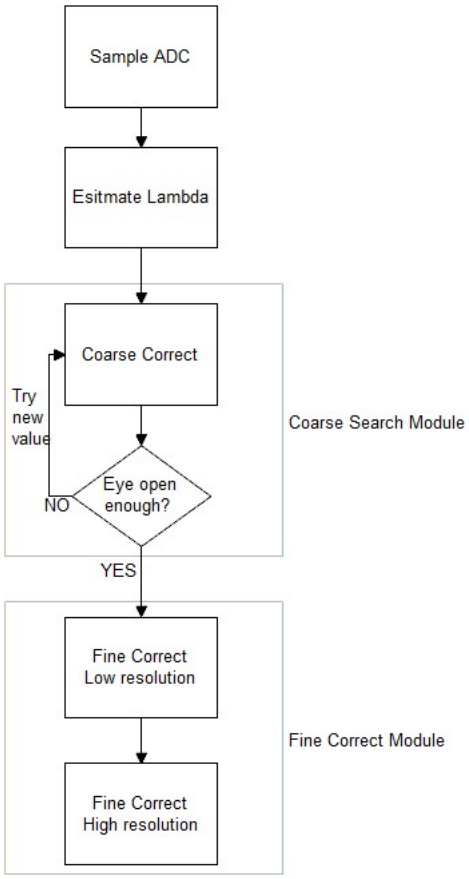


Figure 3-32: Flow chart of reconstruction process

Due to large initial λ_b estimation error, the reconstructed eye will be closed. In coarse search correction phase, a simple search is performed to find a λ_b starting at the initial estimation that provides an acceptable eye opening, which will be used for later fine correct stages.

The fine correct phases look at how the eye opening drifts over 1024 sample points, which is a result of accumulation of λ_b error. λ_b is then deterministically corrected by subtracting the drift velocity. The difference between low resolution and high resolution fine correct phases lies in several parameters such as the sample group size, number of discretization bins and filter buffer size. As a result, a single module could be designed with the ability to use external parameters given which the reconstruction state.

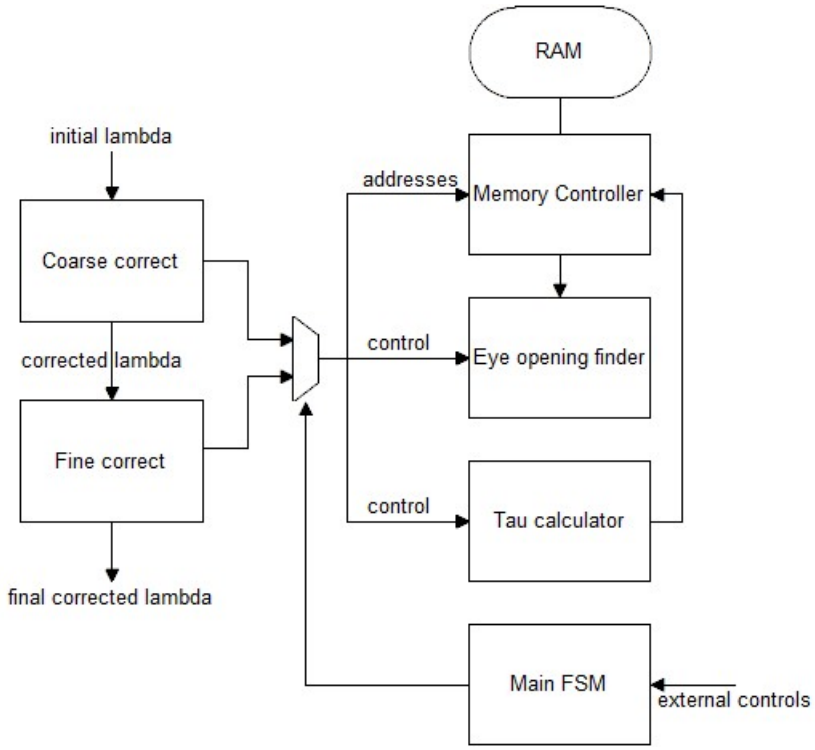


Figure 3-33: Block diagram of reconstruction system

Figure 3-33 shows a block diagram of the reconstruction system. The eye opening finder module and tau calculator module are essential blocks that are shared by either correct modules. Consequently, a main finite state machine (FSM) is designed to multiplex the control signals to these two modules depending on the reconstruction phase. The memory controller module interfaces with eye opening finder and tau calculator for unidirectional read or write operation; it also directly interfaces with undersampling ADC to store samples once given the command from main FSM. Details for specification, challenges in design and implementation for each module

will be discussed in the following subsections.

3.4.1 Memory and Memory Controller

Memory is needed to store the sampled points and their corresponding phase location. Each sampled point and phase location uses a byte, so 6KBytes of memory is needed. For the scope of project, each memory block is modeled as register arrays in Verilog with [7:0] memory[0:3071], and samples and tau have their individual memory array. In an actual implementation, such register arrays will be replaced by actual SRAMs to save area. An estimate of area will be given in 3.4.6.

The memory controller serves as the interface between memory and other modules. In this algorithm the memory controller's job can be reduced to three simple functions

1. WRITE 3072 ADC sample data to sample memory block. This will be one time operation for each snapshot of eye diagram.

2. WRITE 3072 reconstructed tau data to tau memory block. Starts writing when prompted by tau calculator block.

3. READ requested data out of both memory block to eye opening finder, marked by a start and end address.

The memory controller will also have to rewritten when a real SRAM is used, but these functions are simple enough to implement with this behavioral model.

3.4.2 Tau Calculator

Given a λ_b estimation that is passed along by main FSM from initial , coarsely corrected or fine corrected estimation, the tau calculator performs a simple accumulation operation and signals the memory controller to write each result to memory. This module is basically where the reconstruction happens, performing the calculation

$$\tau_n = mod(\tau_{n-1} + \lambda_b, 1) \tag{3.19}$$

The λ_b value is interpreted as a decimal number. With a fixed width accumulator register (18 bits), the mod one operation is done automatically by accumulator overflow. When storing values, only the higher 8 bits are stored, resulting in a high resolution computation and low resolution storage scheme. The accumulator register bit width (18) comes from the 8 bit tau value width and a 10 bit division later on in the correction modules (divide by 1024 for drift velocity).

The module is controlled by either correct module, with a start signal and feeds a done signal back when 3072 tau values have been calculated and stored, emulating a function call. This is one of the simplest modules in the whole system, replacing the complex math method in previous works.

3.4.3 Eye Opening Finder

The eye opening finder is one of the most essential blocks in the reconstruction system, because its accuracy in calculating eye opening and location will directly affect further corrections. Similar to tau calculator, this module is also shared by the two correct modules, and read directly from memory controller with appropriate start and end address. The procedure of finding opening is summarized in the following steps

1. For given data tuples (τ_n, y_n), assigns each τ_n to a lumped bin (bin size dependent on resolution of correct phase).

2. For each bin, keep track of the maximum value smaller than middle value (128), and minimum value larger than 128.

3. When finished reading, take the difference of max below mid and min above mid to get the eye opening for each bin.

4. Apply a running average filer to the difference array to smooth out sharp transitions and random jumps caused by limited samples. Keep track of maximum and minimum values and where max occurs.

5. Returns the range of the eye opening, the deviation between filtered and unfiltered maximums and eye opening time location.

Figure 3-34 shows an example of the filtered (red) and unfiltered (blue) eye opening array mentioned in step 4. The goal is to eliminate any glitches and produce a smooth output to increase accuracy.

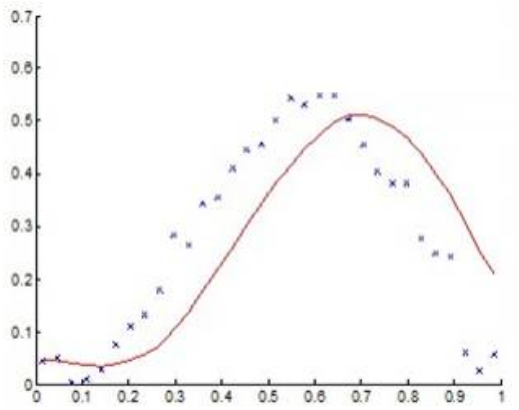


Figure 3-34: Filtered and unfiltered eye opening waveforms

The running average filter is a simple window function. The window size is 8, each with a value of 1/8. In the frequency domain it translates to a low pass filter as shown in Figure 3-35. Ideally, only the eye opening height is needed to access

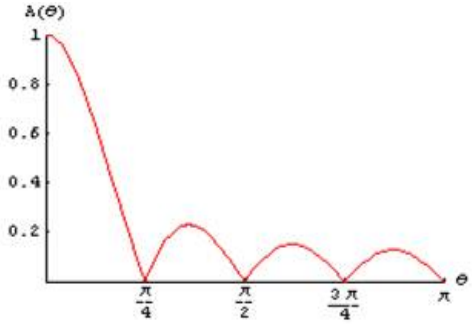


Figure 3-35: Frequency transfer function running average filter with 8 taps

whether there is an opening; however, due to some special cases, a sudden jump in erroneous correction can result in a filtered output exceeding eye opening threshold value. Figure 3-36 shows a comparison of filter outputs using wrong λ_b (left) and correct λ_b (right). A sudden opening in the reconstructed eye on the left resulted in a maximum value that is too large on the filter output, causing correction error, while the correct opening filter output follows closely with the unfiltered waveform in shape and amplitude.

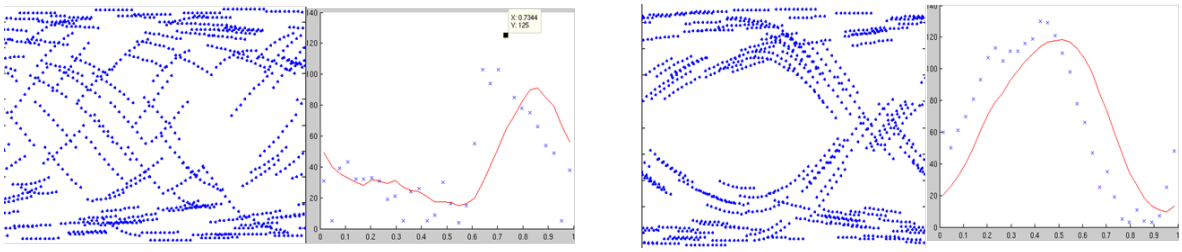


Figure 3-36: Filter outputs of wrong (left) and correct (right) λ_b estimation

If only eye height is given from this module, the correction modules will make false decisions. To prevent such cases from happening, another parameter is generated, which takes the deviation of the filtered output maximum from the unfiltered maximum. Intuitively, the low pass filter reduces the high frequency components of the waveforms by a lot, resulting in a large change in sudden jump. For a correct λ_b estimation, the high frequency component is minimal to start with, so applying a low pass filter will not create large deviations in maximum values such as in the left case.

Actually implementing the running average filter requires evaluation of several design trade-off, mainly between filter time and resource cost. One method is to use an 8 operand adder, and moves difference values in the array in a circular fashion. Each clock cycle will generate a output that's the average of the next 8 elements in the circular buffer. Using a small two operand full adder, one can sequentially add 8 values in 8 cycles and do so for all the elements in difference array, taking a lot more time. One final approach is to pre-populate an accumulator with 8 elements, and for each consecutive cycle, subtract the first value and add the next value to emulate the circular shift. Table 3.1 shows a qualitative comparison between these three methods in terms of resources and computation time for a 64 element difference array. The push and pop method is used in this design for its small resource usage as well as computation time.

Method	Resources/Cost	Time (clock cycles)
One shot add	Large 8 operand adder	64
Sequential add	Small full adder	8×64 = 512
Push and pop	Small subtracter and adder	8+64 = 72

Table 3.1: Qualitative comparison of filter methods

The eye opening module is also implemented as a function call in which calculation is triggered by a start signal and flags when done.

3.4.4 Coarse Search Correct

The coarse search correct module is one phase of correction in which it searches for a λ_b given the initial estimation that will produce an open eye. The goal is not to search for the precise value, but to quickly converge to a value that shows the final eye shape with considerable opening for later stages. As a result, the coarse search module is very iterative, doing trial and error. This could be implemented with a simple state machine, as shown in Figure 3-37. When an external start signal is asserted, the FSM goes to RECONSTRUCT state, in which tau calculator assigns phase values using initial estimation. In FIND_EYE state, the eye opening finder will return reconstructed eye opening parameters, and if the module decides it's not open enough, then it will update the trial λ_b and try a new estimate. This iterative process continues until either an open eye is found and the corresponding λ_b value is latched for next stage, or maximum number of trials have been reached and flags a failure.

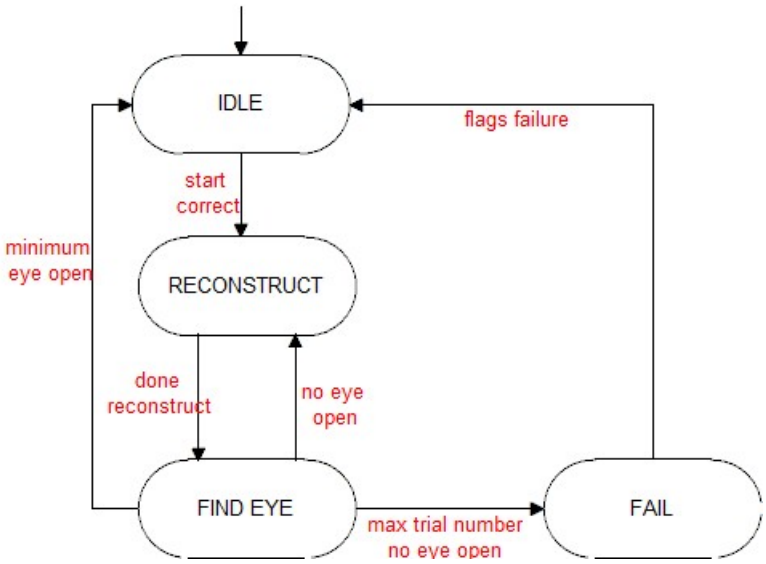


Figure 3-37: Coarse search correct FSM

The state machine will have four main parameters to adjust. MIN_RANGE and MAX_DEVIATION determine the criterion for an open eye, as discussed in the pre-

vious section; empirically MIN_RANGE is set to 32 and MAX_DEVIATION is set to 16. The STEP_SIZE determines the λ_b increment/decrement every time a new value is tried, and it is set to $1/2^{11}$. MAX_TRIAL is the maximum number of trials the search will performed; this design chooses 512 trials, which translates to ± 0.125 range from initial estimation. The MAX_TRIAL can of course be adjusted to cover the whole λ_b space, in which case no initial estimation is really needed.

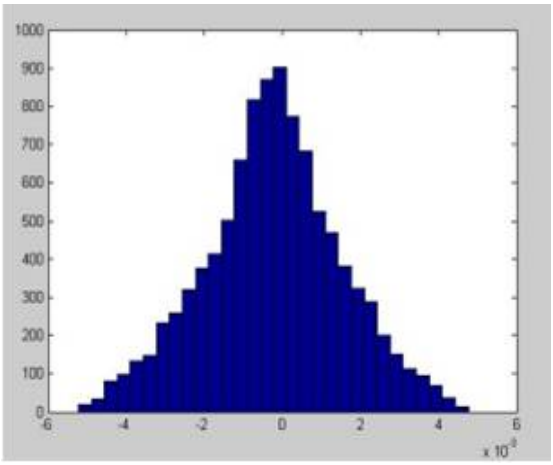


Figure 3-38: Histogram of lambda estimation error

Figure 3-38 shows a histogram of λ_b estimation error running 5000 trials with random variables in data rate (1-10Gbps), ppm error in data rate (0-100ppm), and initial phase offset for counter (0 to 2π). The diagram is Gaussian with mean very close to 0. One design choice is to decide what search mechanism to use. Let M be number of one sided maximum trial steps, N be the actual number of steps of correct lambda away from initial estimate, and n be the actual number of trials to find an open eye. If we choose to search one way first and then the other way when we fail to find a value, the expected number of trials for this method is

$$\bar{n} = \frac{1}{2}(M+N) + \frac{1}{2}N \tag{3.20}$$

assuming we get equal chance of guess the right or wrong direction. If we perform a

left-right alternating search, then

$$\bar{n} = \frac{1}{2}(2N) + \frac{1}{2}(2N+1) = 2N + \frac{1}{2} \tag{3.21}$$

The second method will have fewer trial steps than the first method on average when

$$2N + \frac{1}{2} < \frac{1}{2}(M+N) + \frac{1}{2}N$$
$$N < \frac{1}{2}(M-1)$$

In our case, $M = 256$, which yields a bound of 128 steps, or absolute error of about 31m from initial estimation. Assuming that our initial estimation is relatively accurate, the second method will obvious result in a fast search time. In the coarse search module, a left-right alternating search mechanism is then used, which requires two registers storing trial values and use appropriate value dependent on current trial.

3.4.5 Fine Correct

An overview of fine correct algorithm has already been presented in 2.4. The fine correct module looks at eye opening locations of two consecutive groups of samples, and by finding the drift velocity the module corrects λ_b by subtracting this error. This section will discuss the design issues encountered in this module and their solutions.

Figure 3-39 shows the state transition diagram of the fine correct module. Different from coarse search correct module, the fine correct module is sequential in states. Once a start signal is asserted, the module asks the eye opening finder to find the first sample group's eye location, then the second group's by outputting appropriate start and end read addresses. Once the information is gathered, the new λ_b is corrected during the state transition to RECONSTRUCT, in which new tau values are written to memory with corrected λ_b. The next two states, FIND_EYE and RECONSTRUCT_BAR are used to deal with the ambiguity in correction direction.

Consider the situation drawing in Figure 3-40. There are two ways eye 1 can drift

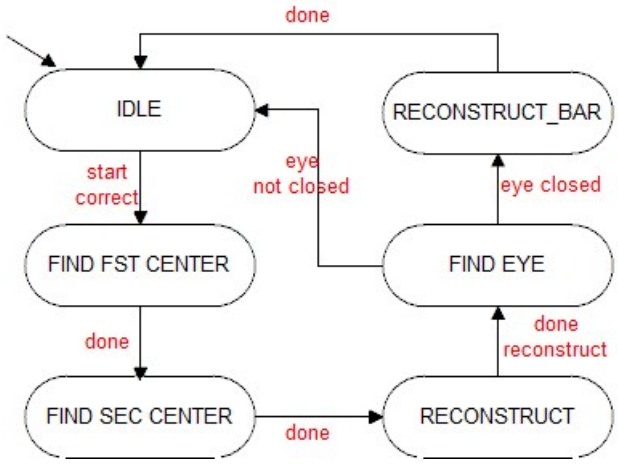


Figure 3-39: Fine correct FSM

to eye 2, following either the red or green arrows. The default way of correcting in this module is take the obvious red path, in which center 1 is directly subtracted from center 1. However, if λ_b goes in the opposite direction and with sufficiently large error, the eye center of next group can wrap around the period (green path), leaving the initial correction completely wrong. Therefore, another checking state FIND_EYE is added in the state machine to check whether the eye is still open after correction. A correction in the wrong direction will cause the eye to close again, so the same criterion used by coarse search module can be reused. If eye is closed after correction, then wrap around must have happened and a correction in the other direction is required, which is RECONSTRUCT_BAR state.

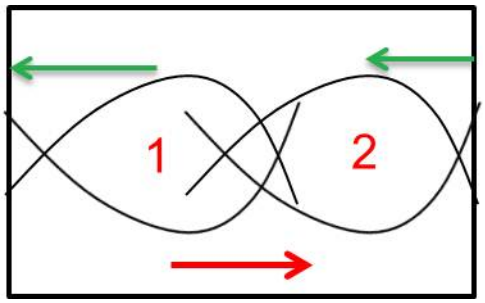


Figure 3-40: Ambiguity in correction direction

The fine correct module will be used twice in this design, with either low or high resolution correction. Table 3.2 shows a comparison in parameters and module outputs when fine correct module is used in low and high resolution. The start and

end read addresses going into eye opening finder is multiplexed by the main FSM depending on the correction phase. The eye opening finder is also going to adjust the filter discretization bins according the phase variable. In this fashion the state machine can be reused again for a second round of high resolution correction without major changes. The simple interface between modules (start, done, and return values similar to a function call) makes this design possible.

	Low resolution	High resolution
Sample group size	1024	2048
Read addresses for 1st group	$0 -> 1023$	$0 -> 2047$
Read addresses for 2nd group	$1024 -> 2047$	$1024 -> 3071$
# of filter bins	32	64

Table 3.2: Parameter comparison for low and high resolution fine correct

3.4.6 RTL Synthesis and Area

Area is one of the main concerns in this system design. In order to make it fully integrated, ideally the area should be small enough to be easily plugged into any existing chips without an excessive area overhead. Power might be another criterion, but since the system is not constantly computing a relatively large power consumption for a short period time is still acceptable. In order to obtain an estimate of the area, the computation logic is synthesized and place and route (PNR) tool is used to take area utility factor into account. Memory will be another major contribution to area. An estimate is obtained by adding the logic area and memory multiplied by a utility factor of 0.75 (taking memory controller into account).

Figure 3-41 is the PNR output of only the computation logic blocks with aspect ratio of 1:1, which includes eye opening finder, tau calculator, coarse correct, fine correct, and main FSM modules. The area is 255 μm by 255 μm ($\approx$65000μm^2). The utility factor is quoted to be 0.804 in this result, which shows that the design is quite compact.

The memory area is estimated with the data book provided by the TSMC 65nm memory compiler. The memory used is the Single Port SRAM. For a 3072$\times$16 Single

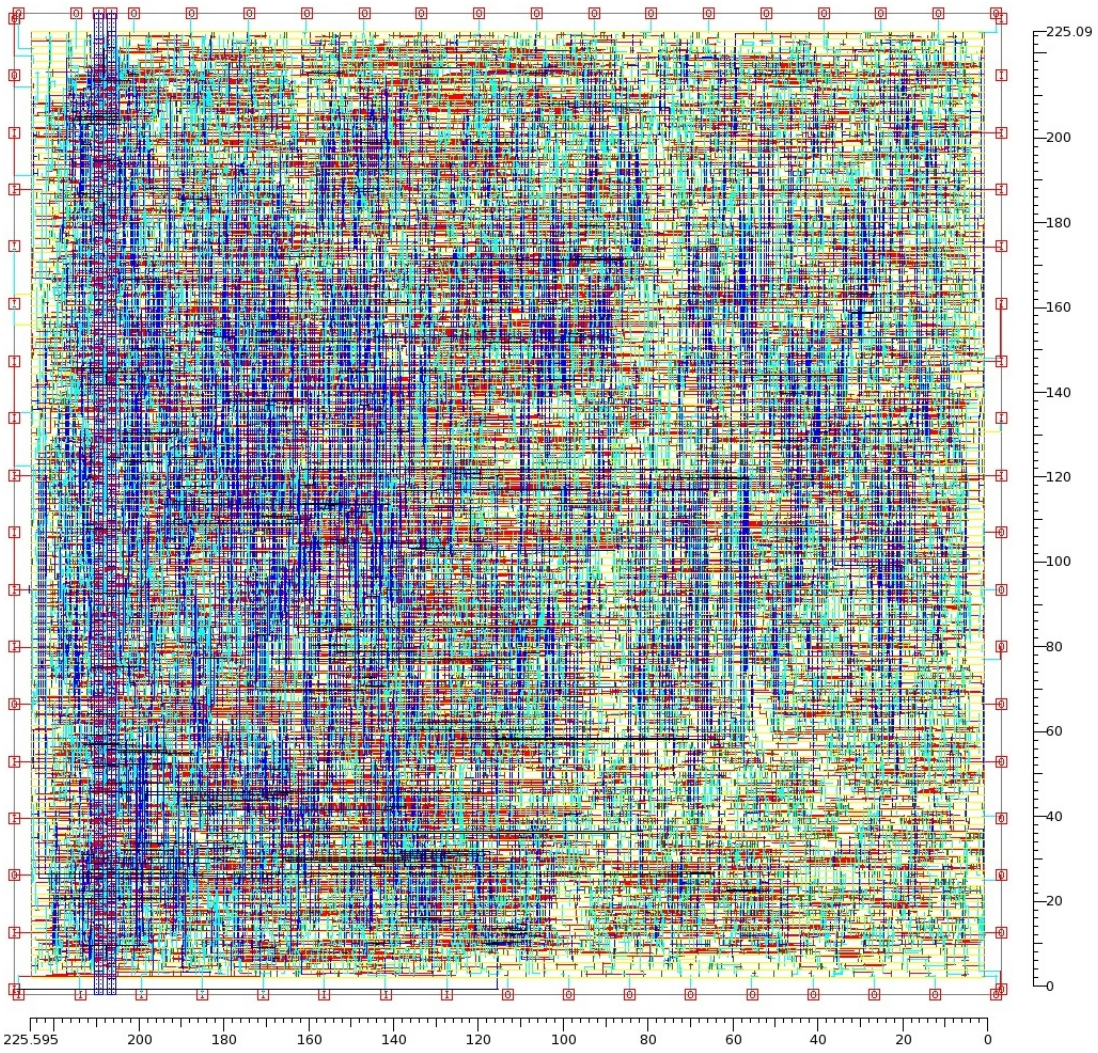


Figure 3-41: PNR logic only output

Port SRAM, the layout area is quoted to be 183.925μm by 231.385μm ($\approx$ 42557μm^2), which is comparable to the computation logic blocks. The design only requires single address access at a given time, which is managed by the memory controller block, therefore a single port SRAM will suffice; just to get a sense of the size of a Dual Port SRAM in 65nm, a 4096 by 16 memory (databook has no data for size of 3072 by 16) is quoted to be 307μm by 380μm ($\approx$116660μm^2), an area quite larger than a SPSRAM. A dual port SRAM is only useful when there are other blocks trying to access the memory at the same time the reconstruction is happening, for example the PC interface is trying to read memory while reconstructing. It adds simplicity to

memory controller design, however incrementally. A SPSRAM is strongly preferred over a DPSRAM in this scenario, and the memory controller should be redesigned to maintain the same simple functionalities and have an extra interface with PC communication modules to read out the final results.

With a SPSRAM (3072 by 16) and redesigned memory controller, it is a safe estimation to say that the memory and logic parts have about the same area consumption. The total area for digital reconstruction block then is approximately $2 \times 65000\mu m^2 = 0.13mm^2$, equivalent as a square of $360\mu m$ as its side. The addition of sample and hold, lambda estimator, ADC, and clock generator will increase the size, but not significantly.

Chapter 4

Simulation and Results

System level testing and simulation pose challenges in this design. The system requires long period transient simulation to verify the functionality of the whole reconstruction flow. This chapter presents testing strategies and results to support the functionality of the system. Behavioral models are developed to speed up simulations. Test benches are created to perform modularized tests. Entire system is tested eventually and special test cases are used for results analysis and gain intuition in limitations in system performance.

4.1 Behavioral Models

Chapter 3 discussed implementation details on the major blocks, however simulating on transistor level will not be efficient. Behavioral models are developed to emulate the analyzed building blocks such as sample and hold and frequency dividers. The digital building blocks, lambda estimator and digital reconstruct, are already in written in Verilog RTL code, which can be simulated behaviorally.

The channel model that produces the eye diagram uses a hybrid transmission line model. The parameters of this model comes from measurement data in the lab in order to mimic real backplane channel as closely as possible. By changing the length of the channel we change the eye shape, jitter, opening, etc.

The sample and hold is modeled by an input amplifier, perfect switches, and an

output amplifier. The combined gain of the amplifiers is unity, assuming the gain error has been calibrated out. The combined bandwidth of the system is 10GHz, the worst case number from the Monte Carlo simulations. An 8-bit ADC model is used to convert the sampled analog value into digital domain. Nonlinearity is not modeled in this case since it doesn't affect the correctness of reconstruction other than simply alter the eye shape a bit. Sampling clock is modeled with a 3ps RMS jitter and a 5ps edge time constant at 201.67MHz.

Frequency divider for data uses a front end differential to single ended comparator with 100mV hysteresis, followed by 5 CMOS divide by 2 stages. The hysteresis models the error from the divider when input eye is too small. The frequency divider for clock is simply a chain of 7 CMOS divide by stages. The lambda estimator and reconstruction system use their RTL behavioral codes directly for simulation. The system clock is 75MHz.

4.2 Test Strategies

The analog building blocks have been characterized individually to conclude their specifications and come up with their behavioral models. The digital blocks on the other hand require more rigorous testing to verify functionality. There are several issues and challenges in these tests; the memory registers have to be pre-loaded with values in order to directly going into correction phase other than sampling new values each time. Special cases have to be generated to detect edge cases such as the eye wrap around mentioned in fine correction. It would be ideal to have a single test bench setup to simulate both the coarse search module and fine correct module since they share memory controller, eye opening finder and tau calculator.

Figure 4-1 shows an example test bench used to simulate the fine correct module. The memory inside memory controller has been preloaded with sample values generated in previous cases when testing sample and hold, and random initial values for tau memory block. The eye opening finder and tau calculator are wired to memory controller with corresponding ports. The fine correct module is the one that is being

tested, and an estimation has been preset by a logic reference block to bypass the coarse search module. Each module in this system is designed to be like a function call, therefore a single start signal will trigger the test. In this test bench there is no high precision analog component, therefore it becomes very fast in simulation speed and results can be easily probed.

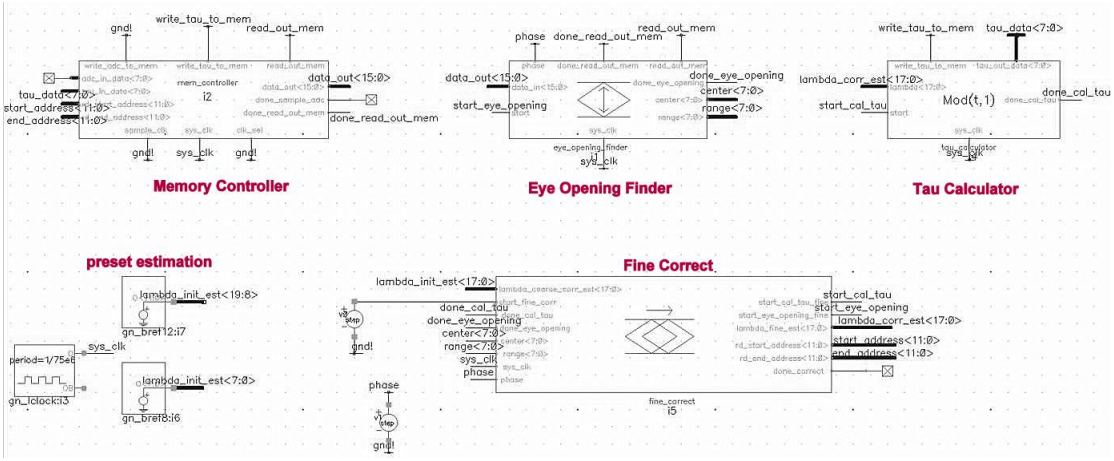


Figure 4-1: Test bench example for fine correct module

4.3 System Test

When each individual block has been tested, the whole system's behavioral model is wired together to perform system level test. The main FSM in this system is designed to perform the whole sequence, from sampling to reconstruction. In a final product, a new FSM can be designed, in which a previous corrected reconstruction λ_b can be remembered and directly used for reconstruction instead of going through initial estimation again; the main FSM is open for design to add in more features to interface with external user inputs, but in this test a simple entire sequence FSM is used.

Figure 4-2 shows the final system. Each block represents a behavioral model for each major building block. Similarly the test is triggered by the start signal to the digital reconstruction block as shown. Figure 4-3 shows example output waveforms for one of the tests. With a 201.67MHz sampling clock for 10Gbps data and 75MHz digital system clock, the whole reconstruction took about 2.3ms to complete.

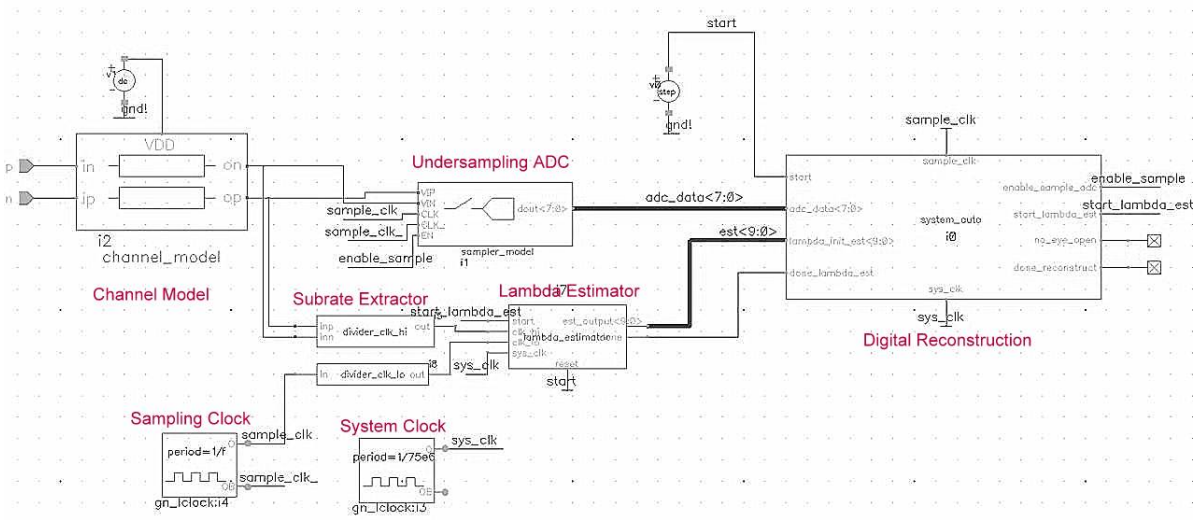


Figure 4-2: Final system test schematic

Figure 4-3: Example output waveforms

The reconstruction is marked by 5 stages, which are marked by 5 arrow brackets in Figure 4-3.The shortest stage is the ADC sampling stage, in which 3072 ADC values are taken and stored, spending $3072/201.67MHz \approx 15\mu s$. The second stage is the initial lambda estimation phase, in which counters in lambda estimator work the hardest, spending $2^{10}/(201.67MHz/2^7) \approx 650\mu s$. The longest time happens in coarse search stage, which depends on the accuracy of the initial estimation. In this particular case, it took 25 trials to reach the desired coarsely corrected lambda with $1.4ms$. The fine correct spent a total of $250\mu s$ for both low and high resolution correction marked by the last two arrows. In this case the high resolution correct stage experience a eye wrap around, resulting in a slightly longer correction time. The λ_b error started with 7m, and ended with 23μ.

4.4 Results

When the system is verified, several different cases are tested to obtain a qualitative sense of the performance and compare to expectations. The four cases used are the combinations high data rate and large eye, high data rate and small eye, low data rate and large eye, and low data rate and small eye. Figure 4-4 through 4-7 show results for these four cases respectively.

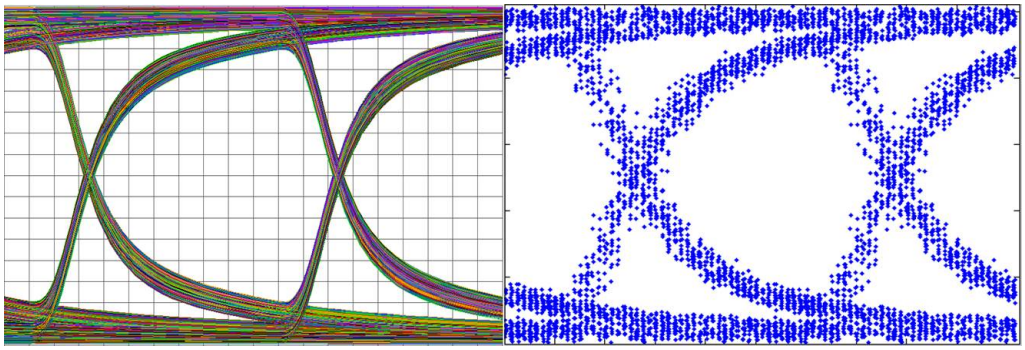


Figure 4-4: Output of high data rate and large eye

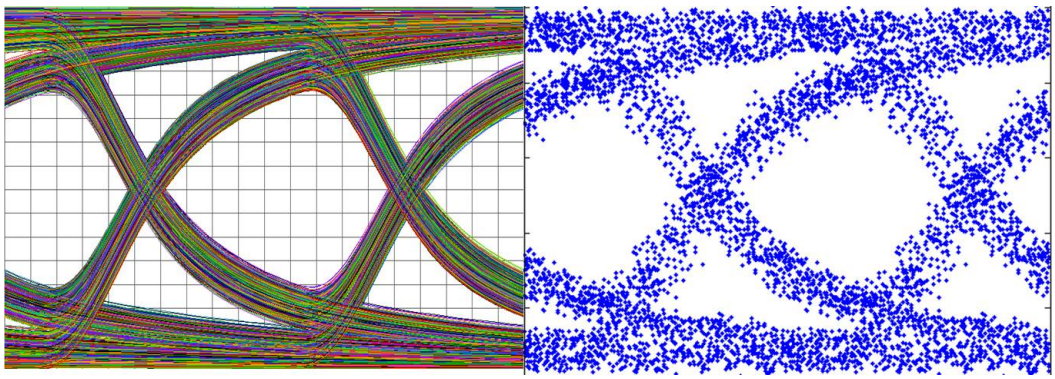


Figure 4-5: Output of high data rate and small eye

The reconstruction outputs resembles the original eye diagram quite well. In the high data rate cases (10Gbps), the reconstructed eyes are closed a bit due to the limited bandwidth of sample and hold circuit. In the large eye case, not only is the eye closed a bit, but there is some added jitter, mainly from the clock source and the finite error in the corrected λ_b. Despite the small error in λ_b, a small drift dispersion can still be seen in the large eye case. In the small eye case, the eye shape is relatively conserved even with the limited bandwidth since high frequency components are not

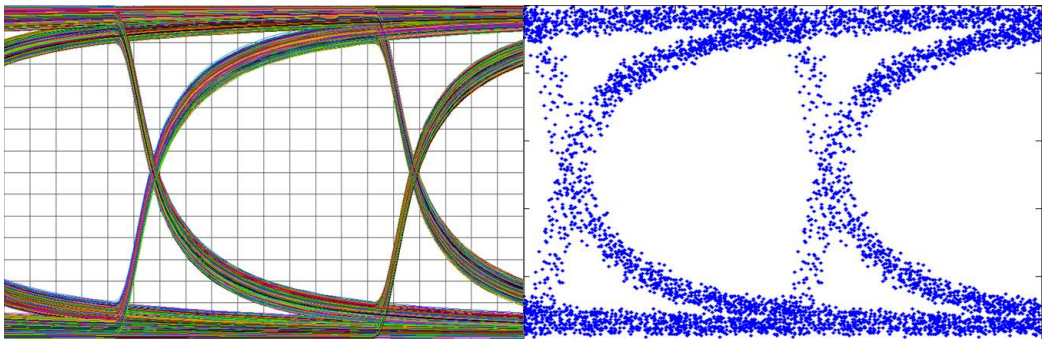


Figure 4-6: Output of low data rate and large eye

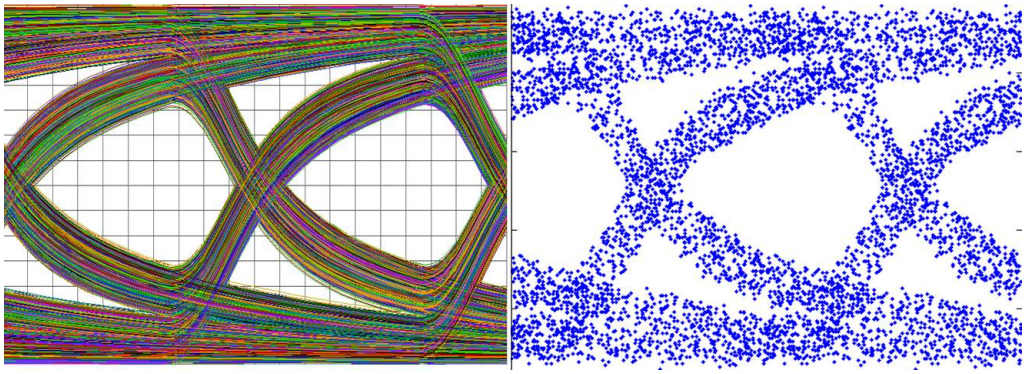


Figure 4-7: Output of low data rate and small eye

large to begin with. The effect of added jitter from sampling clock can still be seen by dispersed points; the drift dispersion is negligible.

The outputs for low data rate cases (1Gbps) looks better in terms of preserving eye shapes. The limited bandwidth effect does not show up in this scenario. However, in the large opening case, the drift dispersion effect is dominant, while there is essentially no jitter in the original eye diagram. The small eye case yields the best result, in which the output almost looks identical to the original eye.

The drift dispersion effect implies that for a finite lambda error, there exists a minimum jitter that we can obtain on the output reconstructed eye. We can obtain more samples and iterate through more fine correct stages, with higher correction resolution. It will be a trade off between the minimum acceptable reconstruction jitter performance is and computation power and area.

Chapter 5

Conclusion

5.1 Summary

This research project is motivated by the demand of on chip signal integrity monitoring, specifically for eye diagram. As data rate increases on channel links, a more cost effective method needs to be developed for quick assessment of eye quality in digital communication. Different from synchronous technique, asynchronous reconstruction does not require explicit knowledge of incoming data rate, and is compatible with a wider range of input speeds.

This work proposes a new asynchronous reconstruction technique based on undersampling theory. A simpler counting mechanism is used to replace the traditional periodogram approach for finding the aliased frequency between data and sampling clock. A new reconstruction algorithm is also developed to iteratively correct for error in the initial aliased frequency estimation by search and deterministic correction. It is advantageous over the DSP method because no complex math is required without significant sacrifice in area, power and/or reconstruction time.

Using TSMC 65nm CMOS process, essential building blocks such as the 10GHz bandwidth sample and hold and a novel CML frequency divider have been designed on the transistor level. Detailed analysis on sampling clock jitter and lambda estimation on PRBS are done to fully characterize performance and specifications. Digital reconstruction blocks are fully implemented in Verilog RTL code and a synthesis at-

tempt is made to estimate the finished block area to be about $0.13mm^2$. Behavioral models are used to perform system level simulations and tests, which yields promising results. The maximum data rate the system is designed for is about 10Gbps, limited by the sample and hold circuit bandwidth.

Full fabrication of the chip is out of the scope of the project, which involves mainly proving the validity of the proposed method and system. However, most important building blocks have been implemented and detailed specifications are presented for other necessary blocks. Floor-planing and full system tests on transistor level are as critical in the future to reach fabrication of such monitoring chip.

5.2 System Usage and Application

The main assumption of the system is that the input sequence exhibits random nature (e.g. PRBS), which is exploited to estimate λ_b. In this section, recommended usage is presented while on the other hand the system is argued to be robust against different input sequences if monitor time is a minor concern.

The frequency offset and error due to deviation from 25% positive transition density will thus result in a longer search time in the digital reconstruction module. When the ppm error results in a data frequency offset greater than sampling frequency

$$\epsilon_f f_{data} > f_{sampling} \tag{5.1}$$

we are no longer confident that our initial λ_b estimation is near the true value. Therefore, it is recommended that the preamble of input data is PRBS of high number of bits, long enough for ADC to finish sampling 3072 points. After the first reconstruction is made, the system locks in onto a corrected λ_b, which shouldn't change if the data and sampling frequencies stay the same. The fine correct module should still be used to correct for low frequency jitter as time goes on every time an eye diagram is reconstructed. Bypassing the initial estimation and coarse search blocks will save reconstruction time significantly (even without bypassing, the coarse search block will

only iterate once if λ_b is locked in).

The worst case then is for the coarse search module to search through the whole λ_b space, when the error condition above holds true. Figure 5-1 illustrates the upper bound on trial steps needed with respect to different positive transition density. The extreme cases will be very low and high transition probabilities. We define the corner transition densities as $0.25(1 \pm \frac{f_{sample}}{f_{data}})$, beyond which the upper bound simply becomes the entire search space. There is a minimum number of search steps even when transition density is 25% due to finite estimation resolution (10 bits).

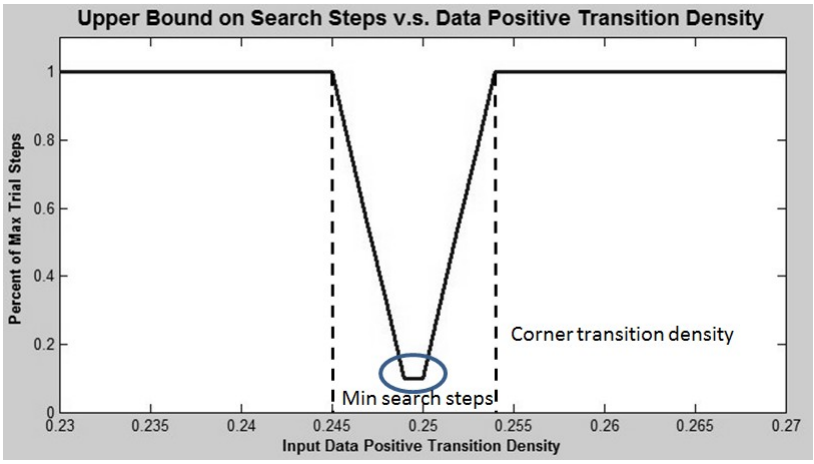


Figure 5-1: Search step upper bound v.s. transition densities

The time it takes to search through the entire λ_b space is dependent on the trial step and system clock frequency. In this example, maximum trial number is 2^{11} and system clock frequency is 75MHz, translating to about 0.11 seconds of search time. When the system is only used as a monitor (similar to figure 5-2, this time scale is still small enough for the application, and the bottleneck should be the communication speed of transferring data from the chip to PC. Once again, this is a one-time search mechanism and once a corrected λ_b is found, following reconstructions can directly use the locked value assuming no frequency changes.

Furthermore, the fine correct mechanism is fast enough for the system to also be used in an equalization adaptation loop, shown in figure 5-3. The proposed system naturally becomes an extension to many link equalizers that utilizes digital blocks. With further digital processing, more detailed eye diagram information can be ex-

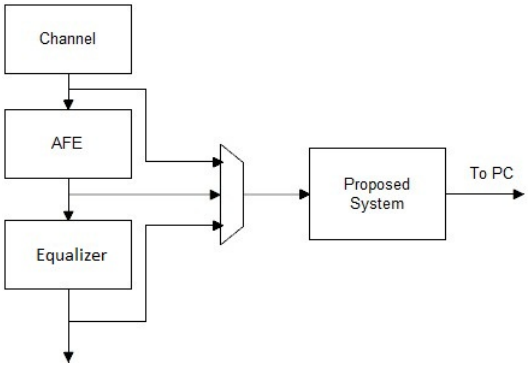


Figure 5-2: Example usage of proposed system as signal quality monitor

tracted to allow better optimized equalizer settings. After the initial λ_b lock, each consequent reconstruction only takes approximately $300\mu s$ at 75MHz system clock. More sophisticated system level design can be realized to embed the eye monitor for both open and closed loop operations.

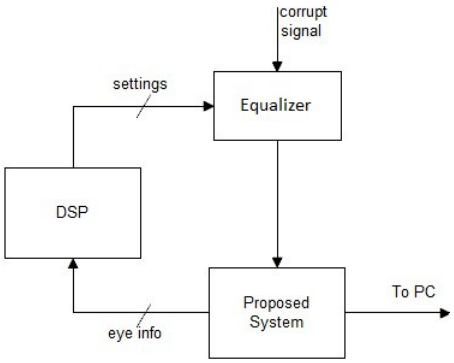


Figure 5-3: Example usage of proposed system in equalizer adaptation loop

5.3 Future Improvements and Explorations

The eventual goal of this project is to fabricate a fully integrated on chip solution for monitoring digital signal quality with eye diagram. This research shows good results that a cost effective method is feasible in building such a system on chip and could be plugged into existing link communication products. Several steps must be taken to reach this eventual goal, and several refinements could be done along the way to make informed trade-off choices.

5.3.1 Fabrication

The main missing blocks in the system that hasn't been fully implemented are ADC after sample and hold, memory and sampling clock generator. The sample and hold might need to be modified to interface with the ADC. One aspect that has not been analyzed extensively is the noise performance in sample and hold and ADC. These might have good and bad implication about the system. If reconstruction block does not require full 8 bit resolution to work properly, then it loosens on the noise spec as well as saves memory. If 8 bit resolution is required, then further techniques should be explored to lower noise if it is found to be unsatisfactory.

The whole system's floor-plan is also critical in that the final area can not be too large. The layout of the sample and hold and place and route outputs of lambda estimator and reconstruction block do not necessarily resemble the final product, but simply provides an estimate of the area. The digital reconstruction block will be the largest portion in the chip due to memory. A PC interface such as a standard I^2C block is to be added.

5.3.2 Area Saving

A small area while keeping reasonable reconstruction quality might be the most attractive characteristic for a final SoC in this application. There are several ways that can be explored to achieve area saving while making informed trade-offs.

The memory size is designed to be 8 bit for each word, both for sample and tau data. A closer look in the reconstruction method shows that only higher 6 bits of tau data are ever used when finding eye opening, which means 6 bits might be good enough for reconstruction quality in terms of lumped eye diagram. The same might apply for sample resolution, in which case is determined by noise and whether reconstruction method has enough information to proceed. If true, then we save $3072 \times 4 = 1.5$K bytes of memory space, going from 8 bits to 6 bits.

One other saving comes from the simplification of reconstruction algorithm to only one fine correct stage. The purpose of multiple fine correct phases is to incrementally

improve the lambda estimation to higher resolution. However, if the number of samples are small, a slightly larger estimation error will still be tolerable since it wouldn't accumulate as much. One fine correct stage only requires 2048 data samples instead of 3072. The trade-off here becomes a slightly larger error in lambda for simpler logic and smaller memory and area. Changing from 3072 to 2048 points results in 2K bytes of memory saving. Experiments show 2048 points with only one fine correct stage is still acceptable as shown in figure 5-4. The experiment data rate is at 10Gbps, and even smaller eye can be reconstructed. Fewer samples also mean shorter sampling duration, loosening sampling clock jitter specification.

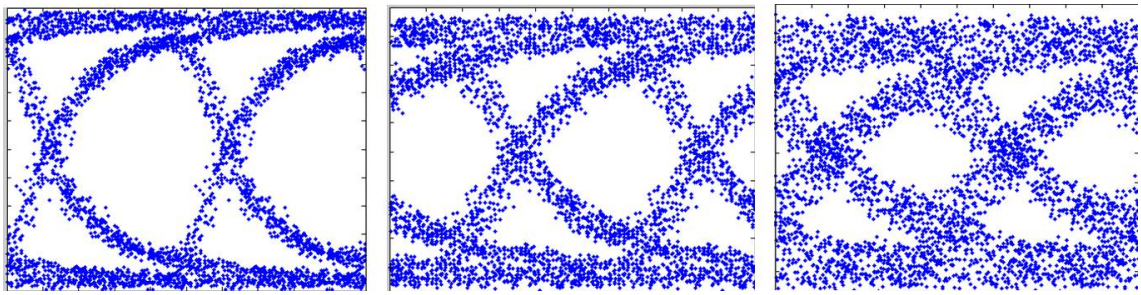


Figure 5-4: Reconstruction results of 2048 samples

In conclusion, if only 2048 points are needed and 6 bit resolution is sufficient for correct operation, that results in half of memory required, a significant area saving while trading off with finished reconstruction quality.

5.3.3 Other Input Signals

The test cases in this work uses one type of channel model derived from a specific type of backplane. Its low pass and relatively clean jitter characteristics make the eye diagram quite regular and predictable. Different input signals are yet to be tested for reconstruction. Other link models can be used and even equalizer outputs should be tested.

The reconstruction algorithm relies on the eye opening and its location. There could be eye shapes in which ringing occurs and multiple maximum opening can happen within a single period. Current algorithm guarantees that the first maximum location is used, and seems to work quite well.

The effect of the low pass filter in finding eye opening might also play a role. For simplicity, a running average filter is implemented, but a more complicated filter can be used aiding the correctness of finding eye opening location with more computation power. Again, there seems to be no best option, but rather always a trade-off between different aspects in the system.

5.3.4 External Control Features

The core of the system is presented in this work, but there is room for expansion with features that allow users to have further control externally. The software interface that communicates with this SoC is the other part of the story – the simplest control would be just an interface to send a start signal and when done reads back all the data to be plotted in Matlab or other numerical software. Extra possible features can include an initial estimation for lambda provided by user when the input data frequency is known, separate control to different reconstruction stages for debugging purposes, further image processing to extrapolate more eye information, etc.

There could be many more possibilities that can be appended onto the core design in this thesis. The method proves to be cost effective in simulation and analysis, and we look forward to further developments to fully carry this SoC into a product after making important trade-off decisions and user specifications.

Bibliography

[1] G. Moustakides, O. A. Frederic Cerou, and L. Noirie., "eye diagram reconstruction using asynchronous imperfect sampling, application to ber estimation for fiber-optic communication systemsl," in *European Signal Processing Conference - EUSIPCO*, 2002.

[2] E. Mobilon, M. de Barros, and A. Lopes, "Low cost eye diagram reconstruction and morphological analysis for optical network performance monitoring using digital signal processing techniques," in *Telecommunications Symposium, 2006 International*, pp. 643 –646, sept. 2006.

[3] L. Noirie, F. Cerou, G. Moustakides, O. Audouin, and P. Peloso, "New transparent optical monitoring of the eye and ber using asynchronous under-sampling of the signal," in *Optical Communication, 2002. ECOC 2002. 28th European Conference on*, vol. 5, pp. 1 –2, sept. 2002.

[4] H. Choi, A. Gomes, and A. Chatterjee, "Signal acquisition of high-speed periodic signals using incoherent sub-sampling and back-end signal reconstruction algorithms," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, pp. 1125 –1135, july 2011.

[5] G. Huang and P. Lin, "A fast bootstrapped switch for high-speed high-resolution a/d converter," in *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*, pp. 382 –385, dec. 2010.

[6] R. Inti, W. Yin, A. Elshazly, N. Sasidhar, and P. Hanumolu, "A 0.5-to-2.5gb/s reference-less half-rate digital cdr with unlimited frequency acquisition range and improved input duty-cycle error tolerance," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pp. 438 –450, feb. 2011.

[7] A. Hajimiri, S. Limotyrakis, and T. Lee, "Jitter and phase noise in ring oscillators," *Solid-State Circuits, IEEE Journal of*, vol. 34, pp. 790 –804, jun 1999.

[8] P. Heydari and R. Mohanavelu, "Design of ultrahigh-speed low-voltage cmos cml buffers and latches," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, pp. 1081 –1093, oct. 2004.

[9] M. Usama and T. Kwasniewski, "New cml latch structure for high speed prescaler design," in *Electrical and Computer Engineering, 2004. Canadian Conference on*, vol. 4, pp. 1915 – 1918 Vol.4, may 2004.

[10] R. Mohanavelu and P. Heydari, "A novel ultra high-speed flip-flop-based frequency divider," in *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, vol. 4, pp. IV – 169–72 Vol.4, may 2004.

[11] J. McNeill, "Jitter in ring oscillators," in *Circuits and Systems, 1994. ISCAS '94., 1994 IEEE International Symposium on*, vol. 6, pp. 201 –204 vol.6, may-2 jun 1994.

[12] A. Demir, "Computing timing jitter from phase noise spectra for oscillators and phase-locked loops with white and noise," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, pp. 1869 –1884, sept. 2006.

[13] F. Herzel and B. Razavi, "A study of oscillator jitter due to supply and substrate noise," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 46, pp. 56 –62, jan 1999.

[14] L. Brooks and H.-S. Lee, "A 12b, 50 ms/s, fully differential zero-crossing based pipelined adc," *Solid-State Circuits, IEEE Journal of*, vol. 44, pp. 3329 –3343, dec. 2009.

[15] Y. Borokhovych, H. Gustat, B. Tillack, B. Heinemann, Y. Lu, W.-M. Kuo, X. Li, R. Krithivasan, and J. Cressler, "A low-power, 10gs/s track-and-hold amplifier in sige bicmos technology," in *Solid-State Circuits Conference, 2005. ESSCIRC 2005. Proceedings of the 31st European*, pp. 263 – 266, sept. 2005.

[16] L. Brooks and H.-S. Lee, "A zero-crossing-based 8-bit 200 ms/s pipelined adc," *Solid-State Circuits, IEEE Journal of*, vol. 42, pp. 2677 –2687, dec. 2007.