50-250MHZ ΔΣ DLL FOR CLOCK SYNCHRONIZATION

CHENG SAN JEOW
(B. Eng.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2011

ABSTRACT

With the advent of high data rate wireline applications in microprocessors and memory integrated circuits (ICs), clock skew becomes a significant portion of the overall timing margin issue in system design. A variable delay line or a delay locked loop (DLL) is often used for flexible timing control not only in source-synchronous serial interfaces but also in clock-and-data recovery systems. The conventional analog delay line, however, suffers from process, voltage and temperature (PVT) variations, and calibration of the analog delay line takes substantial design effort. A digitally controlled delay line is preferred due to its better testability and robust characteristics. Although the semi-digital or all-digital DLL may have more robust delay control, achieving fine timing resolution comparable to that of an analog delay line is still challenging due to minimum achievable delay resolution posed by technology limitations.

Spurred by demands of fine tuning resolution, low jitter performance and operation robustness, the aim of the research work in this thesis is to address these issues in a digitally controlled DLL. A delta sigma (ΔΣ) modulator based DLL architecture for clock synchronization application is designed and fabricated in 0.35μm CMOS technology as a proof of concept for demonstrating the fine timing resolution and low jitter performance achievable by such architecture. By incorporating a ΔΣ modulator in the DLL, it can have a fractional step delay of 15ps and can operate from 50MHz to 250MHz. Clock synchronization is straightforward and occurs in 2 phases – coarse tuning and fine tuning phases. In fine tuning, a successive approximation (SA) method is used to quickly shift the output clock near to the input clock. It draws about 6.9mA from 3V supply at 200MHz.

Unlike other existing ΔΣ DLL designs, the proposed DLL makes use of the ΔΣ modulator in the feedback path rather than at the input, which enables it to eliminate the additional multi-phase generator (MPG), and hence simplifies the architecture. Besides the simplification in architecture, it also has 2 novel features, a 2^{nd} order filter whose 2 poles can be adaptively adjusted and a unique anti-harmonic detector.

Through simplification in the structure and noise shaping contribution from the ΔΣ modulator, it exhibits a low rms jitter of 2.137ps. Hence, the proposed digitally controlled DLL is straightforward and combines digital architecture robustness with fine tuning similar to analog designs.

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Assistant Professor Heng Chun Huat and Dr Zheng Yuanjin for their willingness to take me as their student and especially Dr Heng who has shown great patience and guidance to a student who is much older than his peers and trying his best to fit back into school.

I would like to dedicate this thesis to my family especially to my wife, Siewhwi, for bringing the bacon home while I slog at school and the mental support she has given me. I also dedicate it to my son, Shervin, who has provided me with laughter and stress relief when I'm burdened from all the school and project work.

I would also like to thank the staff from VLSI and Signal Processing for the support they provided to make this project possible and last but not least, I would like to thank my colleagues in the same lab, with whom I held discussions with, in order to gain valuable insights to enable me to complete this project.

Without the necessary funding, this project would not have been made possible. Many thanks to AcRF R-263-000-317-112 for funding this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

DLL	Delay locked loop
PVT	Process, voltage and temperature
$\Delta\Sigma$	Delta-sigma
$\Delta\Sigma$M	Delta-sigma modulator
SA	Successive approximation
MPG	Multi-phase generator
FIR	Finite impulse response
OSR	Over-sampling ratio
PLL	Phase locked loop
CMOS	Complementary metal oxide semiconductor
CP	Charge pump
PD	Phase detector
VCDL	Voltage controlled delay line
DCDL	Digital controlled delay line
FSM	Finite state machine
ϕ_{ref}	Reference phase
ϕ_{DLL}	Feedback phase from delay line
ϕ_{in}	Input phase
ϕ_{out}	Output phase
I_{CP}	Charge pump current
K_{DL}	Delay gain of delay line
V_F	Loop filter control voltage
$F(s)$	Loop filter transfer function
I_D	MOS transistor current
g_m	Transistor transconductance
V_{th}	Transistor threshold voltage
RC	Resistor capacitance time constant
ω_p	Pole frequency
PM	Phase margin
AHD	Anti-harmonic lock detector
K	Delta-sigma modulator control word
CPD	Coarse loop phase detector
DUT	Device under test
MUX	Multiplexer

CHAPTER 1. INTRODUCTION

Delay-Locked-Loop (DLL) is gaining a foothold recently for applications in clock synchronization, multi-phase clock generation and data recovery [1-4]. It offers better jitter performance and unconditional stability compared to Phase-Locked-Loop (PLL) [5] due to the fact that there is no cycle-to-cycle jitter accumulation. Its circuit is also much simpler and can be easily implemented in digital CMOS process. DLL is also easier to achieve stability, making it more attractive for timing synchronization. The various architectures are discussed next.

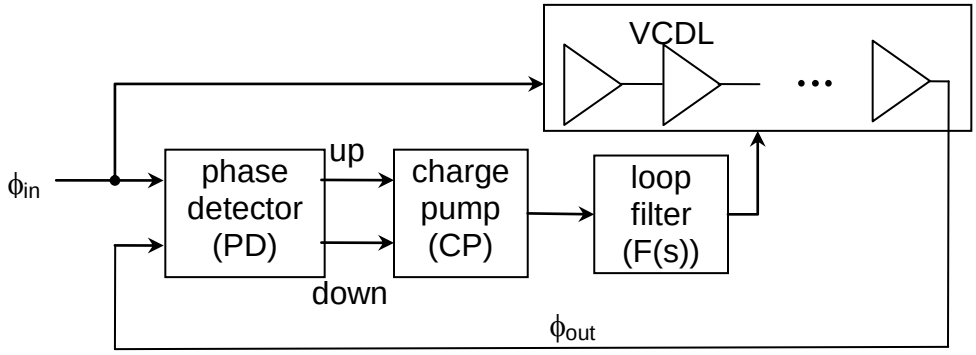


Figure 1.1: Analog DLL

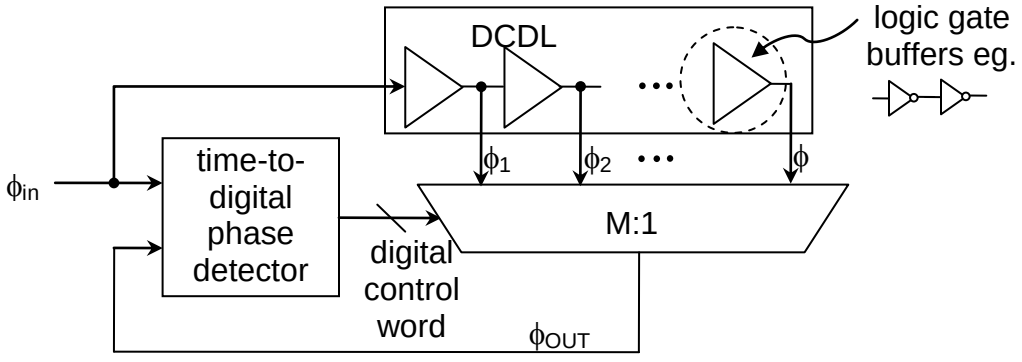


Figure 1.2: Digital DLL

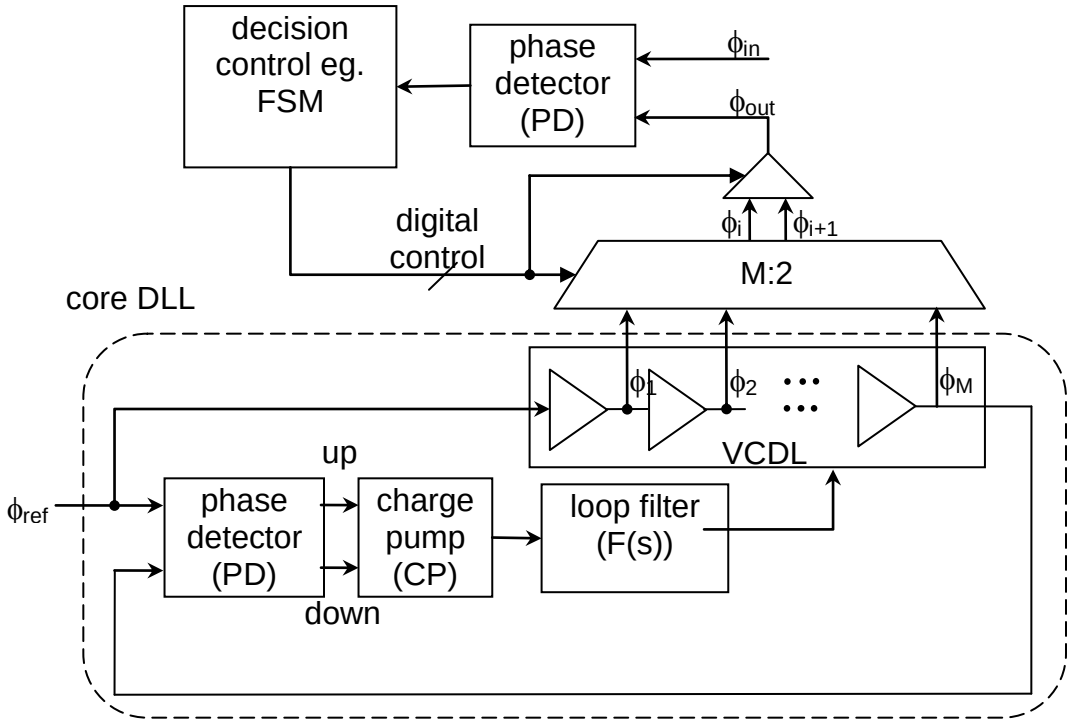


Figure 1.3: Semi-digital DLL [8-9]

1.1 General categories of DLL

Analog DLLs are usually distinguished by its analog building blocks like charge pump (CP), voltage controlled delay line (VCDL), loop filter (F(s)) as illustrated in Fig. 1.1. They provide the finest tuning resolution as any phase mismatch between ϕ_{in} and ϕ_{out} is translated proportionally to the loop filter control voltage that tunes the delay of each delay cell. Although it has continuous delay variation and good jitter performance due to feedback, it suffers from limited phase range [6]. On the other hand, digital DLL is most amenable to digital CMOS process as they only require basic building blocks like flip-flops and logic gates. Characterized by its usage of logic gates for the delay line for digital tuning [7] as shown in Fig. 1.2, digital DLL often requires most advanced CMOS technology to push down the delay resolution

achievable by single gate delay. In addition, large delay range is also needed to guarantee the full phase coverage. Due to its open loop nature, digital DLL can achieve much faster locking by employing simple time-to-digital converter (TDC). However, there will always be residual phase error limited by the achievable delay resolution. Semi-digital DLL [8-9] is proposed to overcome the limitations faced by digital and analog DLL. The need of additional phase interpolator [8-9] to interpolate between the 2 adjacent phases (Fig. 1.4), worsen the overall jitter performance.

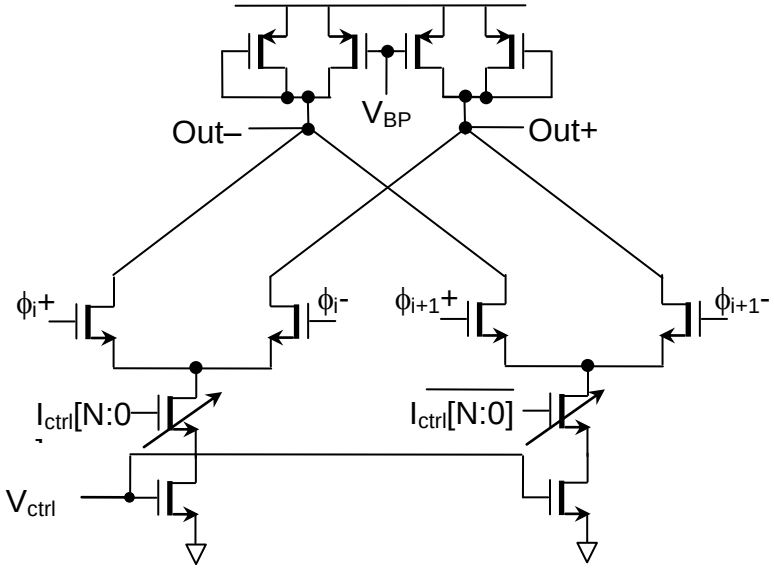


Figure 1.4: Typical analog phase interpolator

Spurred by demands of low jitter and fine timing resolution, we look at alternative architectures for DLL that encompass the benefits of analog and digital DLLs. Recently, delta-sigma (ΔΣ) DLL has emerged [9-10] as a strong contender to achieve fine resolution in pico-second range as well as good jitter performance, without the above mentioned shortfalls. In this thesis, we will examine the issues faced by existing ΔΣ DLL and propose an alternative ΔΣ DLL architecture. We will also discuss some circuit blocks innovation to facilitate the proper functioning of the proposed architecture. The ΔΣ DLL architecture is incorporated in a clock synchronization application to show its achievable timing resolution and jitter performance.

1.2 Organization of thesis

The thesis is organized in the following format. In chapter 2, we will examine existing DLL architectures in detail. This is then followed by the proposed DLL architecture and its utilization in clock synchronization architecture in chapter 3. The modeling aspects of the DLL and clock synchronization architecture are covered in chapter 4 while CMOS implementation is covered in chapter 5. We discuss the simulation and measurement results in chapter 6 and conclude in chapter 7.

CHAPTER 2. PRIOR ΔΣ DLL ARCHITECTURES

2.1 Clock Synchronization

Typical clock synchronization operation is illustrated in Fig. 2.1. It usually involves phase locking an input clock (ϕ_{in}) to a selected output phase (ϕ_{out}) from a multi-phase generator (MPG). As ϕ_{out} is generated from a low jitter reference source (ϕ_{ref}), the resulting architecture can achieve very good jitter performance. Depending on the signal generated from phase control block, different architectures will result. Fully digital delay control will give rise to digital DLL. Analog delay control will result in analog DLL. Semi-digital DLL will have mixed types of delay control signals.

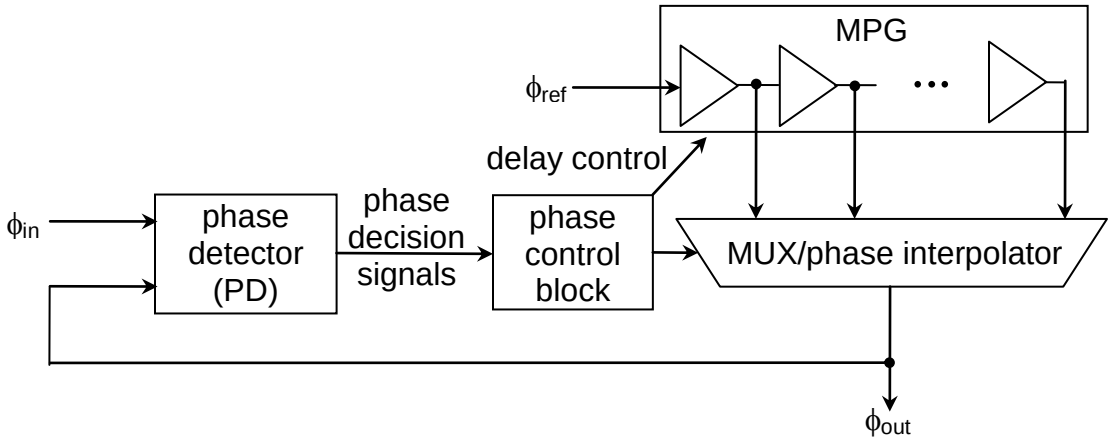


Figure 2.1: Typical DLL architecture for clock synchronization

2.2 General ΔΣ DLL architecture

In contrast to semi-digital DLL where analog phase interpolator is used to obtain fine phase resolution, ΔΣ DLL employs a ΔΣ modulator to randomly select the input phase (ϕ_{ref}) from a fixed number of adjacent phases (ϕ_{i-1}, ϕ_i, ϕ_{i+1}, etc.) from a MPG to produce the desired ϕ_{ref} as shown in Fig. 2.2. The resulting ϕ_{ref} will then be compared with the feedback phase (ϕ_{DLL}) from the delay cell through the phase detector (PD). Based on a given control word, *K*, the ΔΣ modulator will generate a phase selection sequence which will result in an average phase which lies between the given adjacent phases. The theory is very much similar to a ΔΣ digital-to-analog converter (DAC), where the interpolation occurs between a fixed number of output voltage levels, and analog output voltage with fine resolution can be obtained on average after filtering. The random phase error after filtering by the loop filter (*F(s)*) will then produce the desired control voltage to achieve the delay that will match the mean input phase. Not only the resultant average input phase can be tuned very finely depending on the bit resolution of the ΔΣ modulator, by the virtue of pushing the in-band noise towards the higher frequencies through ΔΣ modulator and the effective filtering through the loop filter, good jitter performance can also be achieved. It should be pointed out that the resulting ΔΣ DLL can function as a digital-to-time converter.

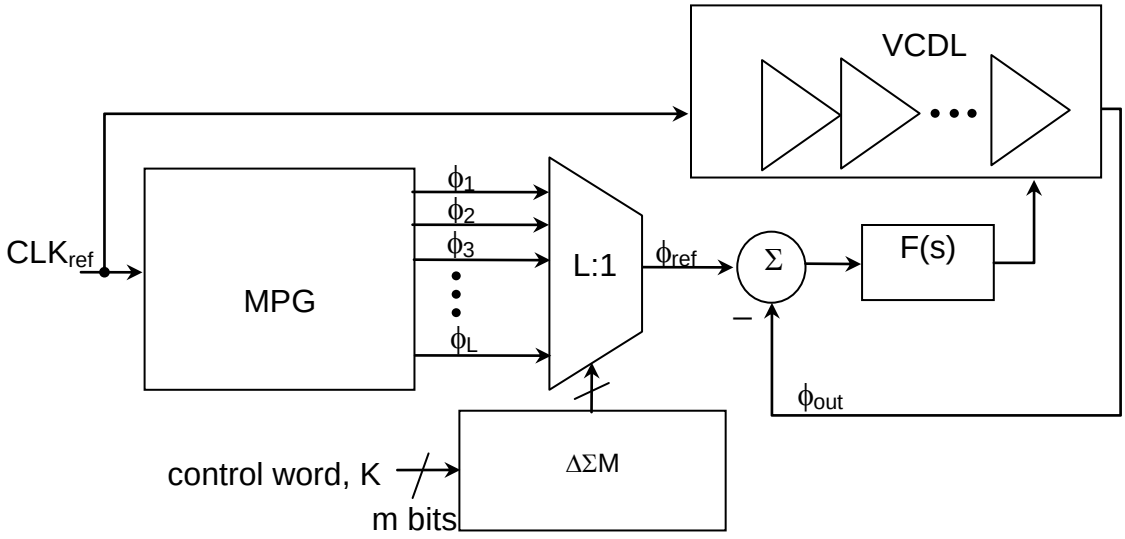


Figure 2.2: Existing ΔΣ DLL architecture

2.2.1 ΔΣ DLL employing PLL as MPG

Both [10] and [11] have implemented the ΔΣ DLL architecture shown in Fig. 2.3 and they mainly differ in the way of implementing the MPG. In [10], a PLL with a multi-phase ring oscillator as highlighted in Fig. 2.2 is employed as the MPG. The 3 most significant bits (MSB) of the 14 bits control word (*K*) are used to select three adjacent phases ($\pm 45^o$ and 0^o) out of the eight phases. The remaining 11 bits are then input to ΔΣM to randomize the three selected adjacent phases to obtain fine phase step. This structure is simple and elegant, directly controlling the phase delay with a fixed control word. However, having a PLL as a MPG incurs more power and area. Although the ΔΣ DLL itself do not contribute much additive jitter, the overall jitter performance is already handicapped by the PLL as the noisier multi-phase ring oscillator is now employed as the input to the DLL rather than the low jitter reference crystal clock.

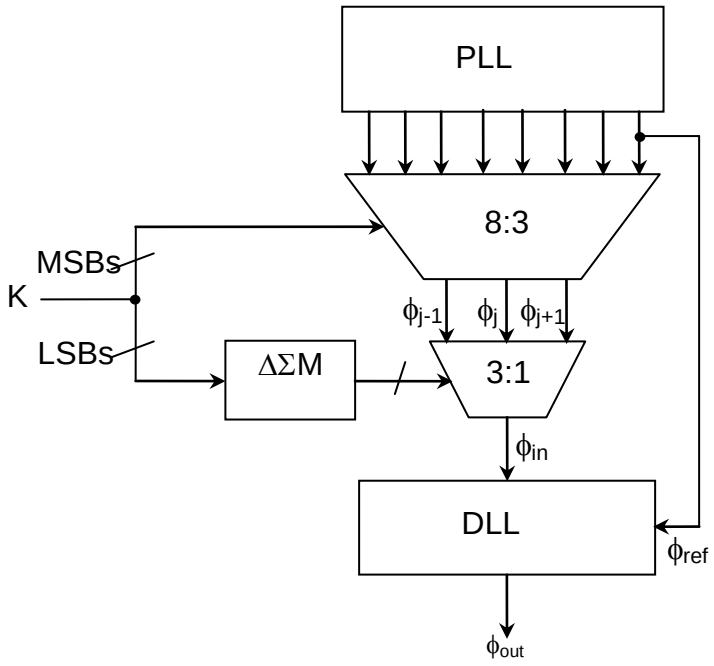


Figure 2.3: ΔΣ DLL employing a ring oscillator PLL as a MPG

2.2.2 Low OSR ΔΣ DLL with self referenced multiphase generation

In another structure [11] as illustrated by Fig. 2.4, a ΔΣ modulator is employed to randomize the multi-phases from a MPG in the similar manner as [10] and eventually phase lock to *SIG*. Shift registers and a clock divider are employed to form the MPG as shown in Fig. 2.5. The desired multi-phases ($ref_{\pi/16,0,-\pi/16,-\pi/8}$) are obtained by dividing down a very high frequency input clock (*CKI*) by 32 and then shifting the divided clock (*DCK*) by the same *CKI*. The resolution of the ΔΣ modulator must be very high in order to provide very small delay tuning for *CKO* due to the larger clock period when phase comparison is performed in a much lower frequency domain relative to the reference clock sources. This also indicates a slower lock time due to a smaller loop bandwidth.

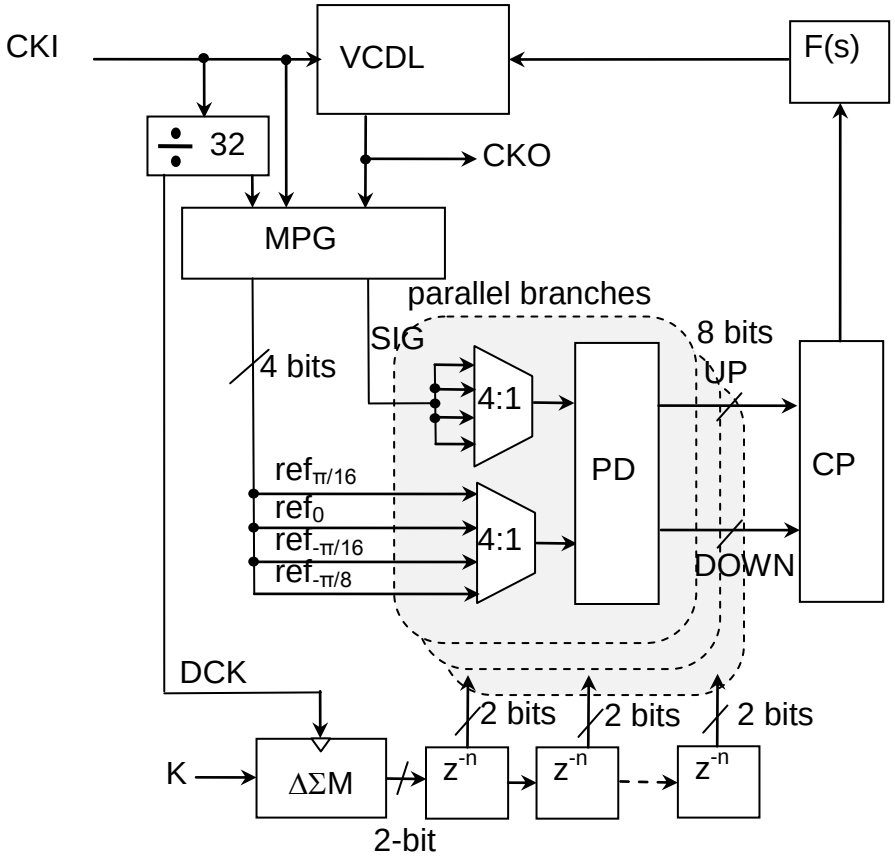


Figure 2.4: ΔΣ DLL with self referenced multiphase generator as a MPG

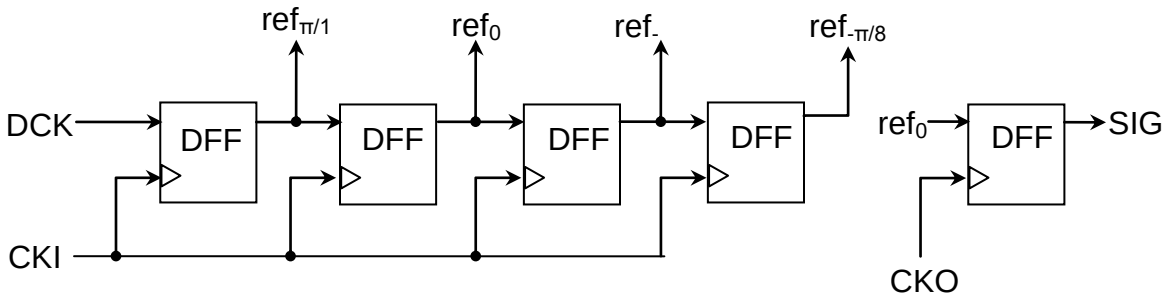


Figure 2.5: Shift register as a MPG

Although re-sampling the divided clock signal with the input clock is advantageous as it reduces the jitter of its multi-phase outputs, this limits the achievable phase quantization to one input clock period and lowers the over-sampling ratio (OSR). To compensate for these limitations, a technique commonly known as finite impulse response (FIR) embedding is used. Parallelism in structure by employing multiple PDs with a multi-bit input charge pump serving as a summer for the multiple paths provides the required averaging to make up for the reduced OSR. The dithered input reference phase of each parallel branch is controlled by a delayed version of the ΔΣ modulator output. The parallel structure of PDs, coupled with the delayed control, forms an FIR filter in the analog domain. This FIR filter helps to reduce the out-of-band quantization noise caused by the large quantization step, so as to achieve better jitter performance. The application of the embedded FIR technique complicates the overall architecture (eg. issues of path mismatch etc) and incurs area penalty.

While ΔΣ DLLs look promising in overcoming the limitation of digital DLL phase resolution, we look to improve on the existing ΔΣ DLLs structures in terms of less complexity and better jitter performance. We shall discuss our proposed ΔΣ DLL in the next chapter.

CHAPTER 3. PROPOSED ARCHITECTURE

3.1 Proposed dithered feedback path ΔΣ DLL

Given that the additional jitter from the PLL in [10] and the complexity and lower OSR introduced by [11] are not preferred, we try to reinvent the architecture to circumvent these undesired features. The proposed ΔΣ DLL is shown in Fig. 3.1. Instead of employing the ΔΣ modulator to randomly select the input phase, we employ the ΔΣ modulator at the feedback path to randomly select the feedback delay. In this manner, we eliminate the need of an additional multi-phase generator by making full use of the multi-phase output in the voltage controlled delay line (VCDL). This is the clear advantage when compared to previous architectures since we already make use of the multi-phases readily available in the VCDL without the need of additional MPG. For the proposed ΔΣ DLL, the phase quantization can be kept as a fraction of one input clock period, thus minimizing quantization noise. Together with a clean crystal reference clock employed at the input, both features can help in improving the jitter performance.

3.1.1 General operation of proposed ΔΣ DLL

The ΔΣ modulator provides the dithered phase delay, ϕ_{DLL}, from delay taps ϕ_8 to ϕ_{11} of the VCDL based on a given control word, *K*. The chosen feedback delay is then compared with the reference phase through the PD. The filtered error from the PD will generate a controlled voltage that will produce the delay to match the reference phase. More details on tuning are covered in the operation of the finite state machine (FSM) in chapter 5.

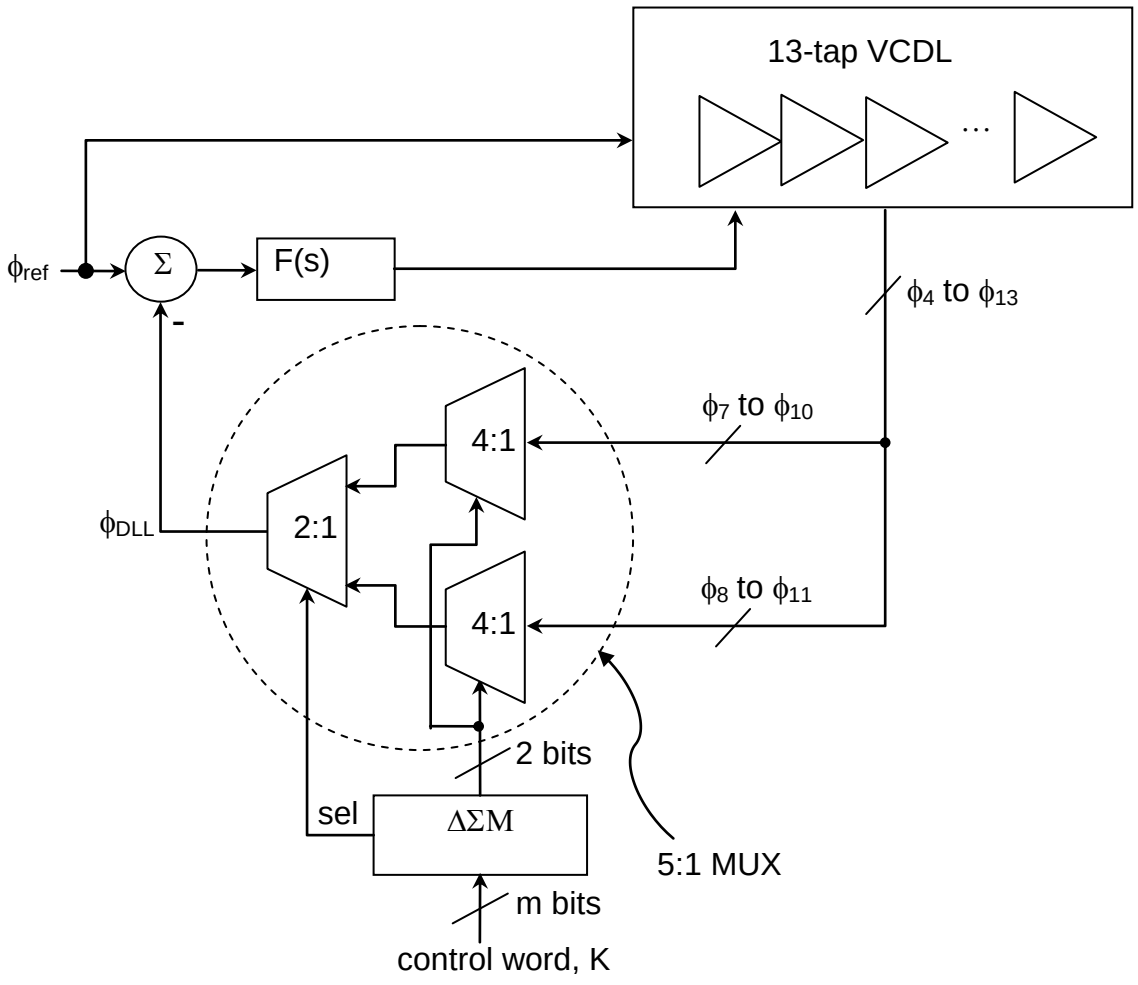


Figure 3.1: Proposed ΔΣ DLL

3.1.2 Structural description of proposed ΔΣ DLL

In the proposed DLL, thirteen delay cells form the delay line, from which ten of its outputs (ϕ_4 to ϕ_{13}) are used to cover all possible phases of the entire clock. The feedback delay is tapped from the 7th to 11th outputs (ϕ_7 to ϕ_{11}) of the delay line. The ΔΣ modulator produces a two bit output that will randomly select four phases. Either 7^{th} to 10th taps or 8^{th} to 11th taps can be chosen by the ΔΣ modulator. The provision of two delay tap groups enables the DLL to cover all the phases required for the clock synchronization in fine step and will be covered in detail later. In this design, a passive 2^{nd} order loop filter with adaptive pole tuning and a 1st order modulator is chosen to complement each other. The design of these blocks will be discussed in chapter 5. It should be pointed out that the number of bits and thus time step resolution are ultimately limited by the achievable clocking speed of digital circuitry implemented in the ΔΣ modulator. Fortunately, this implies that better time step resolution can be expected with the down scaling of transistor size for more advanced CMOS technology. Therefore, we adopt programmable bit resolution for the ΔΣ modulator to maintain similar time step resolution across different input clock frequencies. At lower input clock frequency, the ΔΣ modulator can tolerate more adder delay and larger *m* can thus be used. In this design, *m* is made programmable from 5 to 7 bits to achieve relatively consistent time step resolution of roughly 15ps throughout the input clock frequency range of 50MHz to 250MHz.

3.1.3 Non-linearity in delay tuning

The delay per cell generated by the proposed architecture can be estimated as follows:

$$T_{delay} = \frac{T_{clk}}{N_{average}} = \frac{T_{clk}}{N + \frac{K}{2^m}}, \tag{3.1}$$

where N can be either 8 or 9 depending on the selected delay tap groups, K is the input control word to the modulator, m is the resolution of the ΔΣ modulator, resulting in the average number of delay taps, $N_{average}$, having values from 8 to 10. T_{clk} is the period of the input clock. The time step resolution per cell is determined by the difference between the two consecutive delays and is defined as follows with each successive decrement of the control word, K:

$$T_{step} = T_{clk}\left(\frac{1}{N + \frac{K-1}{2^m}} - \frac{1}{N + \frac{K}{2^m}}\right). \tag{3.2}$$

Equation (3.2) indicates that the time step resolution is input dependent. The resulting digital-to-time transfer characteristic is therefore non-linear. This is totally different from [10-11] where a linear transfer characteristic can be observed. However, the resolution of the ΔΣ modulator (m) can always be chosen to be large enough such that the DLL can cover the desired phase range even with the non-linear transfer characteristic. The ΔΣ modulator not only determines the time step resolution, it also affects the overall jitter performance.

3.2 Clock synchronization architecture using the proposed ΔΣ DLL

The complete clock synchronization architecture employing proposed ΔΣ DLL is shown in Fig. 3.2. The clock synchronization occurs in two steps. First, the core DLL loop is initialized to give a fixed T_{delay}. The resulting multiphases (ϕ_4 to ϕ_{13}) are then compared with the incoming phase (ϕ_{in}) to determine the closest phase through a coarse phase detector (CPD). The CPD will generate Up/Down or Hold signal depending on the error between ϕ_{in} and ϕ_{out}. The FSM will then select one of the phases from ϕ_4 to ϕ_{13} that is closest to the input phase ϕ_{in}. Once the closest phase has been identified, the algorithm will move on to the second step, where the FSM will start changing the input control word K to fine tune the T_{delay} of the core ΔΣ DLL. This will move the selected phase edge (ϕ_{out}) from the first step closer to the incoming clock edge (ϕ_{in}). To speed up the second step, successive approximation (SA) technique is employed to obtain the correct input control word K. Therefore, it only requires m steps to cycle through all the possible K values which will produce the T_{delay} that closely match to the input phase (ϕ_{in}). Compared to the semi-digital DLL proposed in [7-8], this architecture eliminates the additional phase interpolator by simply tuning the DLL phase edges directly through the ΔΣ modulator. This not only simplifies the design but also eliminates possible additional jitter source. Fig. 3.3 exemplifies the above-mentioned delay matching concept.

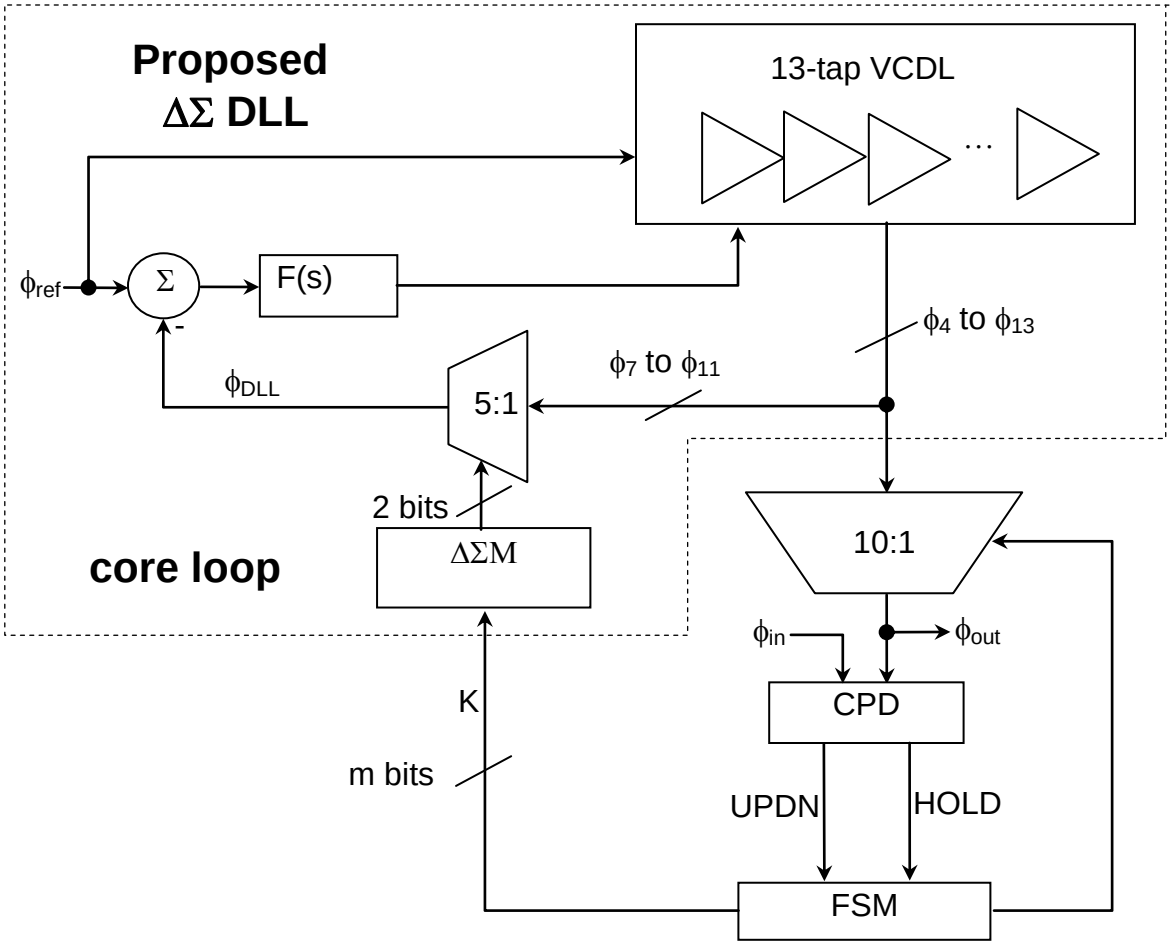


Figure 3.2: Clock synchronization architecture with proposed ΔΣ DLL

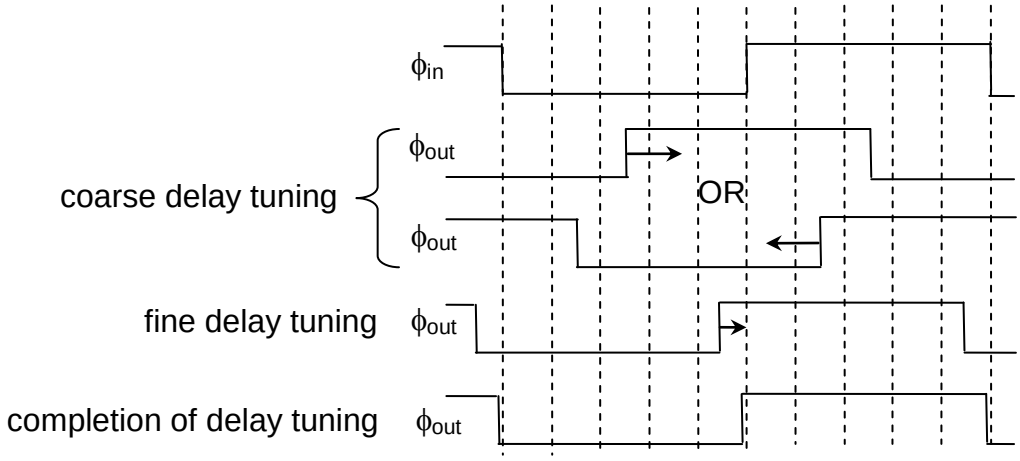


Figure 3.3: Clock synchronization operation

3.3 Design Considerations

Like most other DLLs that suffer from static mismatch issues, load and path matching must be taken into account while designing the DLL. Design for full phase coverage is also needed for the proper operation of a clock synchronization system.

3.3.1 Mismatch Consideration

Fig. 3.2 presents a simplified view of the proposed architecture. In fact, two major sources of mismatches will impact the performance and need to be addressed carefully. Firstly, the delay cell mismatch needs to be minimized. As the different phase taps constitute the whole phase range, any mismatch among the delay cells would give rise to output phase inaccuracy (see Fig. 3.4).

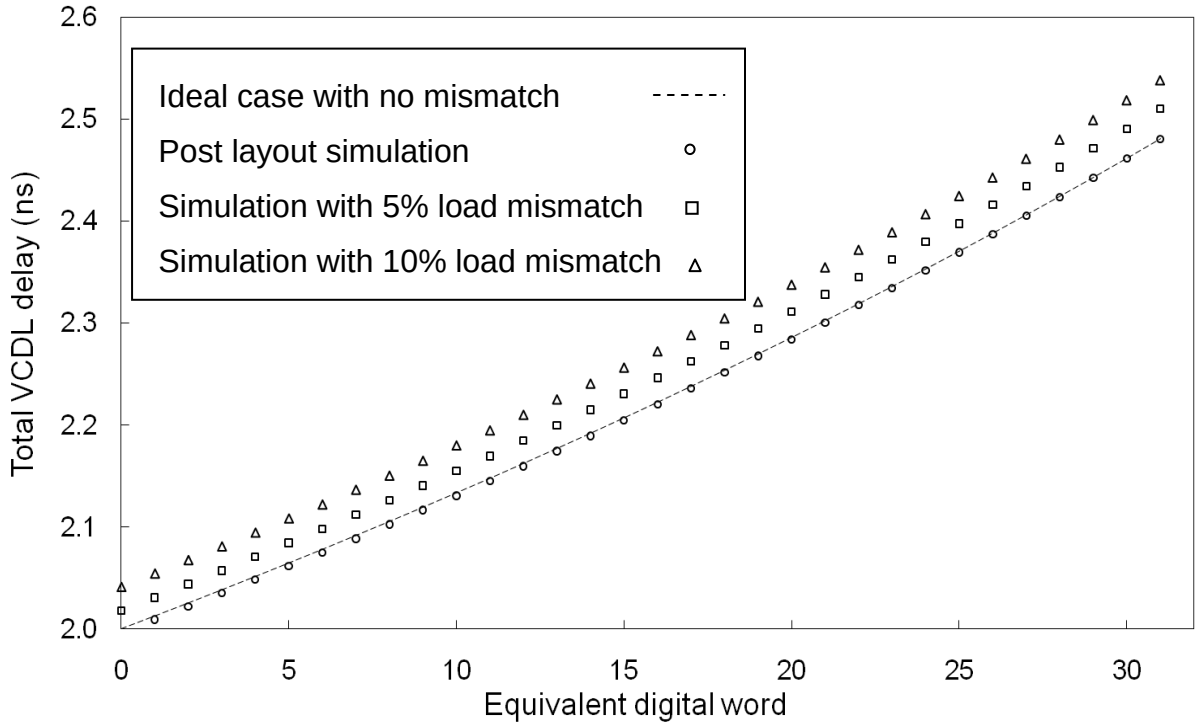


Figure 3.4: Simulated transfer characteristic of ΔΣ DLL from output of 4th delay cell with and without load mismatch

This is a common problem faced by all DLL with multiple phase taps [8, 10-11], and is solved by ensuring equal loading seen by all the delay cells. Similar approach has been adopted here. Although the feedback path only requires 5-to-1 multiplexer (ϕ_7 to ϕ_{11}) and the output phase selection only requires 10-to-1 multiplexer (ϕ_4 to ϕ_{13}), identical 13-to-1 multiplexers are employed for both of them to ensure equal loading seen by all delay cells as illustrated in Fig. 3.5. In addition, dummy delay cell is also added to the output of 13th delay cell for better load matching. Secondly, the additional multiplexer employed in the feedback path introduces additional multiplexer delay (t_{mux}).

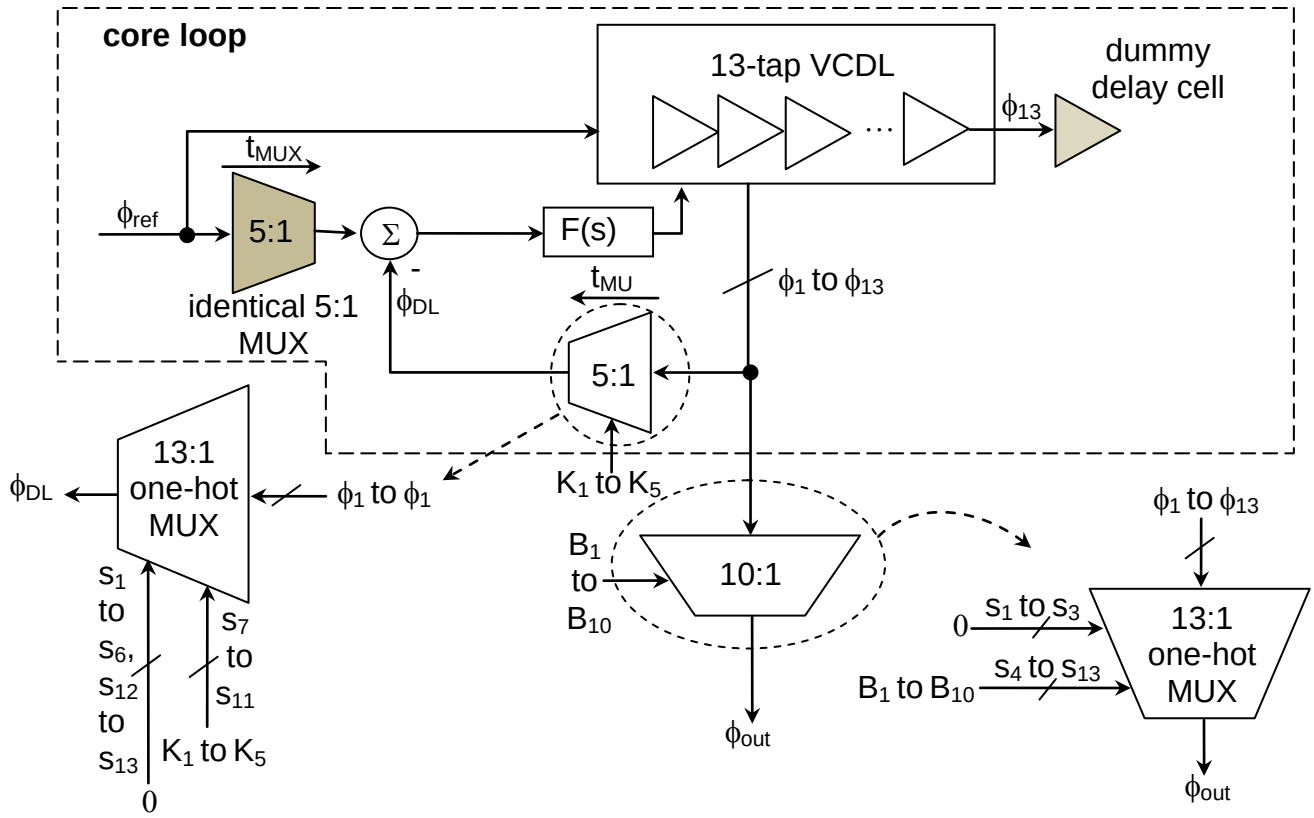


Figure 3.5: Load mismatch and feedback path mismatch consideration

The resulting T_{delay} in (1) will now become

$$T_{delay} = \frac{T_{clk} - t_{MUX}}{N + \frac{K}{2^m}} \quad . \tag{3.3}$$

Therefore T_{delay} will not be accurately defined due to the process dependent t_{mux}. This issue is resolved by employing identical multiplexer at the input path as shown in Fig. 3.5. This will eliminate the additional process dependent t_{mux} introduced in (3). It should be pointed out that this issue is also not unique in our proposed architecture. In fact, careful examination of Fig. 2.3 and Fig. 2.4 reveals similar problem faced by other ΔΣ DLL [10-11] due to the additional process dependent delay introduced by MPG and multiplexer. Similar technique has also been employed in [10-11] to eliminate this phase inaccuracy.

To validate the impact of the mismatches on the output delay characteristics, delay characteristic of Fig. 3.4 with extracted post-layout delay is also shown. As illustrated, with the employed technique to minimize delay mismatch, there is not much deviation of the resulting delay characteristic compared to the one with ideal matching delay. However, when the delay mismatch deteriorates to 5~10%, noticeable static phase error will result. Nevertheless, for clock synchronization application proposed in this thesis, the proposed algorithm will automatically adjust the ΔΣ DLL input to compensate for this phase offset as long as the proposed DLL exhibit continuous phase range coverage. For digital-to-phase converter application, this static phase error can be calibrated with additional time-to-digital converter (TDC), which can be easily implemented using shift registers.

3.3.2 Phase range consideration

The first three delay taps are not used during the coarse tuning step because of the phase range issue. When $N_{average}$ is changed continuously from 10 to 8 through the control word *K*, each delay cell will experience different phase variation as indicated by $\Delta\phi_i$ in Fig. 3.6. In general, the later the delay cell being placed in the delay line, the larger the phase variation it will encounter, i.e. $\Delta\phi_i > \Delta\phi_k$ if $i>k$. By modifying the ΔΣ modulator input during the fine tuning step, the combined phase variation from ϕ_4 to ϕ_{13} will cover the entire phase (2π) of the clock period continuously, as shown by the darkly shaded region. However, if ϕ_1 to ϕ_9 was chosen instead, the combined phase variation from ϕ_1 to ϕ_9 will result in discontinuous phase coverage of the entire clock period as illustrated by the broken lightly shaded rectangle region. Equation (2) and Fig. 3.6 clearly indicate that the phase variation is limited by the earlier delay tap. To ensure continuity in the phase coverage, the following relationship can be derived:

$$T_{clk}\left(\frac{i+1}{N_{\min}}-\frac{i+1}{N_{\max}}\right)\geq\frac{T_{clk}}{N_{\min}}$$

$$\Rightarrow i\geq\frac{N_{\max}}{N_{\max}-N_{\min}}-1 \quad , \qquad (3.4)$$

where N_{min} *and* N_{max} *are* the minimum and maximum number of delay taps per one clock period and *i* is the earliest delay tap used for the phase synchronization. Equation (4) reveals that the larger the *i* value, the smaller the difference that can be tolerated between N_{max} and N_{min}, and thus the smaller number of delay tap groups needed ($N_{max}-N_{min}+1$). However, larger *i* also means more delay cells along the delay line for full phase coverage, and thus more power and jitter. In

this design, *i*, N_{min} and N_{max} are chosen to be 4, 8 and 10 respectively to optimize between number of delay tap groups, jitter and power.

The other clear advantage of not using the first three delay cells is that it relaxes the design parameter of the delay cell. If the first cell is used for delay tuning, this imposes on the cell a full tuning range of Tclk/10, which is very difficult to design. By using more cells in the untapped section of the delay line, it relaxes this constraint further.

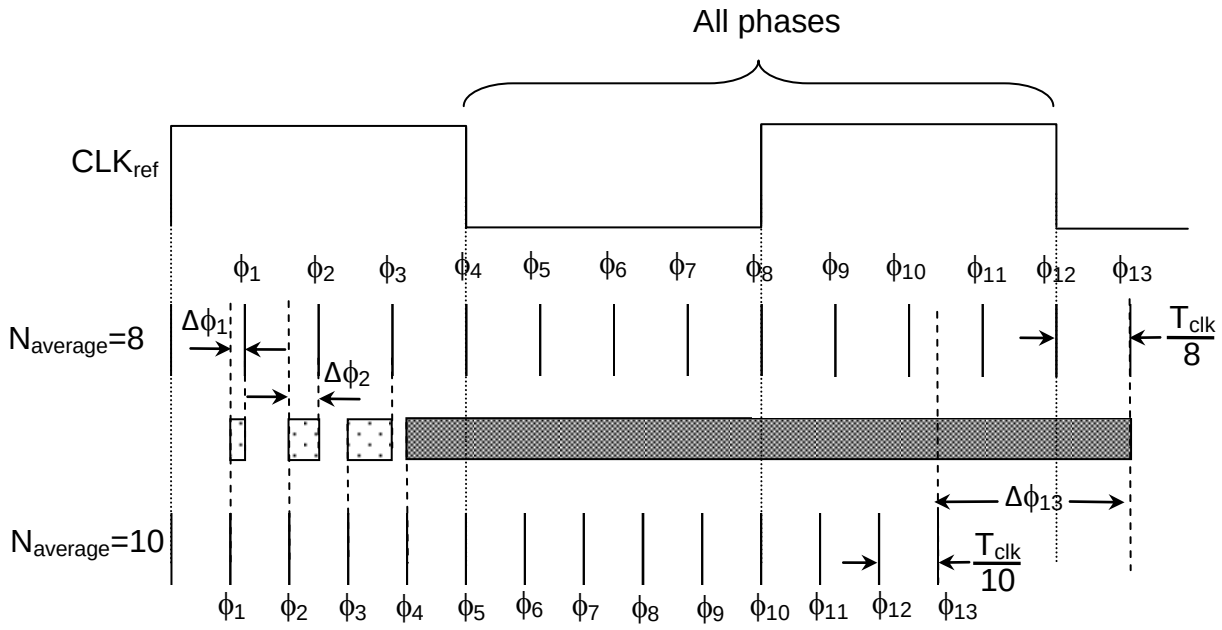


Figure 3.6: Overlapping delays to ensure full clock period coverage

3.4 Conclusion

A new ΔΣ DLL architecture utilizing dithering in its feedback path is presented in this chapter. Its linearity issues is easily mitigated using a higher resolution ΔΣ modulator and hence can be applied to clock synchronization. The common issues of phase mismatch and phase coverage are also tackled. Before continuing to the actual silicon design of the circuit, it is important that modeling and analysis of the entire clock synchronization architecture is performed for a more in depth understanding of the system. This will be covered in the next chapter.

CHAPTER 4. MATLAB MODELING AND LOOP ANALYSIS

4.1 Loop analysis of proposed clock synchronization architecture

In this chapter, modeling and loop analysis of the clock synchronization architecture will be covered in detail. The two most important aspects of loop stability and transient behavior will also be scrutinized based on its defining loop parameters.

4.1.1 Modeling of ΔΣ DLL core loop

The ΔΣ DLL core loop is modeled as a simple feedback loop with its functional blocks merged into a single block with transfer function *H(s)* as shown in Fig. 4.1. *H(s)* can be reduced to a simple representative form consisting of loop parameters by a_0 and a_1:

$$H(s) = \frac{D_{OUT}}{D_{EC}} = \frac{\frac{I_{CP} F_{REF} K_{DL}}{C_1 C_2 R}}{s^2 + \frac{C_1 + C_2}{C_1 C_2 R} s} = \frac{a_0}{s(s + a_1)},$$

(4.1)

using the parameters of each functional block, i.e. charge pump current (I_{CP}), frequency of reference source ϕ_{ref} (F_{ref}), delay gain of VCDL (K_{DL}) and loop filter components (C_1, C_2 and R) based on the filter configuration mentioned in chapter 5. Note that a_0 and a_1 are positive values and the analysis in this section is based on delay and not phase. Each component of delay of signals in the loop are prefixed by '*D*'. e.g. D_{ref}, D_{out} represents the delay values of signals ϕ_{ref} and ϕ_{out} from previous chapters respectively. Quantization noise in the form of delay, $D_{\Delta\Sigma}$, from

the ΔΣ modulator is introduced into the core loop through its' feedback path as shown in Fig. 4.1. Intermediate signals in their delay forms are as shown in the same diagram.

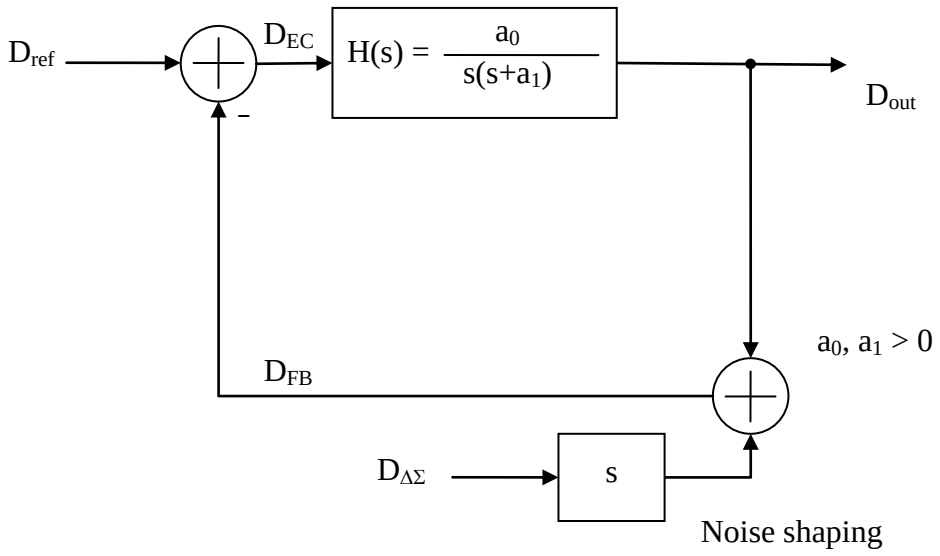


Figure 4.1: Modeling of ΔΣ DLL core loop

4.1.2 Dual loop dynamics of clock synchronization architecture

To model the secondary loop of the proposed architecture, some intuitive approximation is needed. As the secondary loop is fundamentally a digital loop involving FSM, direct incorporation would be difficult and less insightful. Instead, we observe that at steady state, the main objective of secondary loop is to closely match the input and output phase, and produce a phase error (D_{EP}) of zero. According to feedback control theory, a zero error is only possible if integrator controller is employed. Therefore, the secondary loop can be modeled as an integrator with constant a_2 as illustrated in Fig. 4.2. The final linearized model of the clock synchronization system, characterized in terms of delay, is presented in Fig. 4.2.

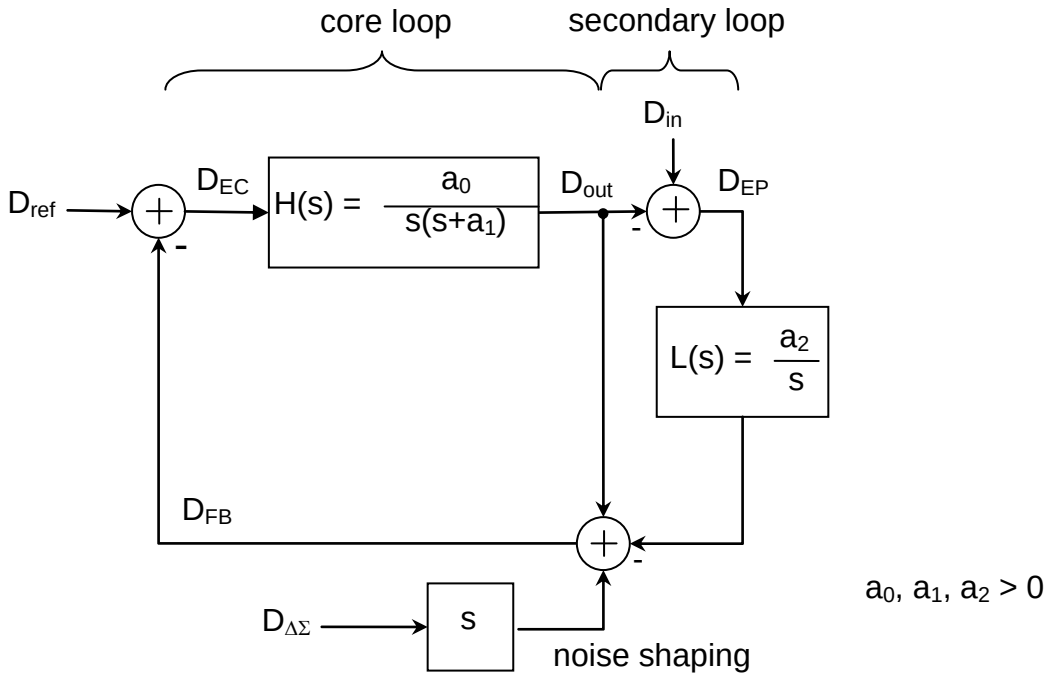


Figure 4.2: Linearized model of clock synchronization system

The effects of applying different values of a_2 with respect to the core loop parameters are studied to provide some intuitive understanding of the design of the peripheral loop in terms of stability and transient response of the delay error, D_{EP}, between ϕ_{out} and ϕ_{in}. The core DLL loop is fundamentally a stable loop. D_{EP} is expected to settle to 0 and ought to remain stable no matter what perturbation is introduced to the loop. The linear model based on Fig. 4.2 is built in Matlab for the verification of the stability analysis and the study of the effects of the core and secondary loop parameters on the transient step response on the phase error D_{EP}.

4.1.2.1 Stability analysis

To study the stability of the dual loop architecture, the close loop transfer functions of D_{EP} with respect to various inputs has to be derived. The close loop transfer functions of D_{EP} with respect to the different delay inputs, D_{in}, D_{ref} and $D_{\Delta\Sigma}$ are found to be:

$$\frac{D_{EP}}{D_{in}} = -\frac{a_0 s}{s^3 + a_1 s^2 + a_0 s + a_0 a_2}, \tag{4.2}$$

$$\frac{D_{EP}}{D_{ref}} = \frac{s^3 + a_1 s^2 + a_0 s}{s^3 + a_1 s^2 + a_0 s + a_0 a_2}, \tag{4.3}$$

and

$$\frac{D_{EP}}{D_{\Delta\Sigma}} = \frac{a_0 s^2}{s^3 + a_1 s^2 + a_0 s + a_0 a_2}, \tag{4.4}$$

respectively. By final value theorem or taking limits of $s \to 0$ of equations (4.2) to (4.4), the responses to a step input disturbance for different inputs are the same. D_{EP} will eventually settle to 0 for all values of a_0, a_1 and a_2 if given enough time. However, in order to ensure stability of the system regardless of any input responses, the close loop poles of transfer functions (4.2) to (4.4) must be located in the left half complex plane. Since all 3 close loop transfer functions have the same characteristic equation, which determines the location of the poles, the Routh array is formed:

$$\begin{array}{c|c|c} s^3 & 1 & a_0 \\ s^2 & a_1 & a_0 a_2 \\ s^1 & \dfrac{a_1 a_0 - a_0 a_2}{a_1} & \\ s^0 & a_0 a_2 & \end{array} \tag{4.5}$$

By Routh Hurwitz theorem for stability [12], elements in the first column in the Routh array must have the same sign resulting in the following condition that be satisfied:

$$\frac{a_1 a_0 - a_0 a_2}{a_1} > 0 \qquad \text{or} \qquad a_1 > a_2 . \tag{4.6}$$

In the case where $a_2 > a_1$, D_{EP} will grow exponentially (Fig. 4.3) and no locking of ϕ_{out} and ϕ_{in} will be achieved.

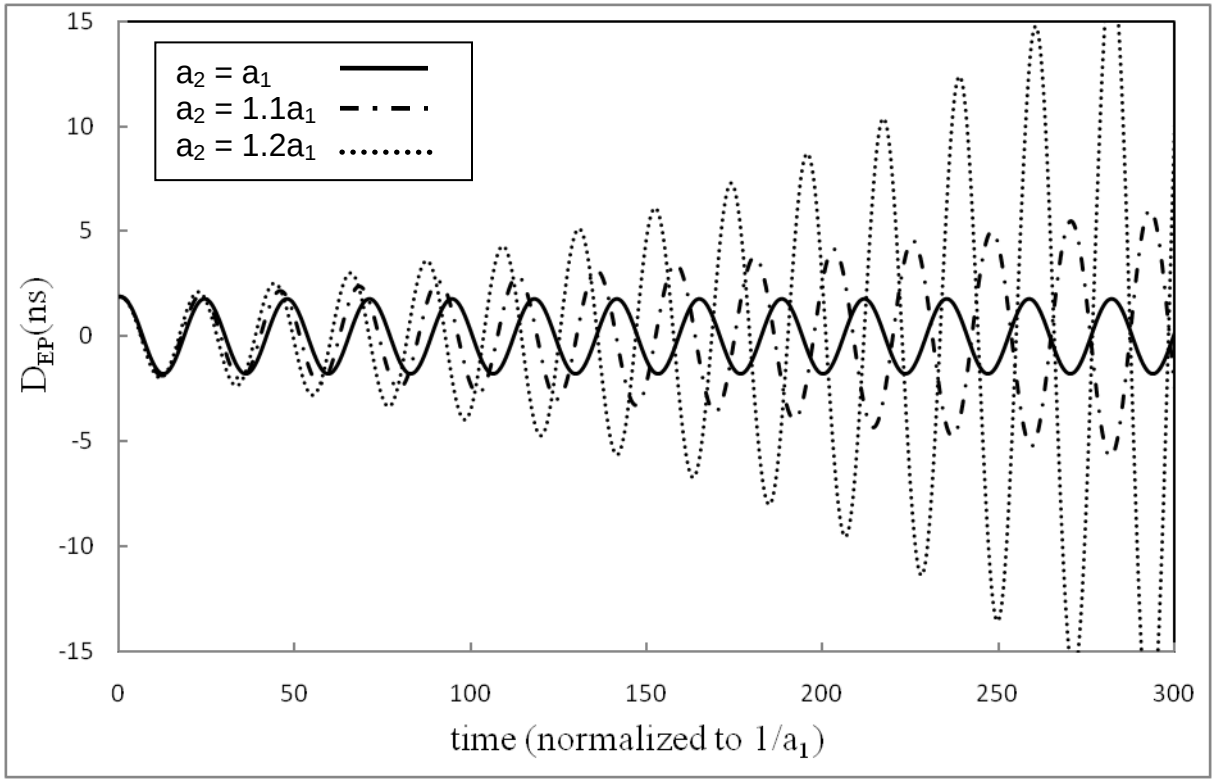


Figure 4.3: Step responses of delay error, D_{EP}, with respect to input step delay, D_{in}, for different values of peripheral loop parameter, a_2 for $a_2 > a_1$

4.1.2.2 Step response analysis

The other area of interest is the transient step response of D_{EP}. As the transfer functions are quite similar in form, only equation (4.2) is used in the study. First, optimal core loop parameters are chosen such that the step response is optimal. While a_0 is less flexible to change because it consists of parameters of the CP and VCDL, it is more convenient to define a_1 to define the core loop characteristics as it consists of only the loop filter parameters. Moreover, we also know from Routh Hurwitz theorem for stability that a_1 is the parameter that directly influences loop stability. Using actual design parameters, different step responses using various values of a_2 relative to a_1 are plotted in Fig. 4.4. The optimal value of a_2 is found to be about $0.05a_1$. Since the reciprocal of a_2 sheds some light on the time constant of the secondary loop, it gives some indication on the time span between each FSM decision. Together with equation (4.6), we know how to adjust the time in between each FSM decision or clock rate of the FSM block relative to the core loop filter design parameters.

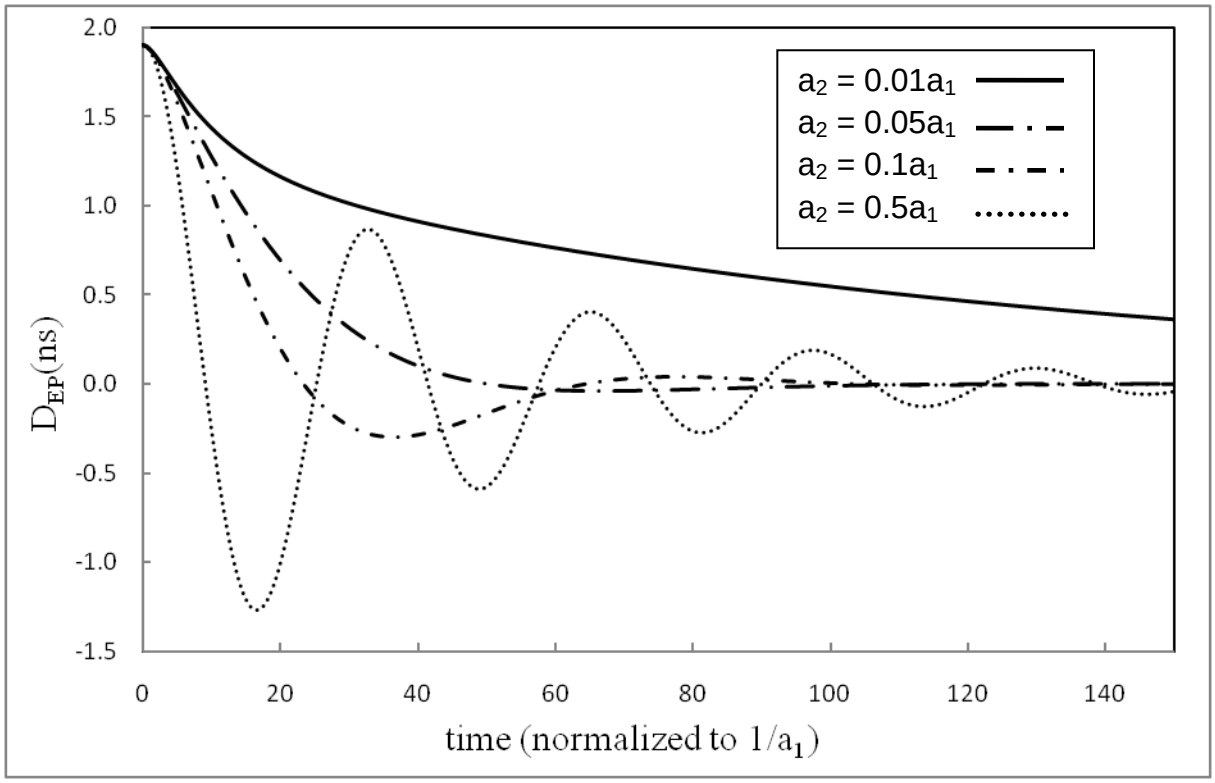


Figure 4.4: Step responses of delay error, D_{EP}, with respect to input step delay, D_{in}, for different values of peripheral loop parameter, a_2 for $a_1 > a_2$

4.2 Matlab modeling of proposed clock synchronization architecture

With better understanding of the dual loop dynamics, we can proceed to formulate a behavioral model in Matlab for full functional simulation before silicon implementation. This way, the functionality of the clock synchronization architecture can be verified before the actual circuit is designed and the behavioral model can also provide valuable insight during integration of all the different building blocks. Moreover, full chip simulation may not be possible due to the complexity of the resulting netlist. The overall behavioral model is shown in Fig. 4.5. Matlab Simulink is used for modeling.

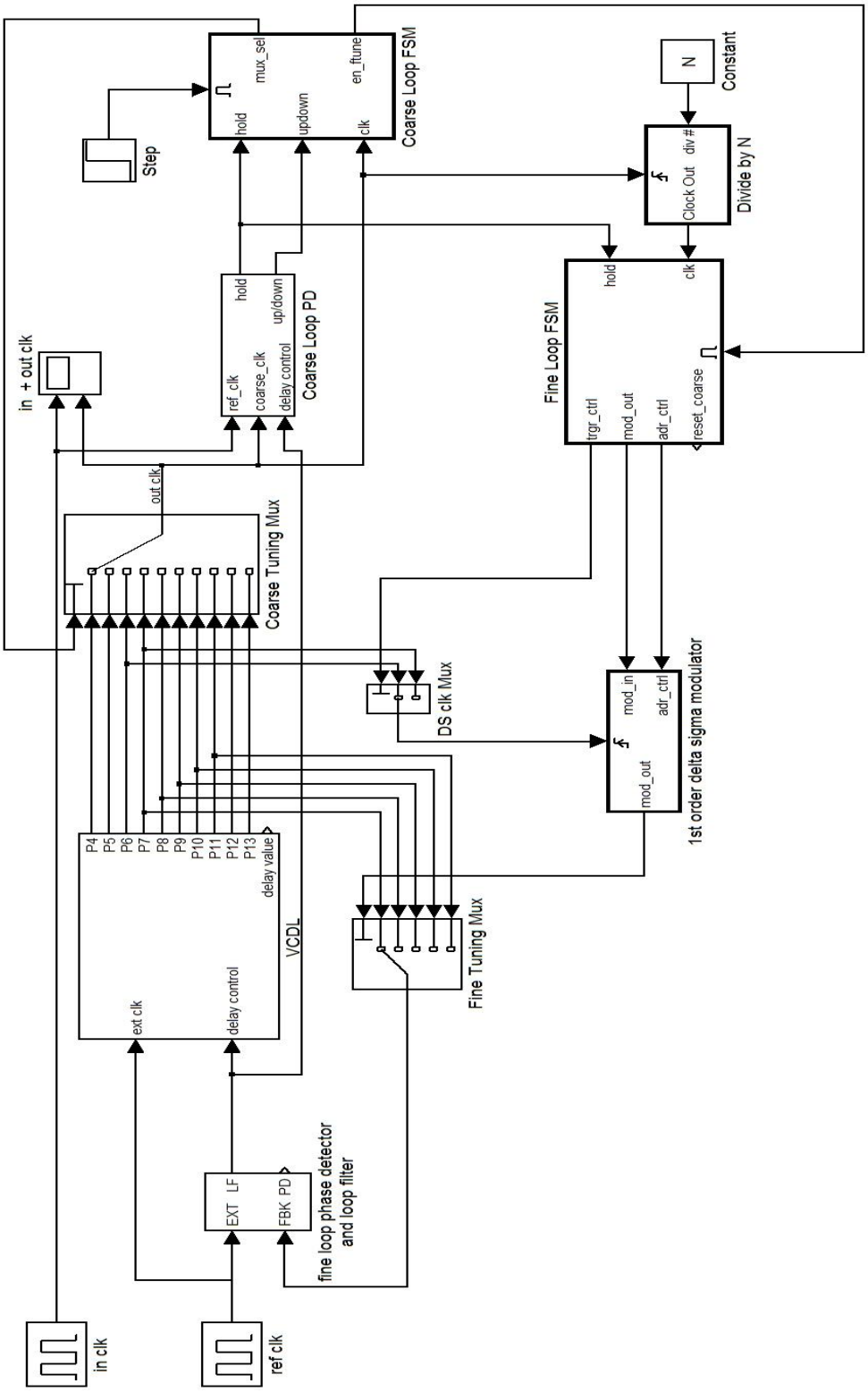


Figure 4.5: Full functional behavioral model of the clock synchronization system

While the PD, CP, loop filter and the VCDL blocks of the DLL can be easily modeled with available phase detector, transfer function and variable transport delay Simulink blocks as highlighted in Fig. 4.6 and 4.7, the fine tuning FSM control is slightly more difficult to integrate using Matlab's Stateflow flowchart.

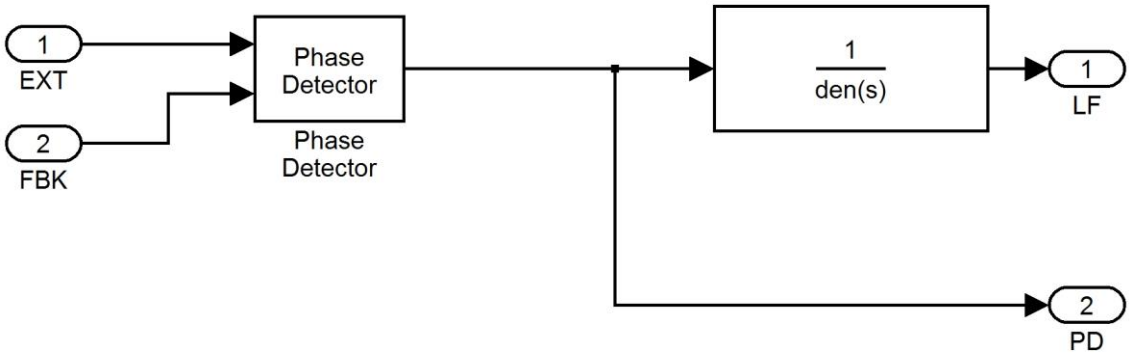


Figure 4.6: Simulink model of PD, CP and loop filter

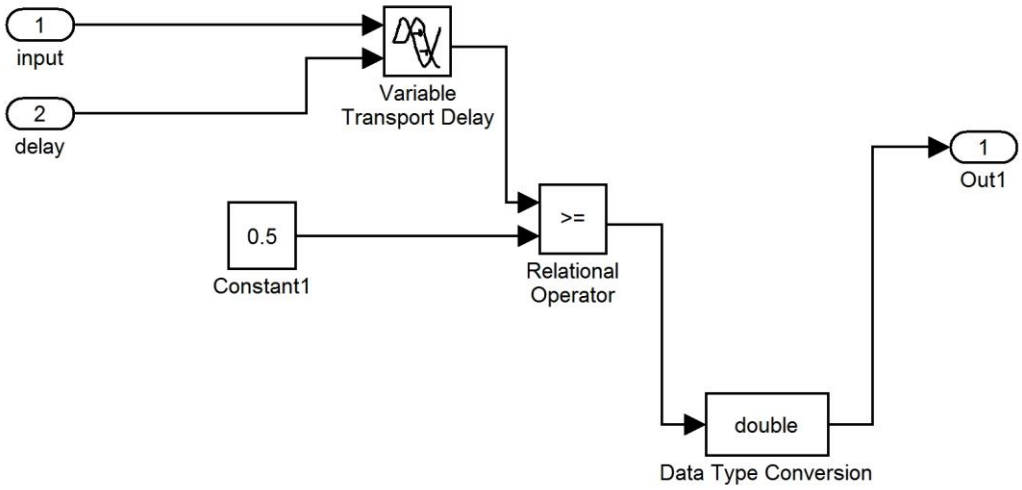


Figure 4.7: Simulink model of a single delay cell

The main difficulty lies in the interface between the flowchart and the remaining simulink blocks. Most of the Simulink blocks deal with integer whereas the Stateflow flowchart deals with bit processing. Therfore, in order to incorporate bit processing within the Stateflow flowchart, we had to call upon matlab embedded functions like bi2de and de2bi to translate between integer values and bit strings, in order to accurately model the peripheral control loop. The Stateflow flowchart for coarse and fine tuning can be found in Appendix A and the Stateflow Help section within the Matlab tool can be used to assist in the understanding of the flowchart semantics. A step function block is used to enable the coarse tuning block to ensure that coarse tuning starts only when the DLL has achieved phase lock. A programmable clock divide by N block is used to define the time span between each FSM decision. The time it takes for the FSM to make each decision should be carefully selected with respect to the core loop parameters based on the dual loop analysis earlier.

The coarse loop phase detector, CPD is modeled exactly after the actual hardware, consisting of delay cells, logic gates and flip-flops as shown in Fig. 4.8.

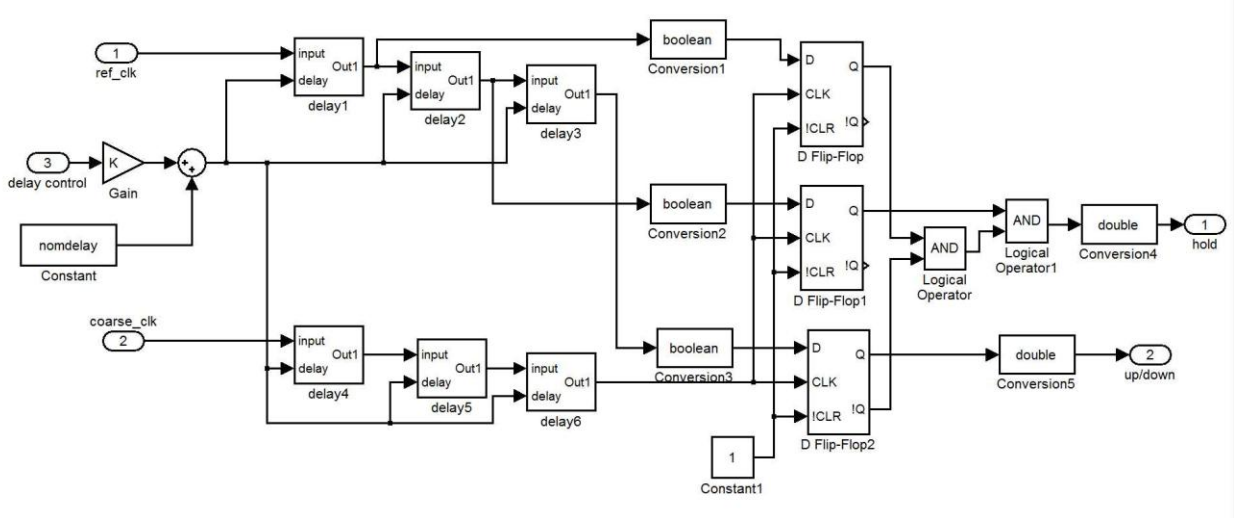


Figure 4.8: Simulink model of CPD

4.3 Behavioral simulation of clock synchronization architecture

The functioning and operation of the proposed architecture can be studied by full system Matlab simulation. Various time plots, such as loop filter settling, MUX selection and etc. are examined. Fig. 4.9 shows the initial lock of the core loop DLL before coarse tuning is enabled. The delay value of each delay cell is tracked in Fig 4.9(a). Fig. 4.9(b) shows locking of the dithered ϕ_{DLL} to ϕ_{ref} after initial settling.

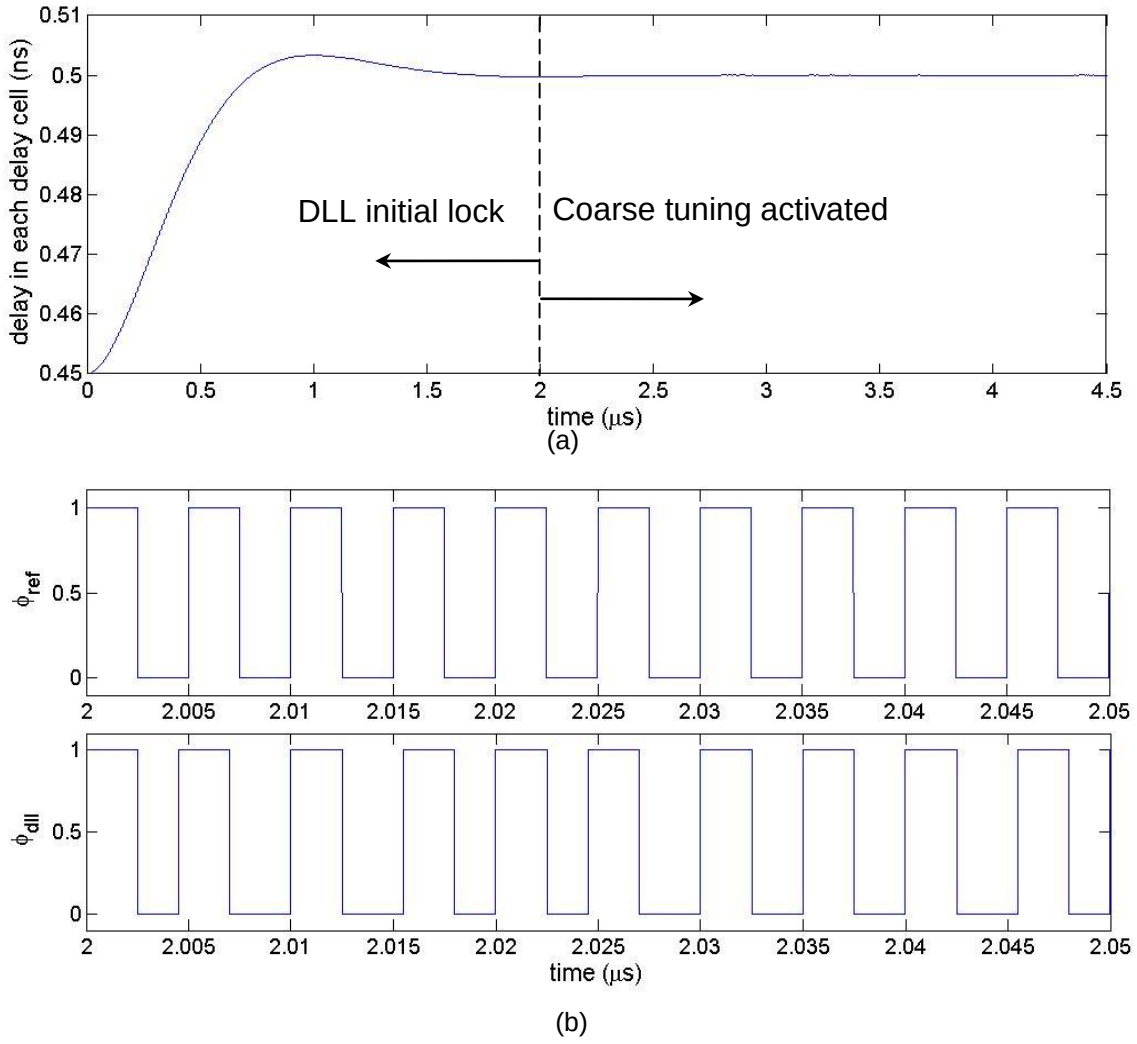


Figure 4.9: (a) Tracking of delay value of each delay cell during initial phase lock and (b) locking of the dithered ϕ_{DLL} to ϕ_{ref} after initial settling

After the initial DLL lock is established, the coarse loop FSM is activated. The step function in Fig. 4.10 enables the coarse loop FSM and the coarse loop FSM adjusts the coarse loop MUX and finally selects the clock edge nearest to the input clock edge. After it determined the nearest clock edge when HOLD=1 is attained, the coarse tuning FSM section relinquishes control to the fine tuning part of the FSM block by sending out the fine tune FSM enable signal. Fig. 4.11 shows the diminishing delay difference between ϕ_{out} and to ϕ_{in} with each progressive step of coarse tuning.

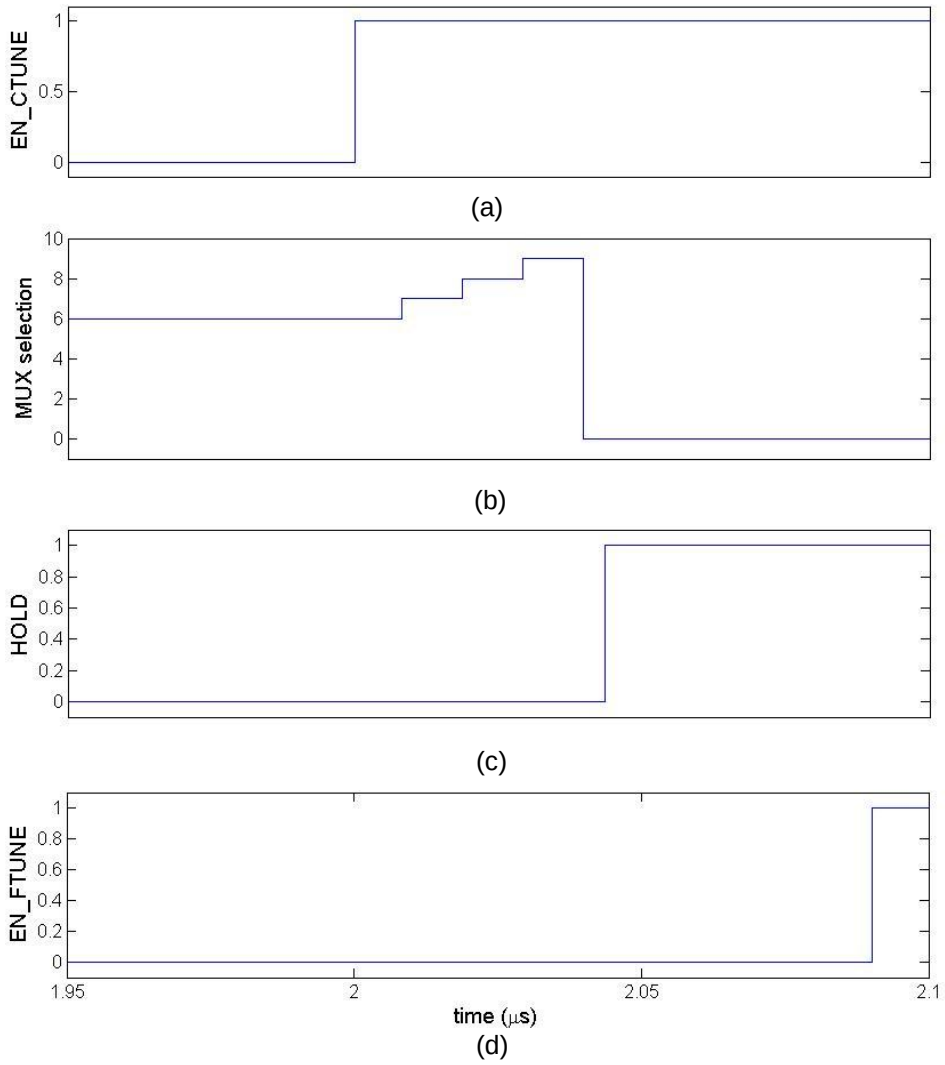


Figure 4.10: Plots showing (a) coarse tuning enable, (b) coarse tuning MUX selection, (c) HOLD signal and (d) fine tuning enable phase during the coarse tuning phase

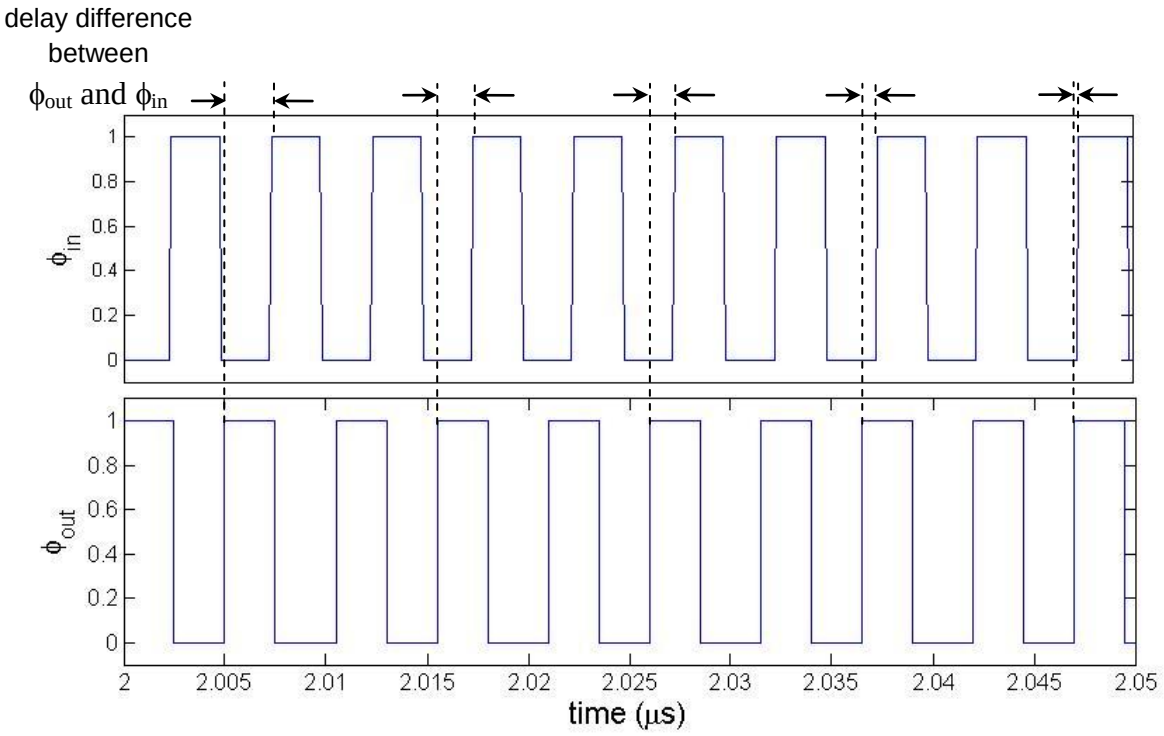


Figure 4.11: Progressive step of shifting ϕ_{out} closer to ϕ_{in} during coarse tuning

After the coarse tuning step has completed its course, fine tuning takes place. The fine tuning control will use the SA (Successive Approximation) approach to quickly fine tune the ΔΣ modulator to find the best ΔΣ modulator input value that will enable ϕ_{out} to shift nearest to ϕ_{in}. In the case where ϕ_{out} is shifted ϕ_{in} Figure 4.12 best illustrates the operation.

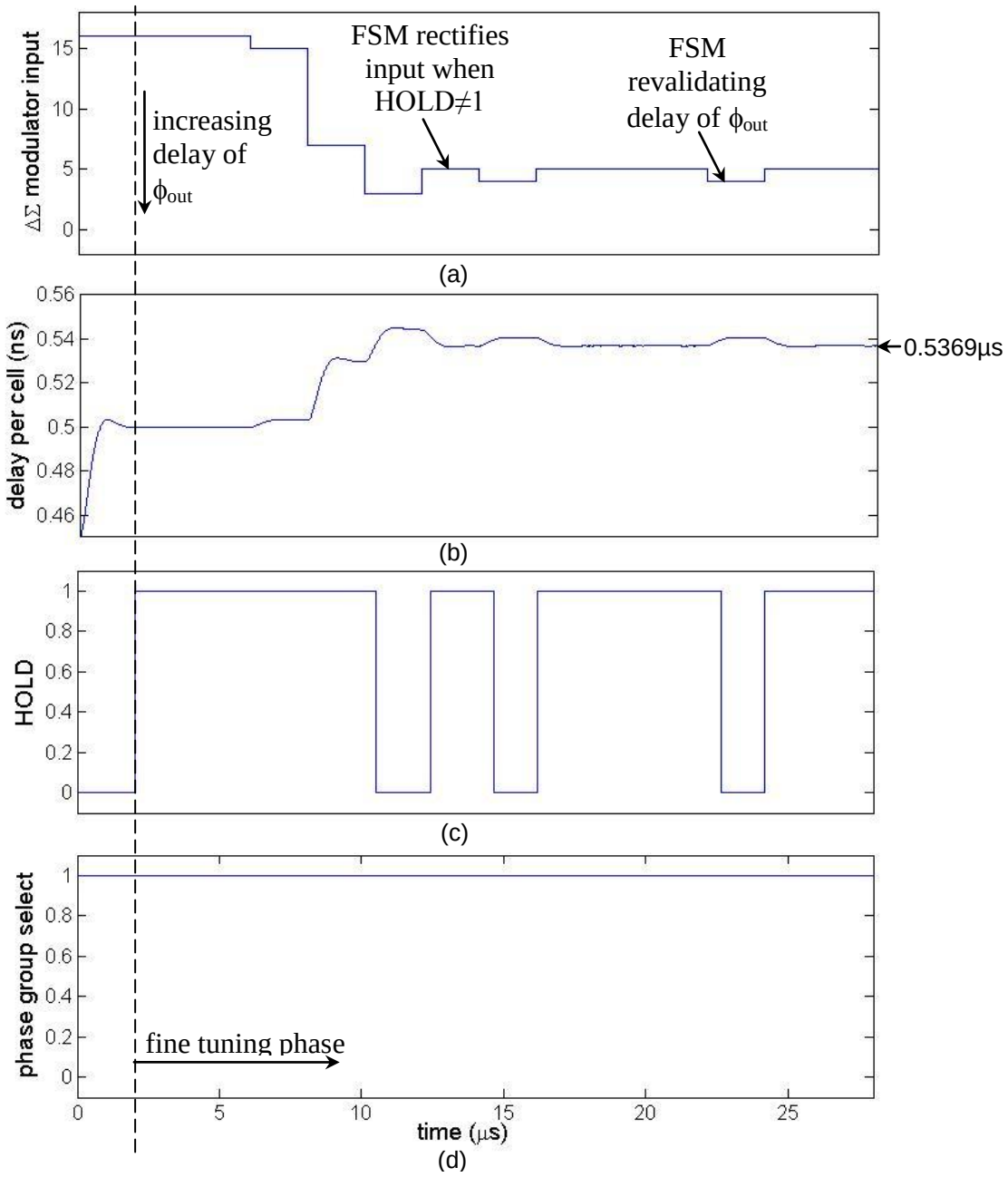


Figure 4.12: Plots showing (a) ΔΣ modulator input, (b) time delay per cell, (c) HOLD signal (d) feedback delay group select signal during fine tuning phase

Fig. 4.12 shows the interactions of the different signals during the fine tuning process. It illustrates how the delay of each delay cell changes when the ΔΣ modulator input is being tuned by the FSM. In addition, it also exhibits how FSM tunes the modulator input based on the HOLD signal.

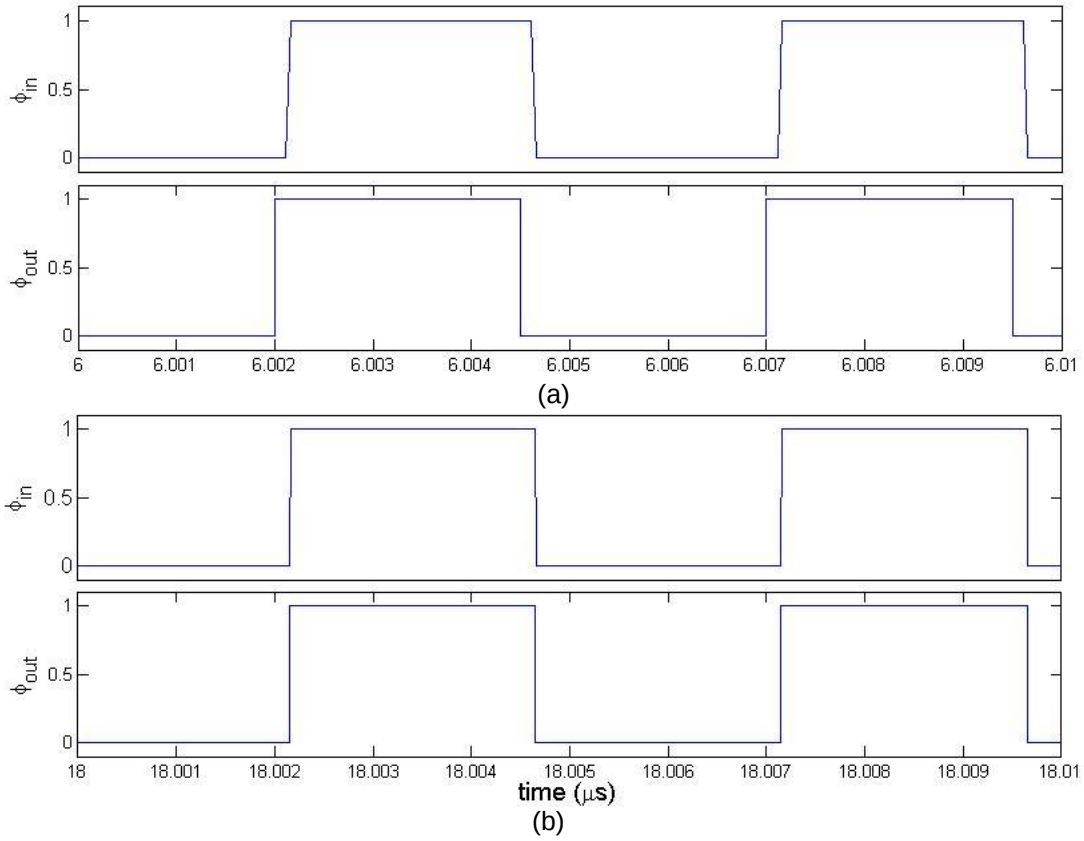


Figure 4.13: ϕ_{in} and ϕ_{out} (a) before and (b) after fine tuning

The FSM uses the SA (Succesive Approximation) approach for fine tuning. It selects a midpoint value (i.e. 7) in the entire ΔΣ modulator input range (0 to15). The HOLD signal is subsequently evaluated. The FSM tunes to another midpoint value in either the upper (8 to 15) or lower (0 to 6) input range depending on the HOLD signal. By reducing the input range with each approximation, the delay of ϕ_{out} eventually approaches the delay of ϕ_{in}. SA is an efficient algorithm especially when handling a long digital input word. Fig 4.13 highlights the end result of fine tuning. It compares the delay difference between ϕ_{in} and ϕ_{out} before and after fine tuning.

As an additional feature of the circuit, the FSM intentionally perturbs the ΔΣ modulator after a fixed number of cycles to revalidate the synchronization of ϕ_{out} to ϕ_{in}.

To demonstrate full clock coverage, we purposely introduce a delay difference between ϕ_{in} and ϕ_{out} such that tuning $N_{average}$ from 10 to 9 is not enough to shift ϕ_{out} close enough to ϕ_{in}. In this case, the FSM would still use SA approach to tune to the smallest input value of ΔΣ modulator. Upon finding that ϕ_{out} can be shifted further, the FSM would change the feedback delay group of 8^{th} to 11th taps to the group of 7^{th} to 10^{th} taps. Subsequently, the FSM would tune the ΔΣ modulator input using the same SA algorithm until the delay of ϕ_{in} almost matches ϕ_{out}. The entire operation is described in Fig. 4.14. Likewise, Fig. 4.15 shows the end result of fine tuning. The residual delay after the first round of SA tuning is also shown in Fig. 4.15(b).

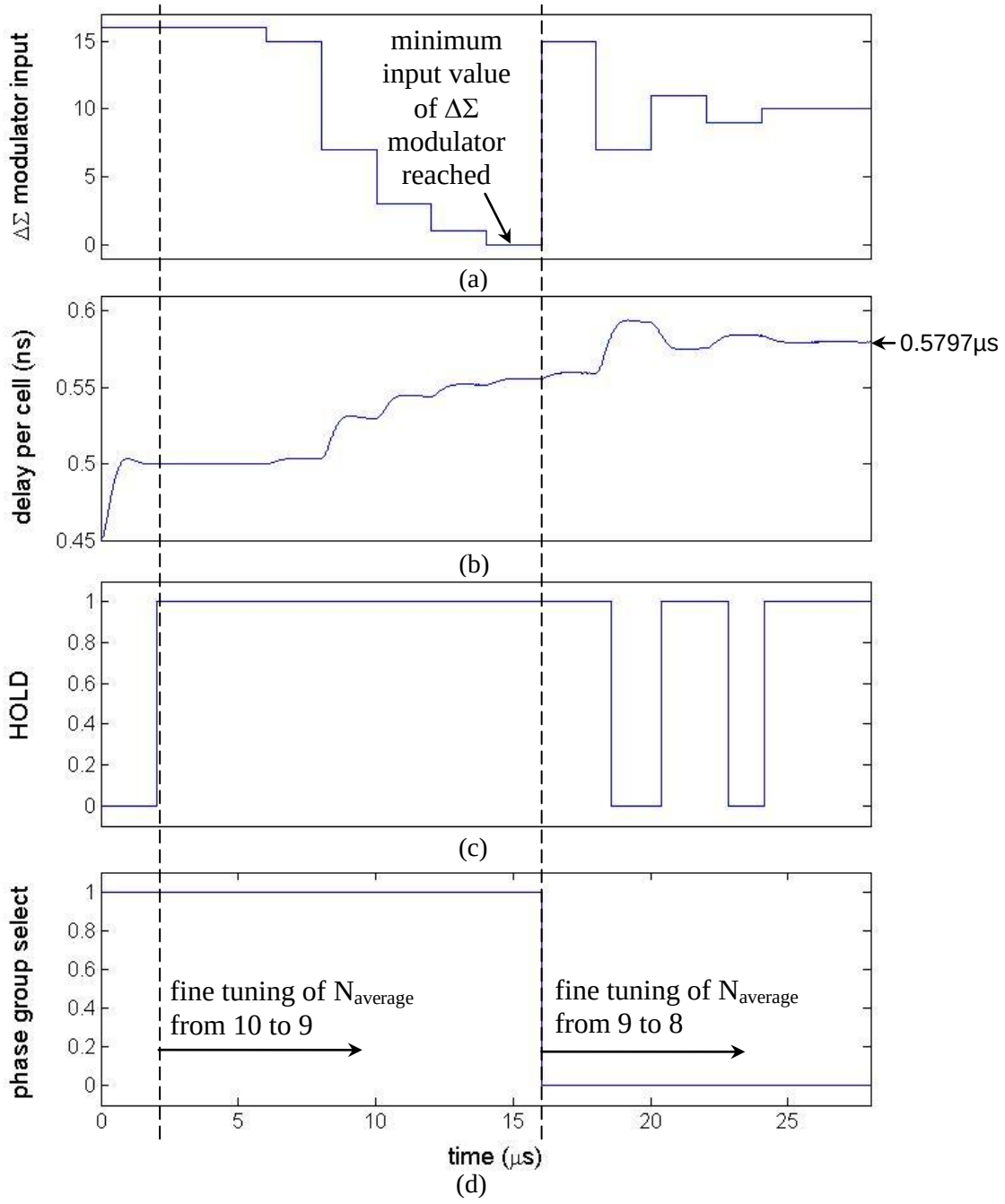


Figure 4.14: Plots showing (a) ΔΣ modulator input, (b) time delay per cell, (c) HOLD signal (d) feedback delay group select signal when tuning $N_{average}$ from 10 to 9 is insufficient

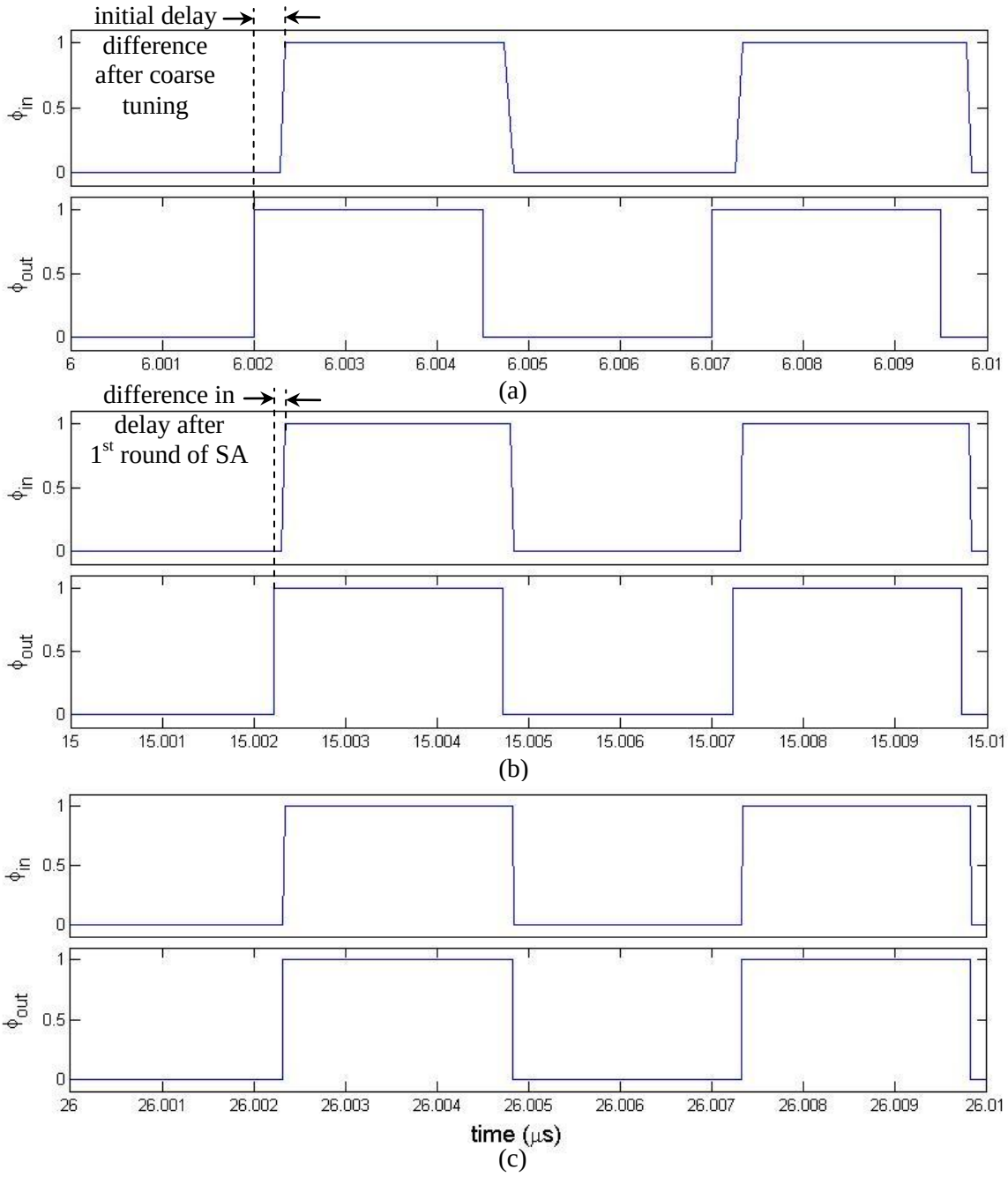


Figure 4.15: Delay differences in ϕ_{in} and ϕ_{out} (a) before fine tuning, (b) after 1st round of SA tuning and (c) after 2nd round of SA fine tuning

Overall, the clock synchronization architecture model works quite well. The model can also used to obtain the transfer characteristic of the DLL, shown in Fig. 4.12, which shows excellent matching to the theoretical delay estimation.

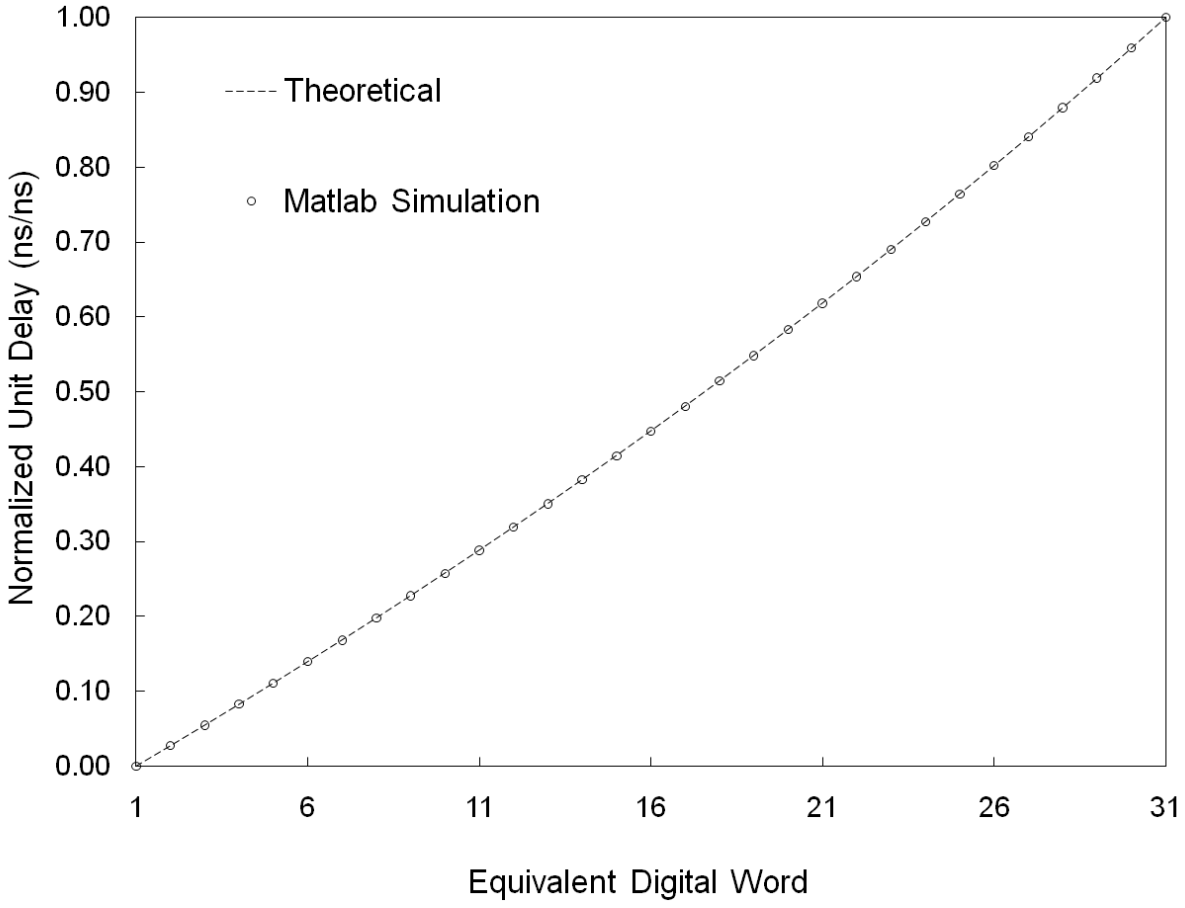


Figure 4.16: Transfer characteristic of the ΔΣ DLL

4.4 Conclusion

A system model and a full functional behavioral model of the clock synchronization architecture are successfully modeled using Matlab Simulink. Both models can provide valuable insight to the design of the clock synchronization system. The system model studies the effects of the core and peripheral loop parameters on stability and transient response of the system. The functional model can be used to generate the transfer characteristic, which concurs with our earlier theoretical analysis.

CHAPTER 5. CIRCUIT IMPLEMENTATION

The overall clock synchronization system is shown in Fig. 5.1. The main blocks of the clock synchronization consists of 13-cell delay line, the anti-harmonic lock detector (AHD), the adaptive loop filter, ΔΣ modulator, the finite state machine (FSM) block and the coarse loop phase detector (CPD). The circuit implementation of these blocks will be discussed in the next few sections of this chapter.

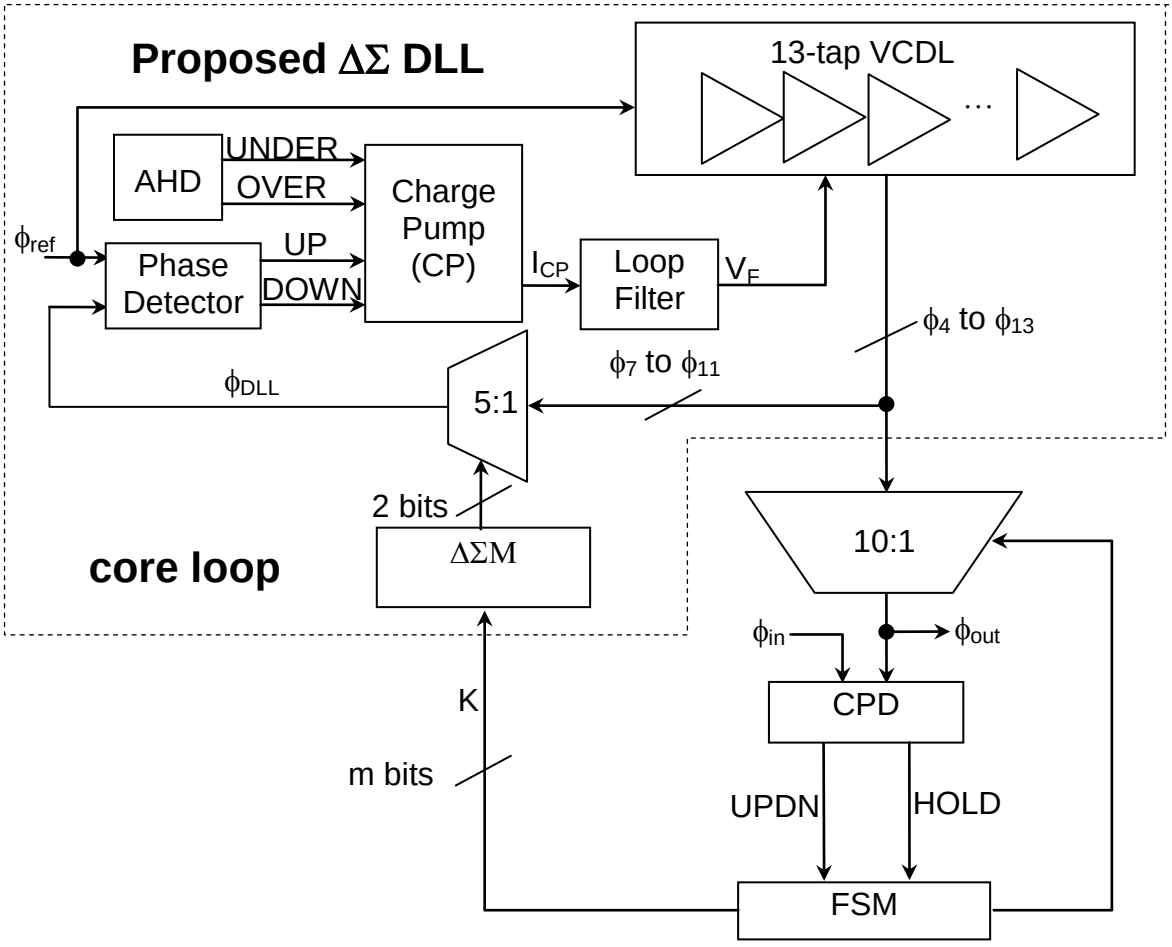


Figure 5.1: Clock Synchronization System

5.1. Delay-cell with replica biasing and load matching

A delay cell with large delay variation is required to cater for wide range of input frequencies for this project. The differential buffer delay stage proposed in [13] is employed in this design, and is illustrated in Fig. 5.2. The differential architecture is chosen for its better supply and common-mode noise rejection. The replica bias buffer is used to avoid loading the charge pump circuitry directly onto the delay cells as well as to maintain constant output swing for each delay cell. From simulation, for control voltage ranging from 1 to 2.1V, the delay per cell can vary from 300ps to 4ns as shown in Fig. 5.3. As observed in Fig 5.3, the delay varies exponentially with the control voltage. This will cause the delay gain, which is given by the slope of the curve, to vary substantially throughout the entire range. Therefore, the DLL loop characteristics will change accordingly.

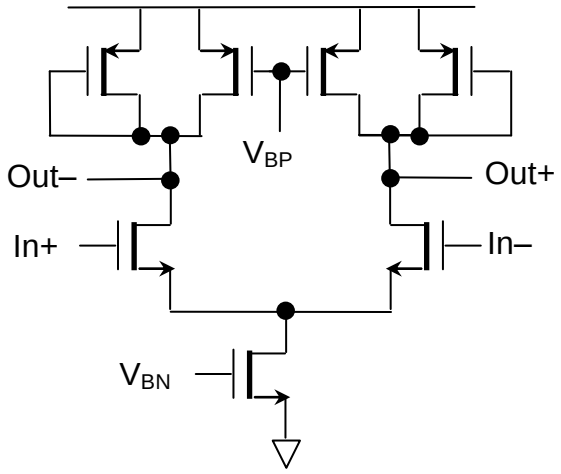


Figure 5.2: Delay cell schematic

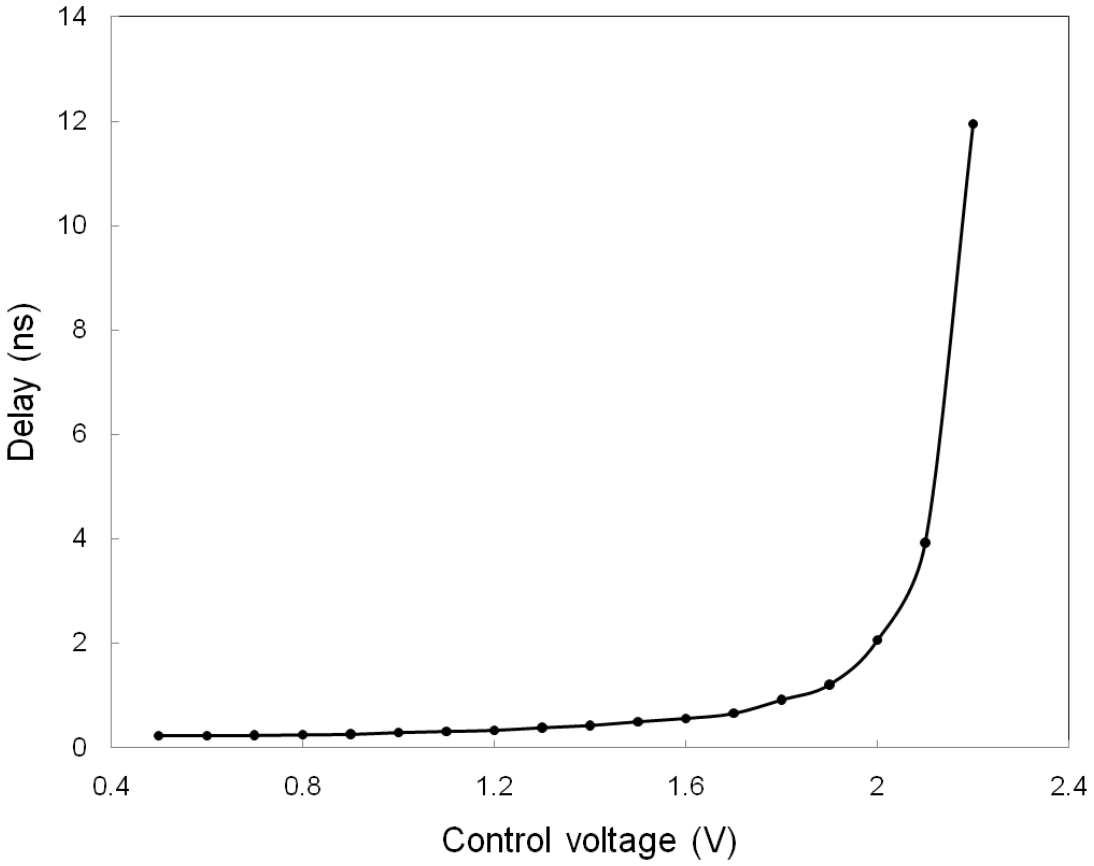


Figure 5.3: Delay cell delay characteristics

Although the equivalent schematic for the replica bias is shown in Fig. 5.4(a), its' layout version is implemented using 2 identical delay cells which is equivalent to Fig. 5.4(b) to ensure better matching. While the layout is structurally the same as that of two delay cells side by side, it only differs in the metal connection of the gate to source at the respective PMOS loads.

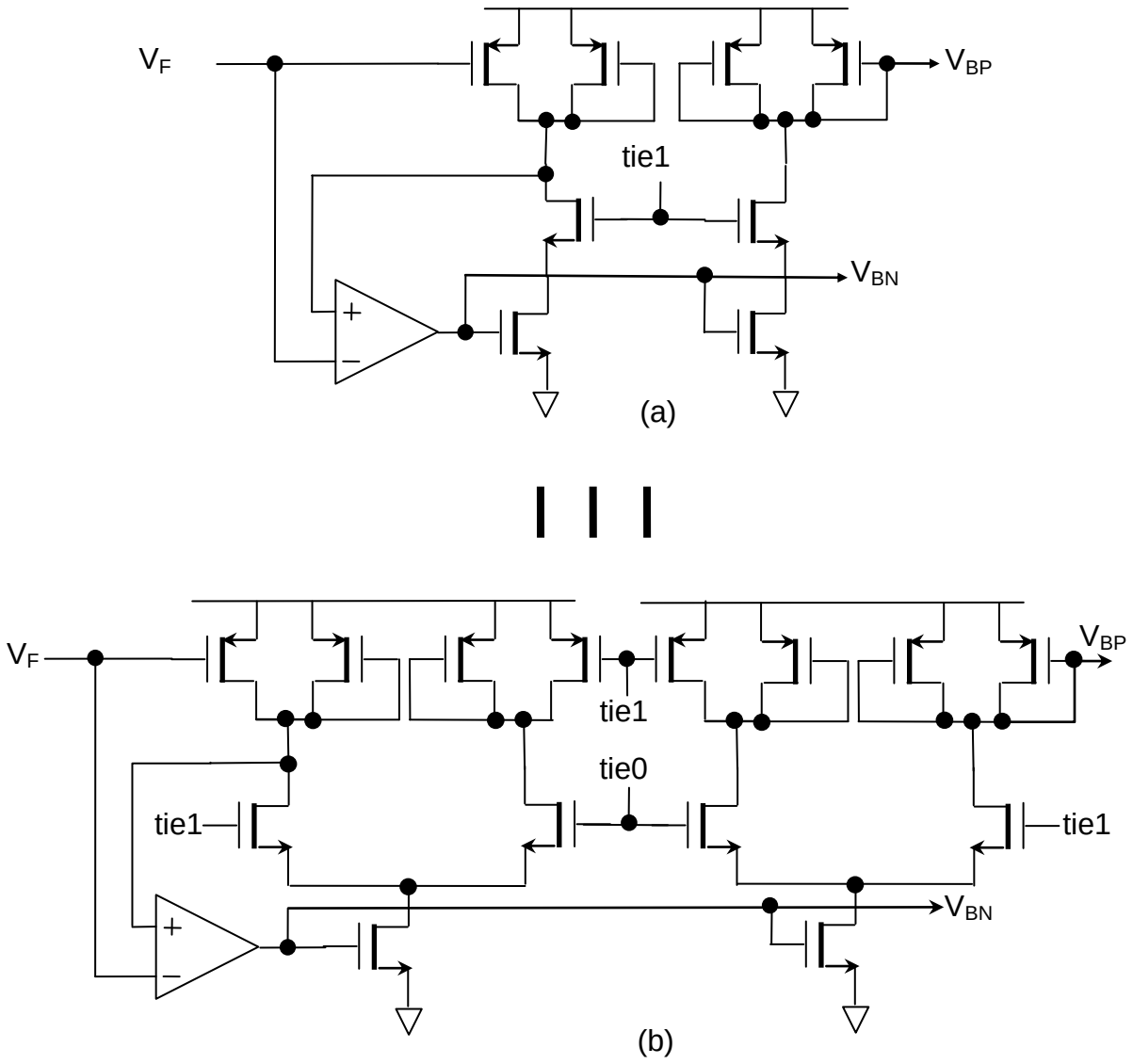


Figure 5.4: Replica biasing (a) equivalent circuit (b) layout version

Differential MUX is also employed in this design. The schematic is given in Fig. 5.5. The current and load values are designed to ensure sufficient swing at the output when driving the next stage, which is the differential to single ended buffers.

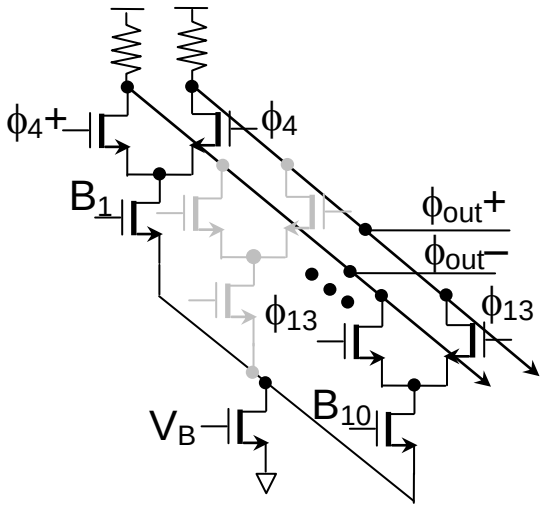


Figure 5.5: Differential MUX implementation

The single PMOS transistor current (I_{DP}) of the delay cell and the delay of each delay cell have been shown in [13] as:

$$I_{DP} = \frac{K_P}{2}\left(|V_{GS}| - |V_{THP}|\right)^2 = \frac{K_P}{2}\left(V_{DD} - V_F - |V_{THP}|\right)^2, \tag{5.1}$$

and

$$T_{delay} = \frac{C_{EFF}}{g_{mp}} = \frac{C_{EFF}}{K_P\left(|V_{GS}| - |V_{THP}|\right)} = \frac{C_{EFF}}{K_P\left(V_{DD} - V_F - |V_{THP}|\right)}, \tag{5.2}$$

where K_P is the PMOS device transconductance, V_F is the loop filter voltage, g_{mp} is the transconductance, V_{THP} is the PMOS threshold voltage, and C_{EFF} is the effective load capacitance of one delay cell.

5.2. Anti-harmonic lock detector

Due to the wide dynamic range of the delay for the delay cell employed, anti-harmonic lock detector (AHD) is needed to avoid false locking. The AHD will send signals to the charge pump to adjust the loop filter voltage, overriding the PD, when the ϕ_{DLL} edge falls outside the valid lock range between $0.5T_{clk}$ to $1.5T_{clk}$. Fig. 5.6 shows the timing operation and how the UNDER and OVER signals are generated.

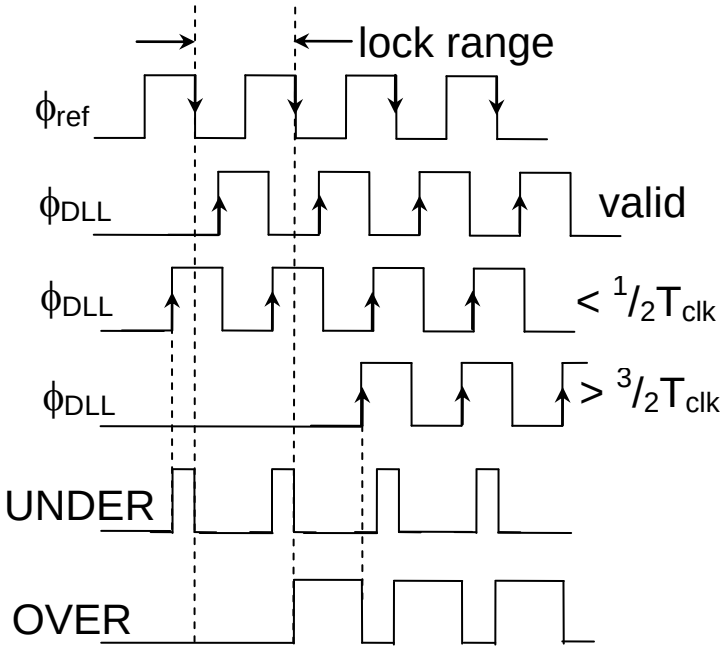


Figure 5.6: Anti-harmonic lock detector operation

The AHD is implemented using two serial shift registers as shown in Fig. 5.7. The outputs of the shift registers are initialized to have logic values as shown. When triggered by the negative and positive edges of ϕ_{ref} to ϕ_{DLL} respectively, the logic values shift to the right and loops back to the 1st register when it reaches the end, corresponding to the relative positions of the respective edges. Using the outputs of the shift registers, the UNDER and OVER signals are then generated based on the relative edge positions of ϕ_{ref} and ϕ_{DLL}. If the negative edge of ϕ_{ref} leads the positive edge of ϕ_{DLL} by 2 positions, the OVER signal will be generated. On the other hand, the UNDER signal is generated when the negative edge of ϕ_{ref} lags the positive edge of ϕ_{DLL} by 1 position. This detector operates based on relative positions of the input signal edges rather than level-sampling based detector described in [14], making it less susceptible to the duty cycle of ϕ_{DLL}. It should be pointed out that due to the randomization of feedback phases of the proposed ΔΣ DLL, ϕ_{DLL} might not exhibit 50% duty cycle. However, the proposed AHD still requires the duty cycle of ϕ_{ref} to be roughly 50%, which is relatively easier to achieve and design compared to the varying ϕ_{DLL}.

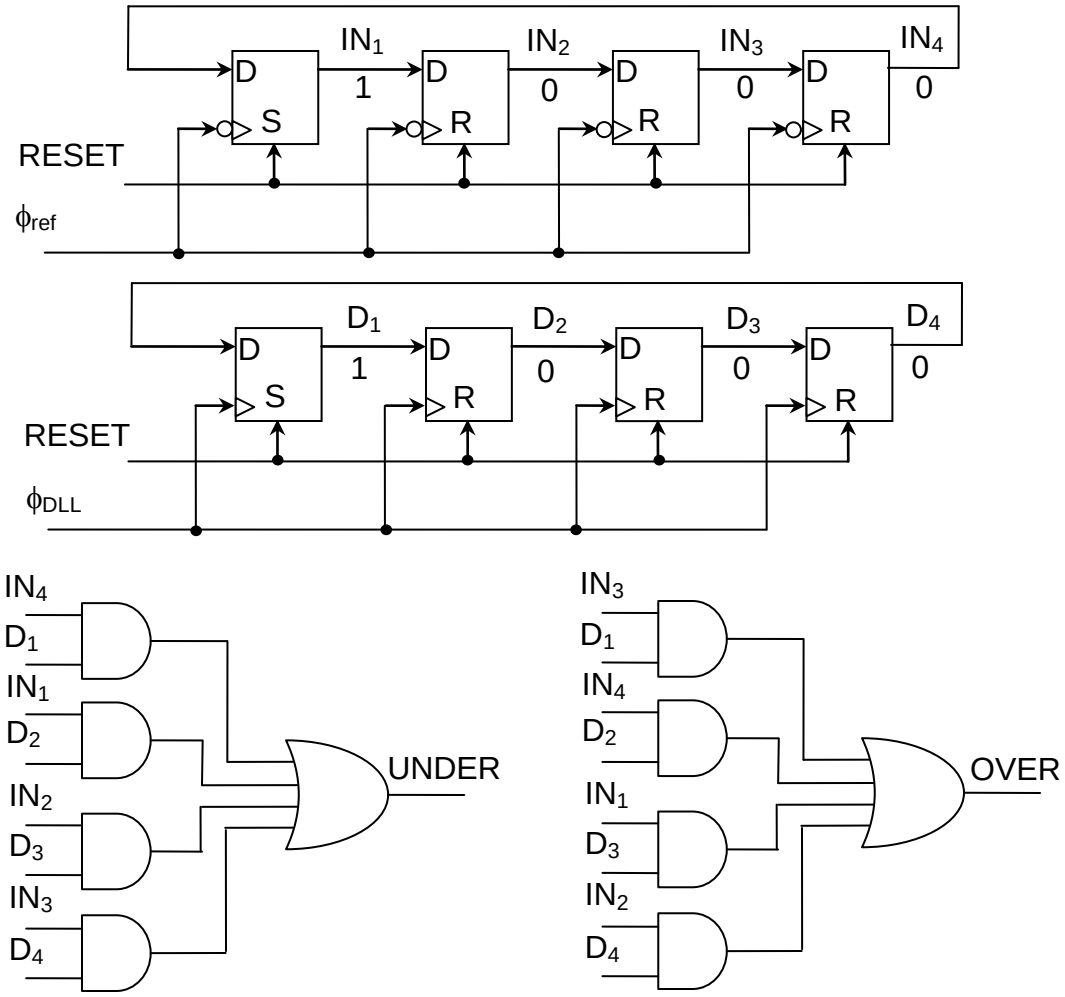


Figure 5.7: Anti-harmonic lock detector implementation

5.3. Adaptive loop filter

Compared to conventional DLL, ΔΣ DLL requires higher order loop filter in order to suppress the high frequency noise due to the noise shaping. As the first order ΔΣ modulator provides first order noise shaping, at least a second order loop filter is required to attenuate the high frequency shaped noise. If the order of both the ΔΣ modulator and the loop filter are the same, high frequency noise will not be fully suppressed. This noise might creep back into the circuit, contributing to jitter in the DLL. Added to the challenge is the input clock frequencies with wide dynamic range that the loop filter needs to operate at. In this design, a simple adaptive RC passive second order filter, consisting of 2 capacitors and an active MOSFET resistor as shown in Fig. 5.8, is implemented.

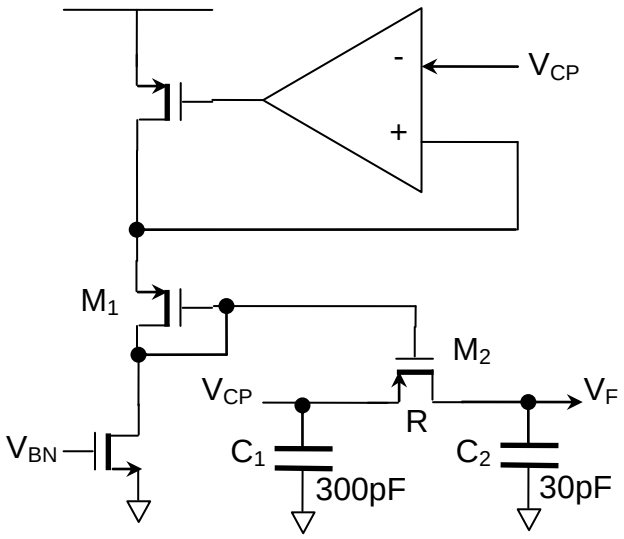


Figure 5.8: Adaptive loop filter schematic

The desired loop filter characteristic of loop filter voltage output, V_F, with charge pump output, I_{CP}, is given by

$$F(s) = \frac{V_F}{I_{CP}} = \frac{1}{s^2 C_1 C_2 R + s(C_1 + C_2)}, \tag{5.3}$$

where C_1 and C_2 are the values of the respective capacitors and R is the resistance of the active MOSFET resistor. The resultant closed loop transfer function have the form of:

$$H(s) = \frac{\phi_{OUT}}{\phi_{REF}} = \frac{\frac{I_{CP} K_{DL}}{T_{clk}}}{s^2 C_1 C_2 R + s(C_1 + C_2) + \frac{I_{CP} K_{DL}}{T_{clk}}} \tag{5.4}$$

Further simplifying the transfer function in (5.4) by making some approximations will give us a convenient form:

$$H(s) = \frac{\frac{k}{C_1 C_2 R}}{\left(s + \frac{k}{C_1}\right)\left(s + \frac{1}{C_2 R}\right)}, \tag{5.5}$$

where $k = \frac{I_{CP} K_{DL}}{T_{clk}}$. (5.5) has two poles given by

$$\omega_{p1} = \frac{I_{CP} K_{DL}}{T_{clk} C_1}, \tag{5.6}$$

and

$$\omega_{p2} = \frac{1}{R C_2}, \tag{5.7}$$

where I_{CP} is the charge pump current and K_{DL} is the delay gain of the VCDL.

The loop bandwidth can be approximated by the dominant pole, ω_{p1} and inferring from equation (5.5), the resultant phase margin (PM) will be given by:

$$PM = 90^{o} - \tan^{-1}\left(\frac{\omega_{p1}}{\omega_{p2}}\right) \tag{5.8}$$

To achieve adaptive bandwidth for wide frequency range operation, the ratio of ω_{p1} to the input clock frequency (f_{clk}) has to be kept constant. From (5.6), this ratio is given by

$$\frac{\omega_{p1}}{f_{clk}} = \frac{I_{CP}K_{DL}}{C_1}. \tag{5.9}$$

To maintain constant ω_{p1}/f_{clk} ratio, I_{CP} and K_{DL} have to be made constant. However, given the wide delay range that we are targeting for in the design, K_{DL} is highly non-linear. In [13], the delay gain (K_{DL}) is found to be inversely proportional to the biasing current of the cell delay (I_{DP}) and varies with the input clock frequency (f_{clk}). Therefore, I_{CP} is mirrored from the replica bias of the delay cell in earlier section to track the delay gain variation under various input clock frequencies to maintain a constant ratio of ω_{p1}/f_{clk} [13]. Similar approach is adopted here with additional programmable element being introduced to the charge pump as illustrated in Fig. 5.9. The charge pump is made programmable to allow fine tuning of the loop filter bandwidth for better jitter performance.

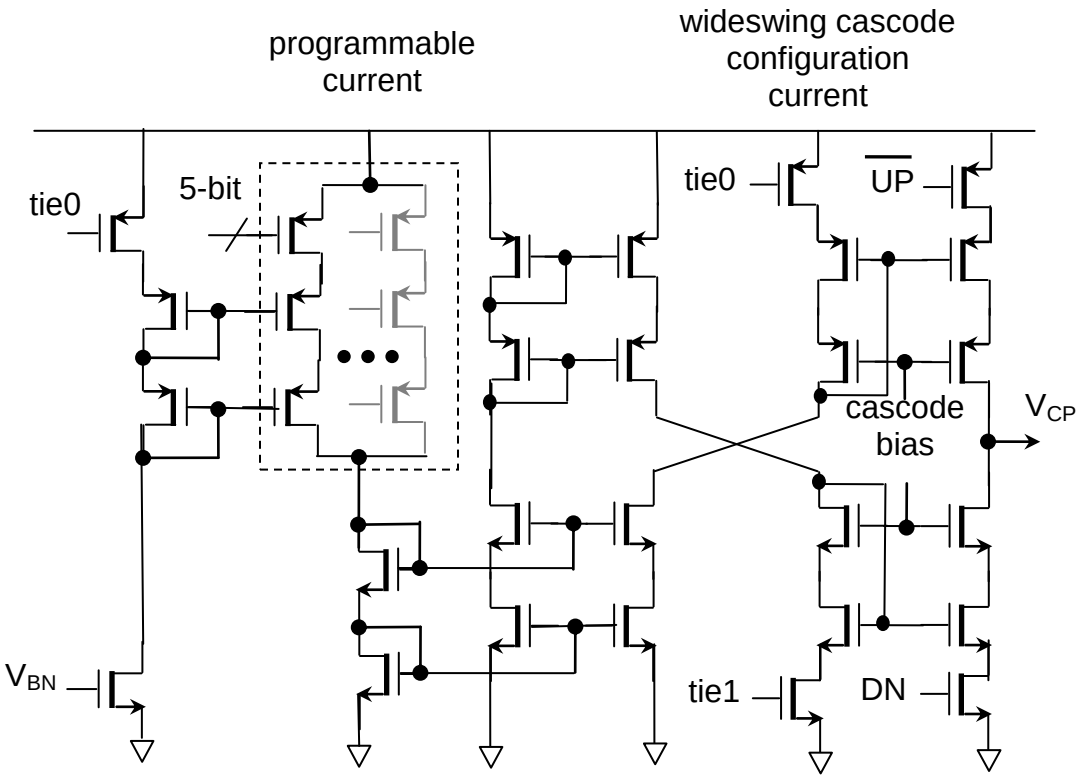


Figure 5.9: Charge pump with current sensing schematic

The resulting ω_{p1}/f_{clk} can be shown as:

$$\frac{\omega_{p1}}{f_{clk}} = \frac{x}{2}\frac{C_B}{C_1}, \tag{5.10}$$

where $C_B=2\times N\times C_{EFF}$ is the total effective buffer capacitance of all the delay stages and x is the proportionality constant between the charge pump current (I_{CP}) and the single PMOS transistor current (I_{DP}) of the delay cell.

Unlike [13], which only employs C_1 as first order loop filter, the need of 2nd order loop filter complicates the adaptive loop filter design due to the additional 2nd pole, ω_{p2}. In practice, ω_{p2}/ω_{p1} is kept at fixed ratio (~2.2) to maintain desired phase margin (PM) while providing sufficient high frequency noise suppression. Combining (5.6), (5.7) and (5.10),

$$\frac{\omega_{p2}}{\omega_{p1}} = \frac{2T_{clk}C_1}{xC_BRC_2}. \tag{5.11}$$

For a given VCDL with N delay stages,

$$T_{clk} = NT_{delay} = \frac{C_B}{2K_P\left(V_{DD} - V_F - |V_{THP}|\right)} = \frac{C_B}{2g_{mp}}. \tag{5.12}$$

By examining (5.11) and (5.12), if R is made to track the inverse of g_{mp} of the delay cell, the ω_{p2}/ω_{p1} ratio can then be kept constant. In this design, the adaptive tuning of the 2nd pole is achieved by a replica bias as shown in Fig. 5.6. The delay cell current (I_{DP}) is first established in a branch passing through a diode-connected PMOS transistor M_1. This will setup the desired $|V_{GS}|$ for the transistor M_1 which gives rise to transconductance that closely tracks the g_{mp} of the delay cell. An opamp is employed to fix the source terminal of M_1 at V_{CP}. The resulting gate bias of M_1 is then applied to the gate of M_2 operating at linear region through replica bias concept with the following conductance:

$$\frac{1}{R} = g_{ds,M2} = \alpha K_P\left(|V_{GS,M1}| - |V_{THP}|\right) = \alpha g_{m,M1} = \alpha g_{mp}. \tag{5.13}$$

where α is the sizing ratio between transistor M_2 and M_1. Substituting (18) and (17) into (16), the constant ω_{p2}/ω_{p1} ratio can then be obtained as

$$\frac{\omega_{p2}}{\omega_{p1}} = \frac{\alpha C_1}{xC_2}. \tag{5.14}$$

5.4. ΔΣ modulator

The 1st order ΔΣM with dithering [15] is shown in Fig. 5.10. First order ΔΣM is employed to reduce the complexity of the adaptive loop filter design. However, 1st order ΔΣM has very poor randomization property and thus mandates a dithering block to eliminate any periodic pattern exhibits at the output. A 25 bit pseudo-random sequence with the gain of one unit of quantization level is employed before the quantizer input [15] to achieve the dithering.

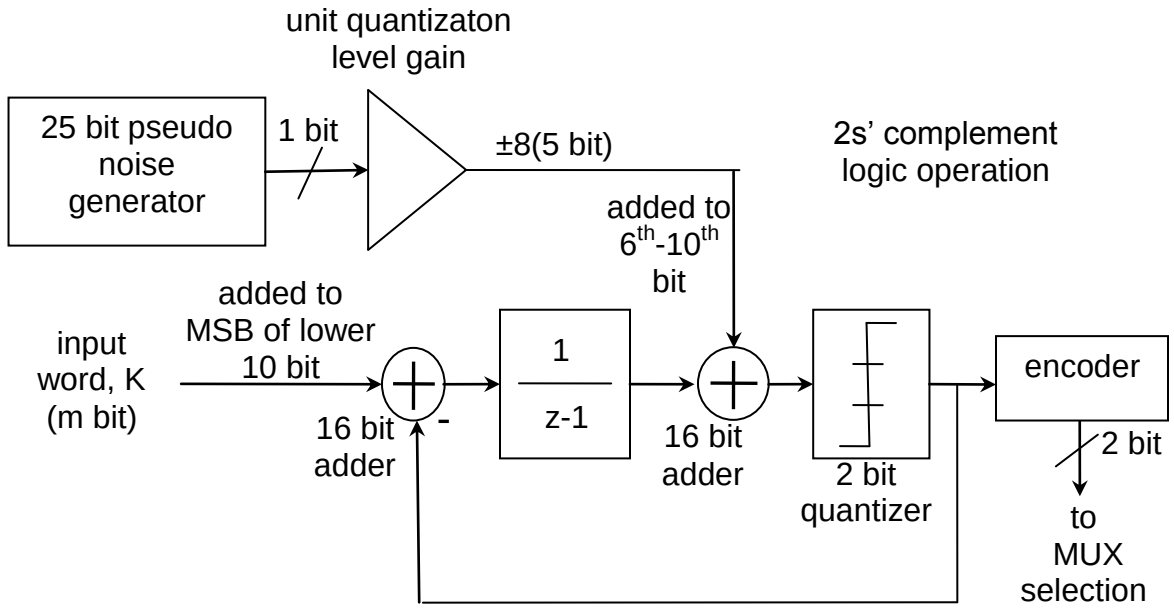


Figure 5.10: Programmable 1st order ΔΣ modulator with dithering

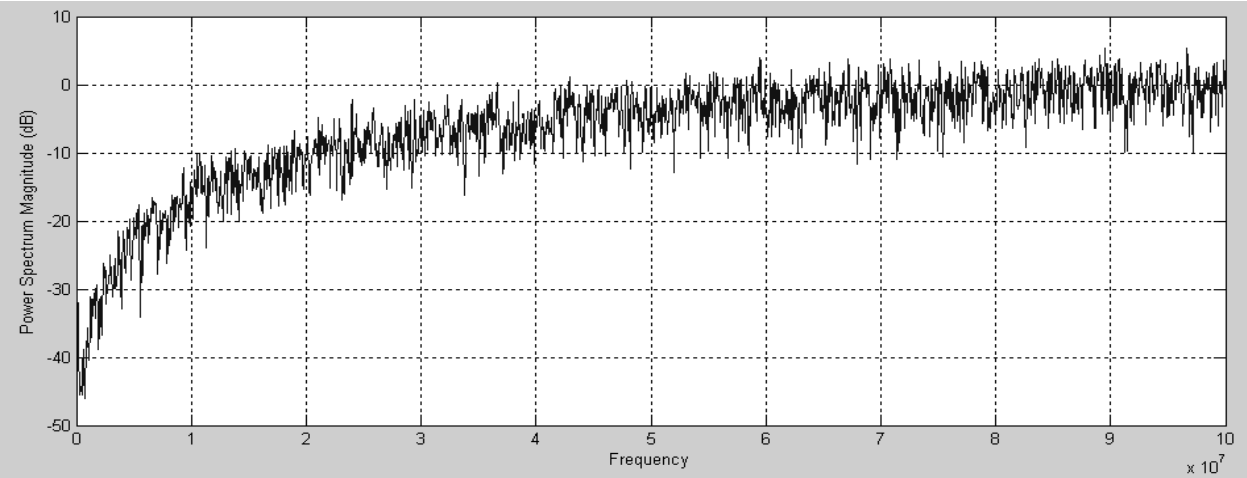


Figure 5.11: Noise shaping from 1st order ΔΣ modulator

As shown in Fig. 5.11, the modulator output demonstrate 1st order noise shaping without significant spurious tones. The spurious tones will result in periodic jitter that might worsen the DLL jitter performance and should be minimized. The ΔΣM is synthesized to run at 250MHz. Although the ΔΣM can receive a 10 bit input, the bit resolution (*m*) of the input control word (*K*) is only 5-7 bit to optimize the running speed of the modulator (50M-250MHz) and the attainable delay step resolution (~15ps), with lesser bits used at higher frequencies. Six additional bits are added to the modulator internal bit width to avoid arithmetic overflow, resulting in an internal bit width of 16. The layout of the ΔΣ modulator block is as shown below in Fig. 5.12 and it occupies a space of 255μm × 240μm. This block consumes less than 0.1mA for all the frequency range of operation.

The verilog code is synthesized using Synopsys DC compiler and constrained with a 4ns clock. There is little margin due to the technology used. The generated netlist has 197 cells.

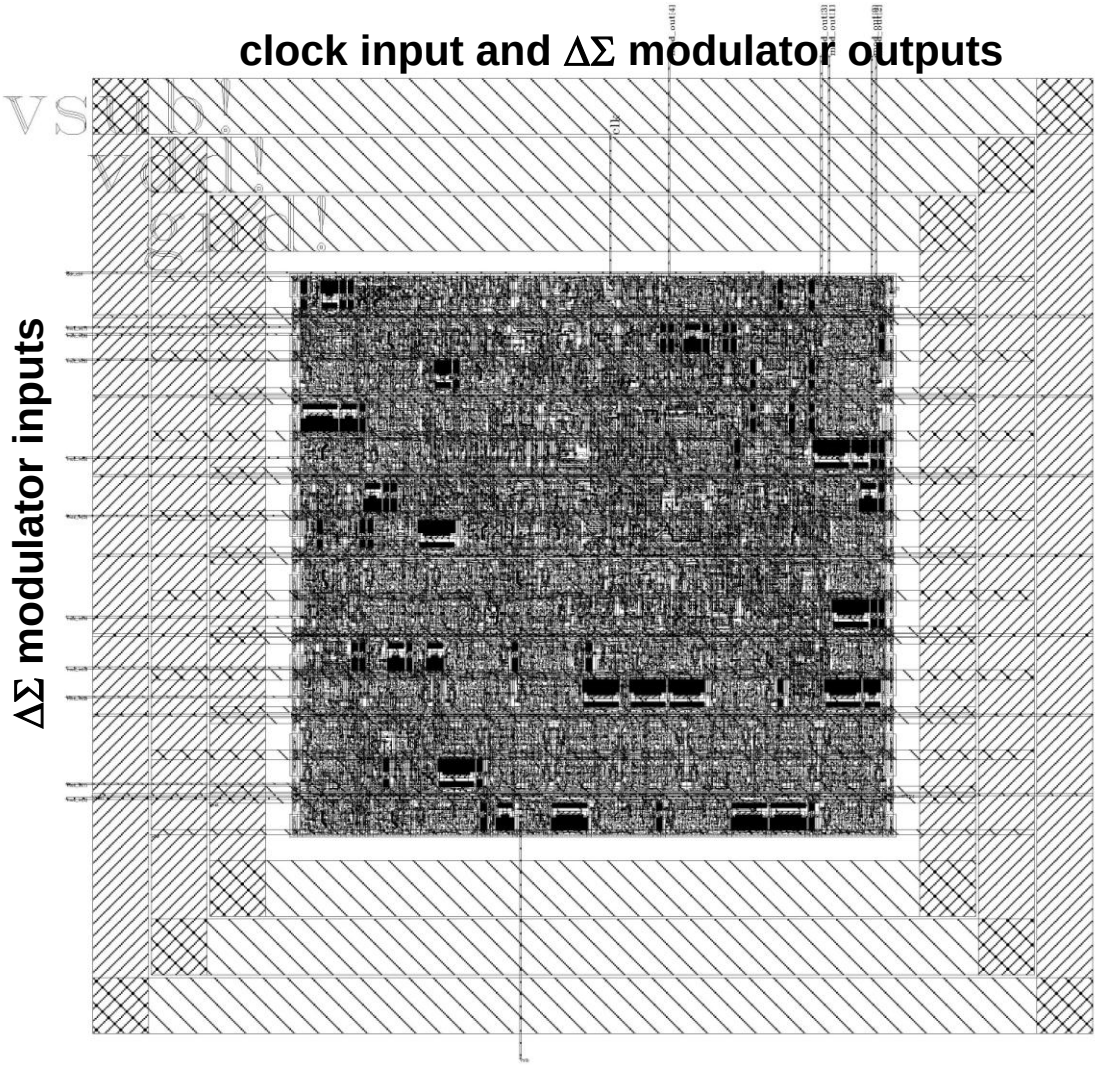


Figure 5.12: Layout of 1st order ΔΣ modulator

5.5. FSM block

The FSM flow chart for clock synchronization is shown in Fig. 5.13.

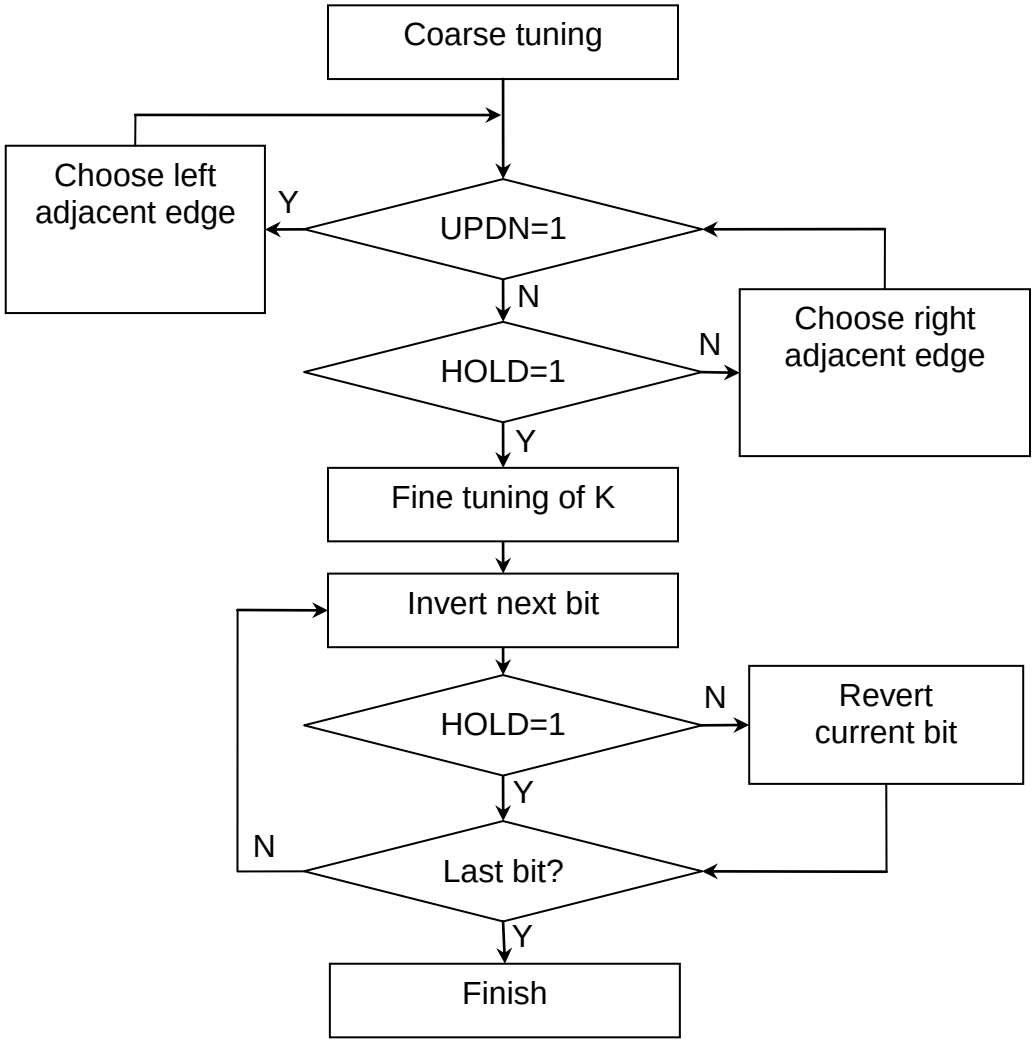


Figure 5.13: Summary of FSM flow chart

The clock synchronization is achieved in two tuning steps. During the coarse tuning step, the selected output phase (ϕ_{out}) is first initialized to ϕ_8 to speed up the coarse tuning. The ϕ_{out} is then compared with the incoming phase (ϕ_{in}) through CPD [16] to determine the desired action. The CPD split the full incoming clock period into ten intervals (A-J), and depending on the relative position of ϕ_{out} with respect to ϕ_{in}, UPDN or HOLD signal will be generated accordingly. While the HOLD signal is false, the UPDN signal of 1 or 0 will select either left or right adjacent clock edge relative to the current chosen output phase. Once the chosen ϕ_{out} falls within the interval E of the incoming clock period, the HOLD signal becomes true, and the FSM has identified ϕ_{out} that is the closest to ϕ_{in}. The entire operation for coarse tuning is also illustrated in Fig. 5.13 while the implementation and truth table is shown in Fig. 5.14 and Fig 5.15 respectively.

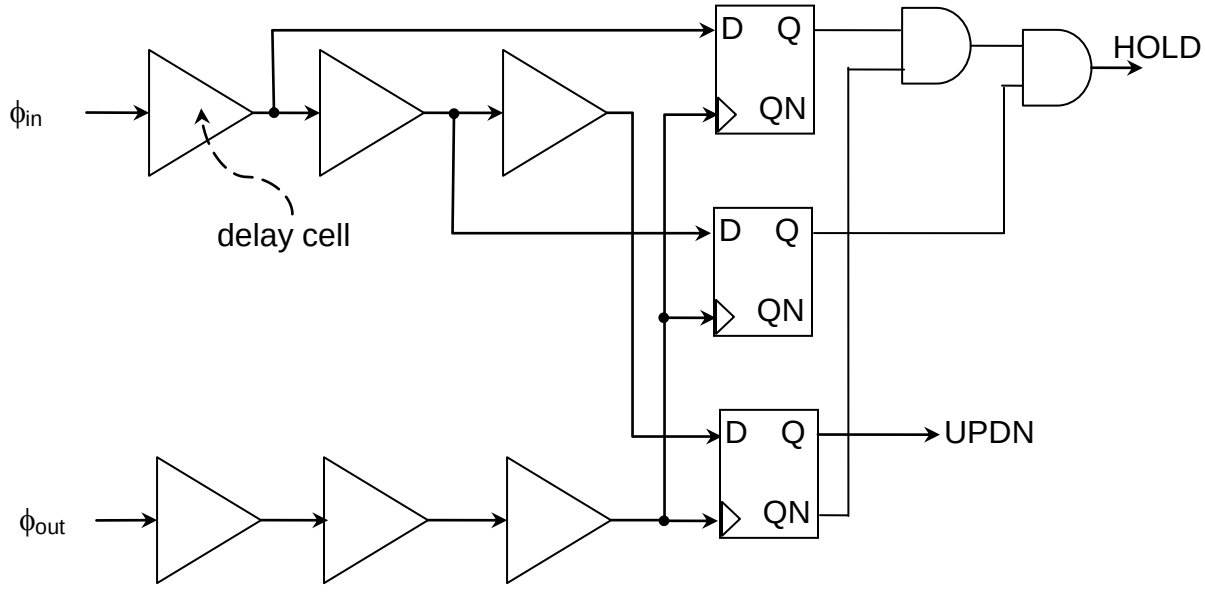
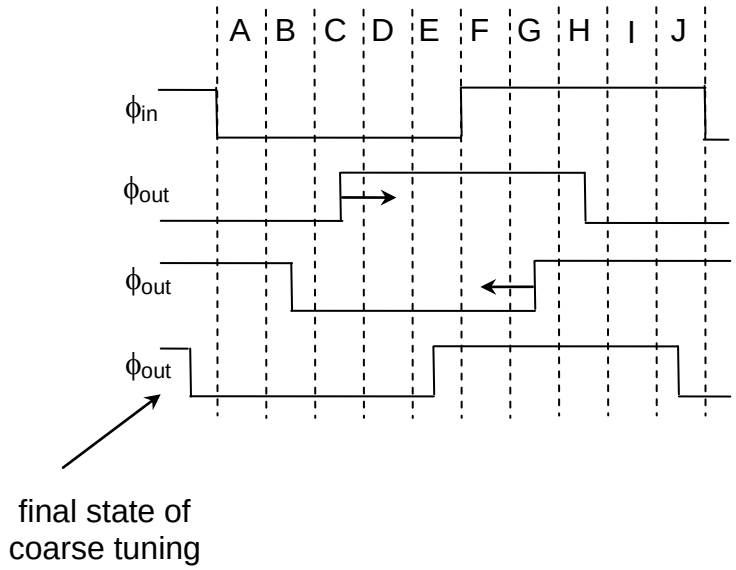


Figure 5.14: Coarse loop phase detector (CPD) implementation



CPD truth table

Region	UPDN	HOLD
A	0	0
B	0	0
C	0	0
D	0	0
E	0	1
F	1	0
G	1	0
H	1	0
I	1	0
J	1	0

Figure 5.15: Coarse loop phase detector (CPD) operation

After coarse tuning, the FSM will now enter the fine tuning step where the input control word *K* of the modulator will be updated in SA approach, starting from the MSB. This guarantees that the clock synchronization can be obtained after *m* steps. The bit is first inverted and the inversion would be kept if the resulting ϕ_{out} does not move into interval F. Otherwise, the FSM would revert the change and move onto the next LSB. The final ϕ_{out} should eventually synchronize to ϕ_{in} to within the step resolution of roughly 15ps. The layout of the FSM block is shown in Fig. 5.16 and measures 340μm by 330μm. The power consumed by this block is negligible as it operates at much lower frequency.

It is constrained with 10ns clock. The generated netlist contains 567 cells.

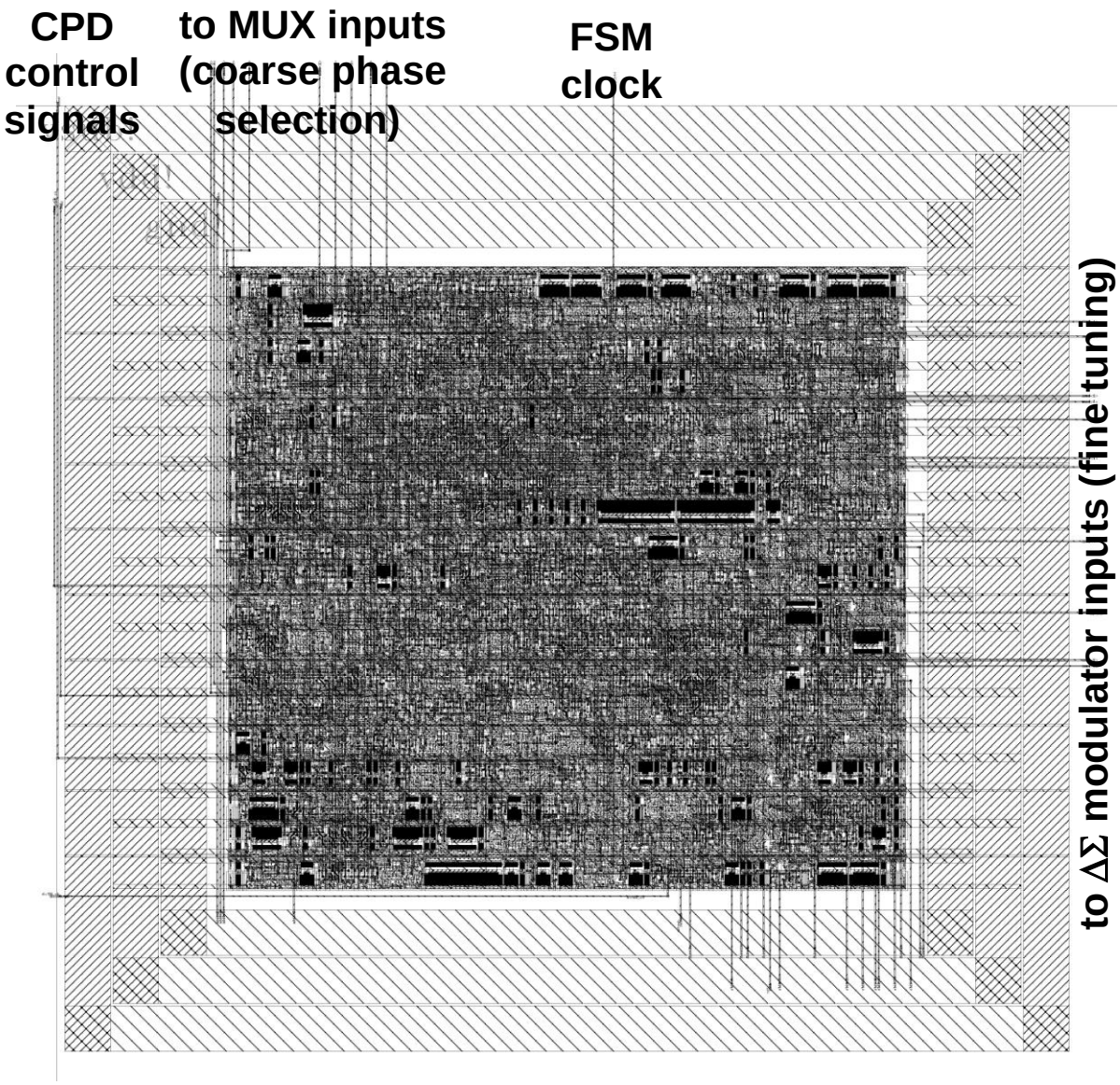


Figure 5.16: FSM block layout

The chip is fabricated in Austria Micro System 0.35μm CMOS Technology. The full die shown in Fig. 5.17 with the ΔΣ DLL core and clock synchronization architecture occupies active areas of 0.4mm^2 and 0.62mm^2 respectively. We adopt careful planning to separate the most noisy regions consisting of digital blocks (FSM, ΔΣ modulator) farthest from the cleaner analog blocks (loop filter, CP and master bias). The other regions are mainly occupied by bypass capacitors and a serial-to-parallel interface (SPI) block.

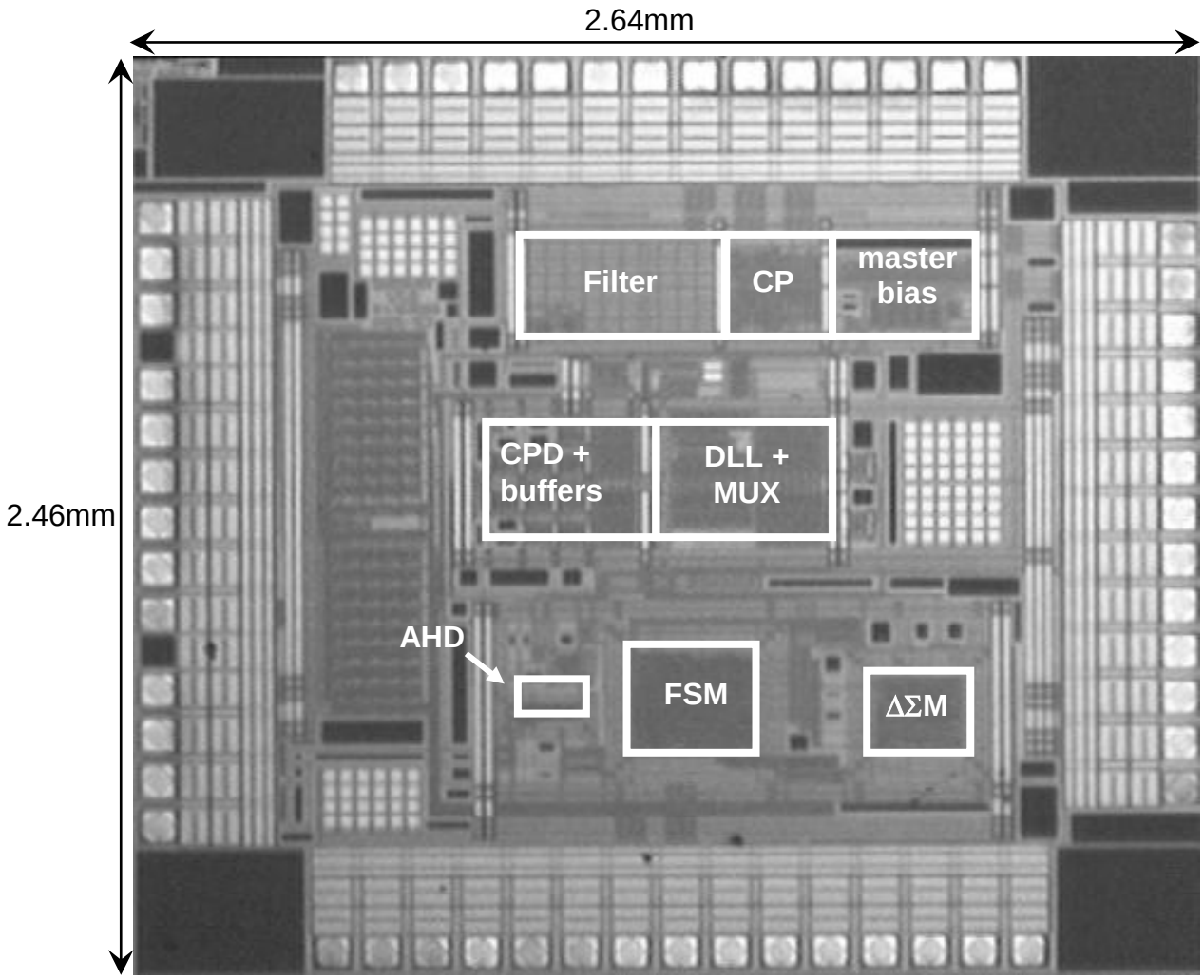


Figure 5.17: Die photo showing regions of clean analog, RF and noisy digital regions

5.6 Conclusion

The circuit implementation of the key blocks is documented in this chapter. Although most blocks are based on reference designs, there is some novelty when it comes to implementing two of the blocks. For example, the anti-harmonic detector (AHD) is unique. It is based on the relative positions of the clock edge based rather than reference clock based sampling. Clock based sampling techniques relies heavily on the 50% duty cycle of the sampled clock for accuracy while the proposed edge based method is less independent of the duty cycle.

Although the adaptive biasing technique [13] has been heavily studied and widely used, only 1^{st} order loop filter with single pole tuning is implemented. Our work is an extension of this technique and a second tuning pole is also utilized, to complement the 1^{st} order $\Delta\Sigma$ modulator used in the design.

CHAPTER 6. MEASUREMENT RESULTS

6.1. Test setup

Most of the measurements are taken using the Tektronix DPO71254 mixed signal oscilloscope. Clock source used for testing is a very clean reference, using Agilent 8133A Pulse Generator. It exhibits approximately 0.5ps rms jitter or 3.4ps pk-pk for most frequencies tested. The power supply used is Agilent E3631A. The test setup is briefly shown in Fig. 6.1.

The chip die is bond-wired and packaged into a QFN40 package which is soldered on the PCB for testing. The digital control signals are sent to the chip via a in-house developed SPI.

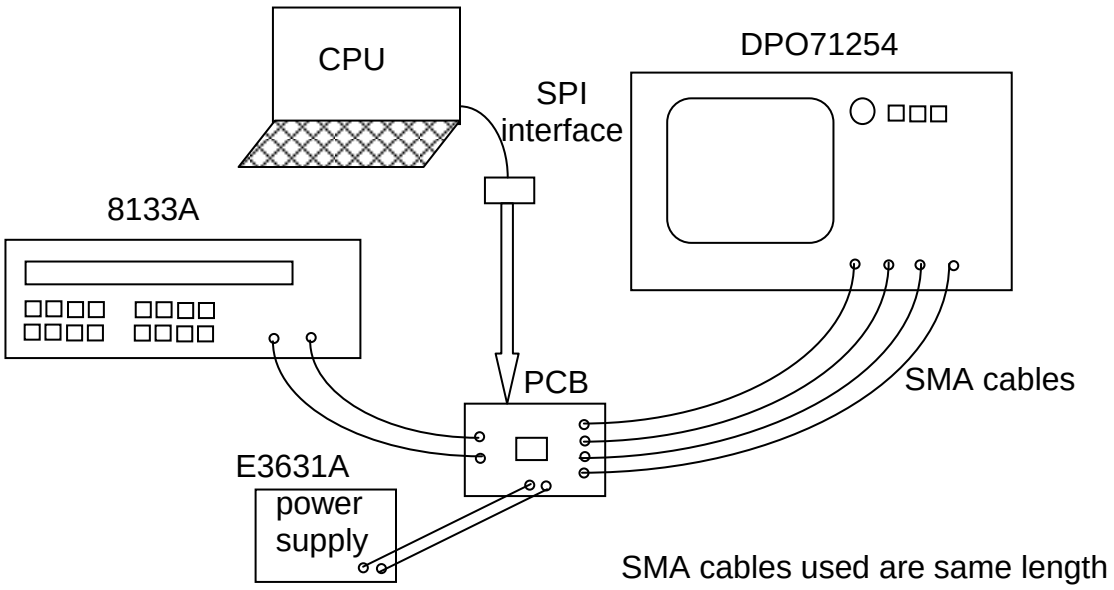


Figure 6.1: Simple test setup diagram

6.2. Timing diagram

The measured timing diagram of the various clock signals from the proposed DLL operating at 200MHz is shown in Fig. 6.2. As illustrated, the ϕ_{ref} and ϕ_{in} are not synchronized initially. Through the proposed architecture, the ϕ_{out} is then synchronized to ϕ_{in} as shown in Fig. 6.2. The observed duty cycle difference between the ϕ_{in} and ϕ_{out} is mainly due to the different travelling paths between the two signals. As expected, ϕ_{DLL} does not exhibit 50% duty cycle due to the dithered switching among a selected group of fixed clock phases. The functioning of the anti-harmonic lock detector is also verified by monitoring the loop filter voltage, which corresponds to the final achievable delay. As illustrated in Fig. 6.2, the AHD is functioning despite the non-50% duty cycle observed in ϕ_{DLL}.

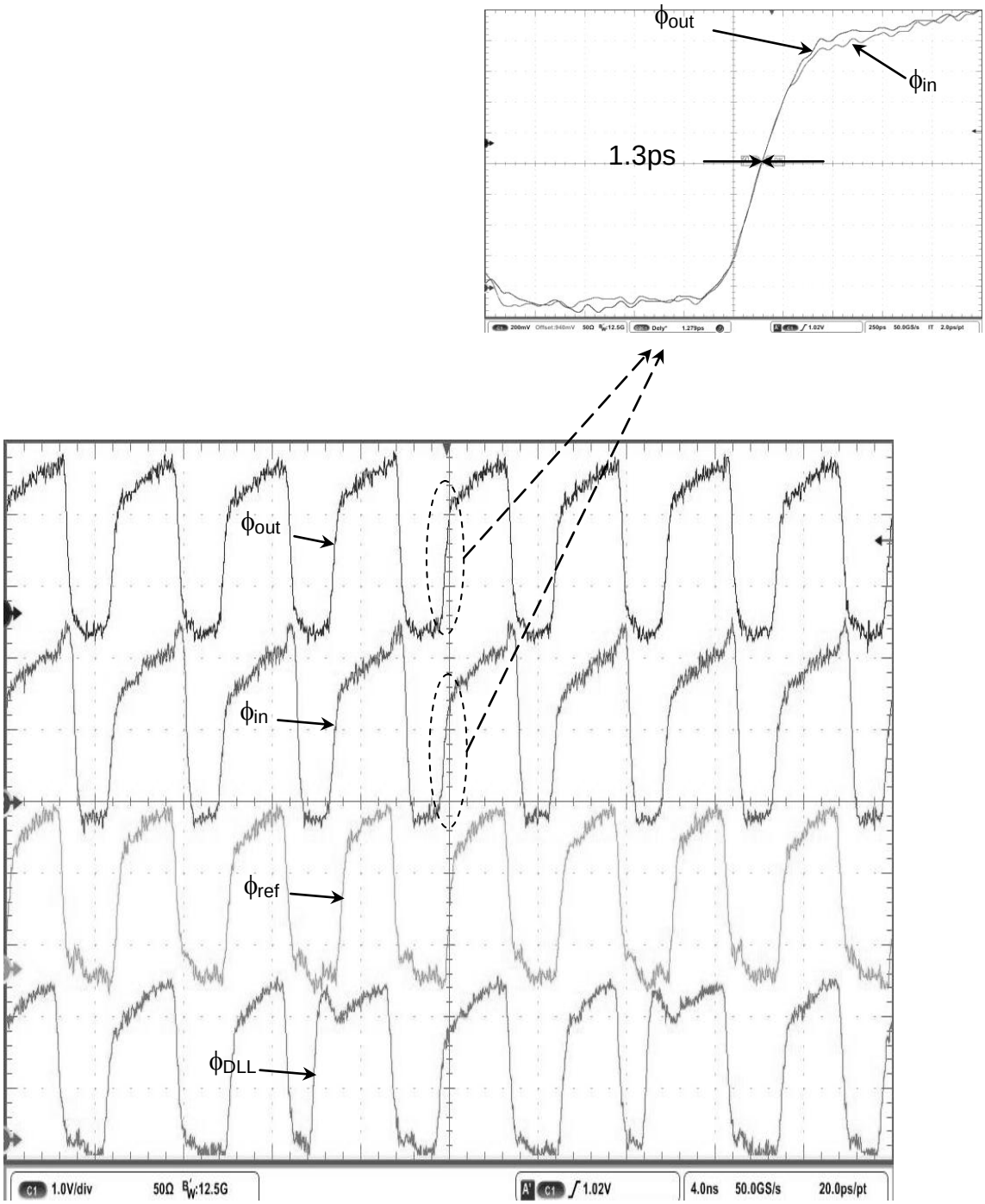


Figure 6.2: Timing diagram for DLL clock signals and synchronized ϕ_{out} with ϕ_{in}

6.3. Jitter performance

Fig. 6.3 to Fig. 6.5 shows the measured clock jitter of the DLL output (ϕ_{out}) at different frequencies and in detail at 50MHz, 200MHz and 250MHz. Jitter is taken for a few other input frequencies and plotted in Fig. 6.6. The jitter deterioration at lower frequency is expected due to the larger delay gain of the delay cell. The proposed architecture exhibits a rms jitter of 2.1ps and peak-to-peak jitter of 14.4ps at 200MHz.

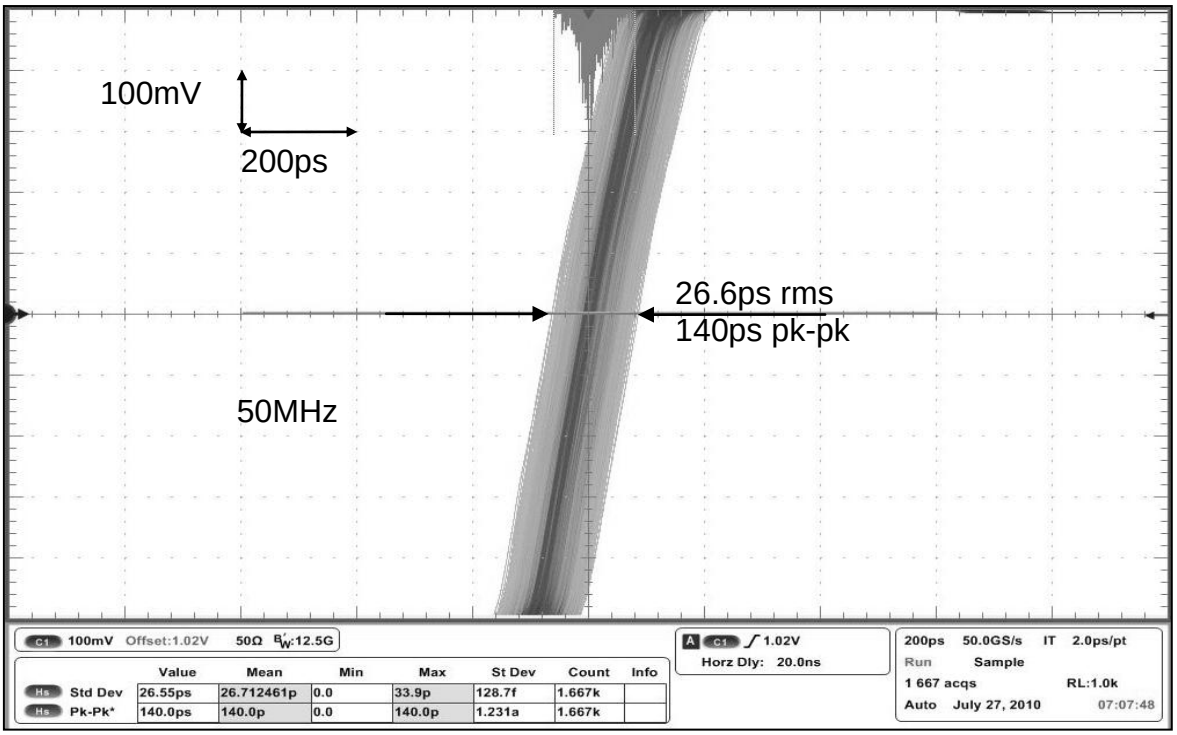


Figure 6.3: Jitter of output clock, ϕ_{in}, at input frequency=50MHz

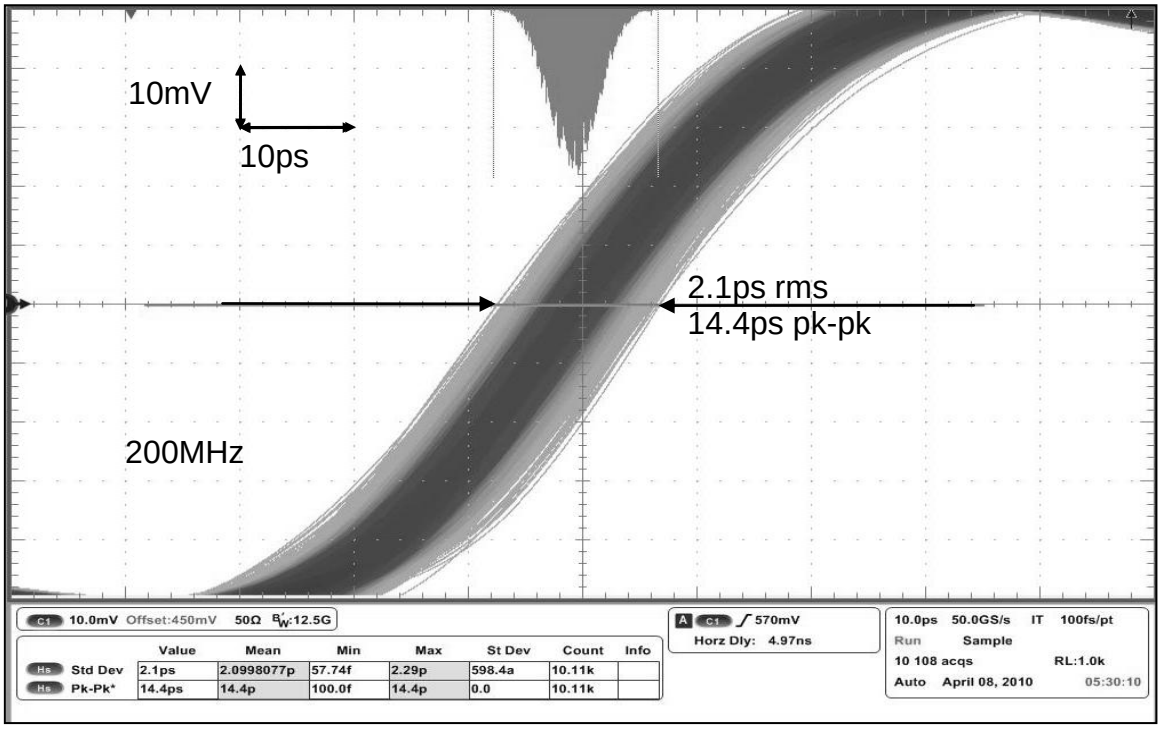


Figure 6.4: Jitter of output clock, ϕ_{in}, at input frequency=200MHz

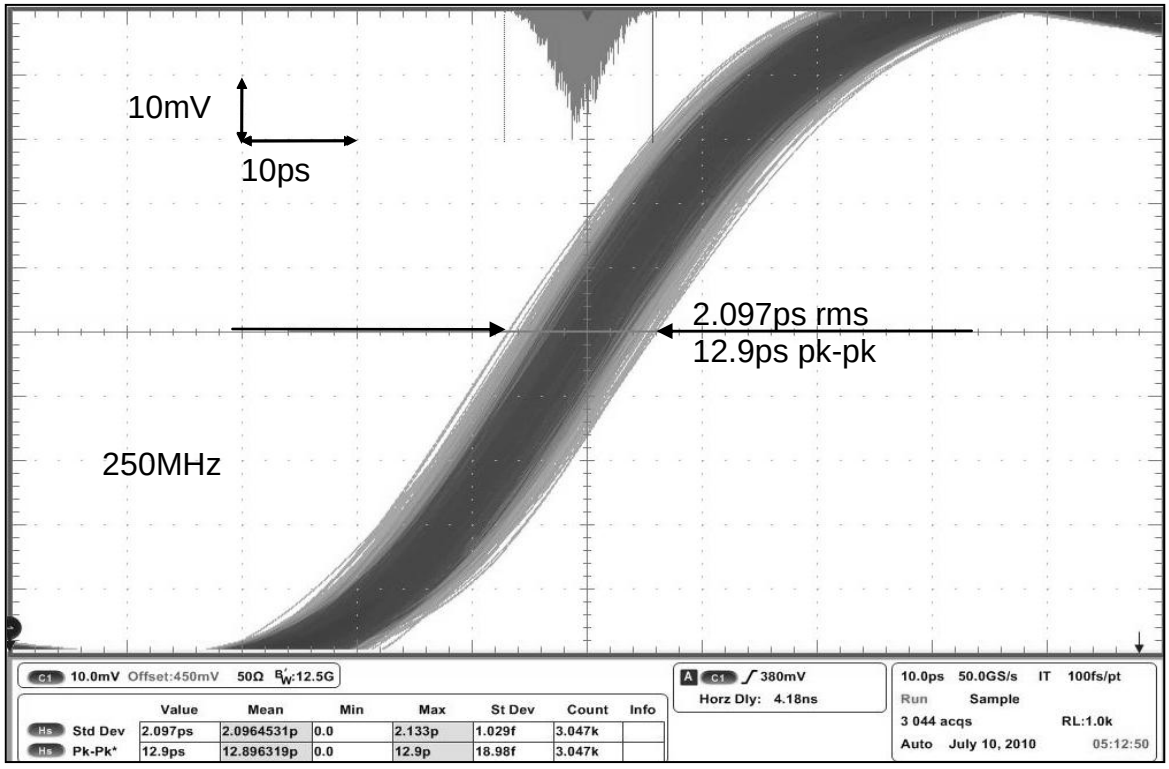


Figure 6.5: Jitter of output clock, ϕ_{in}, at input frequency=250MHz

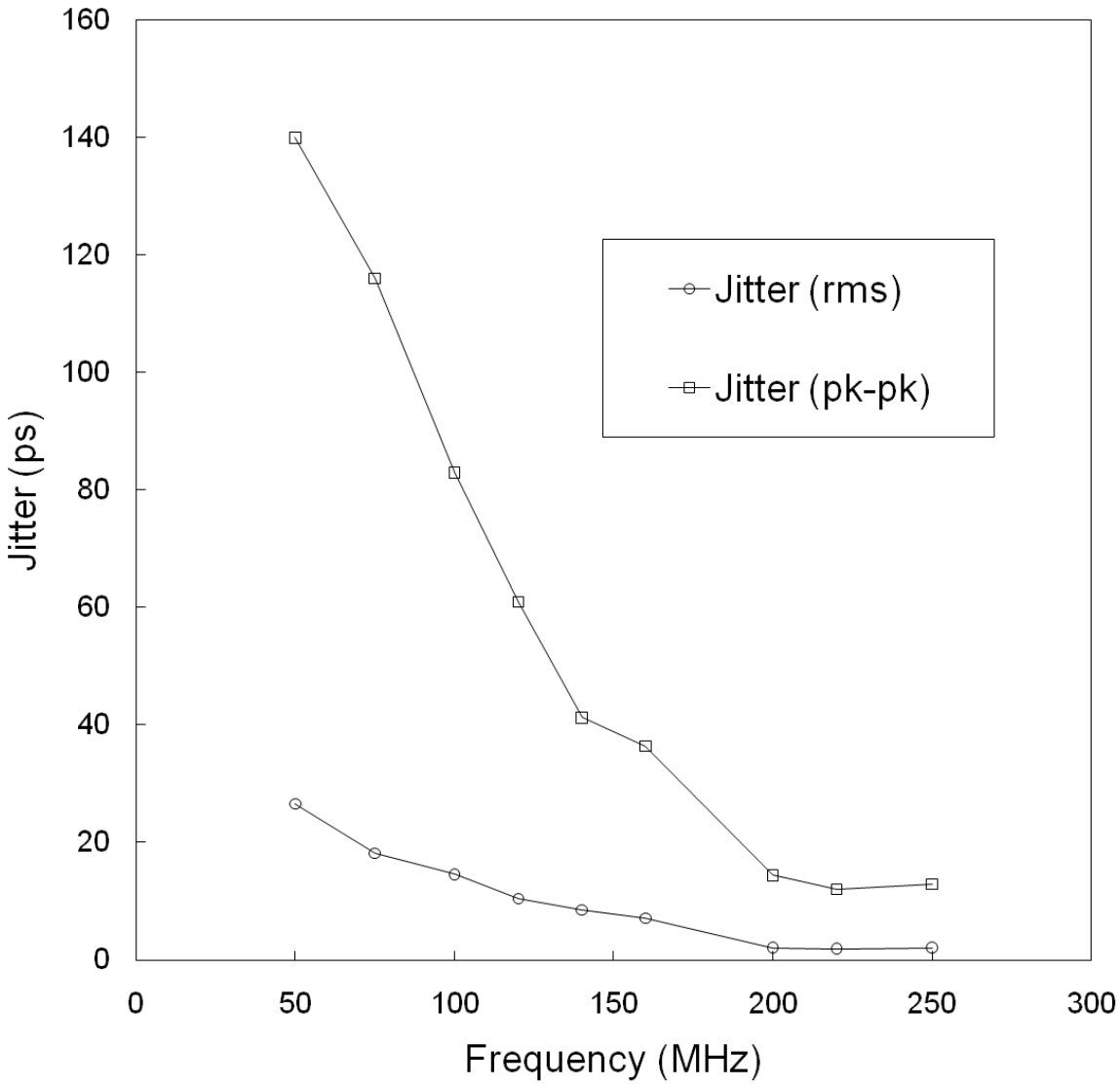


Figure 6.6: Measured jitter performance of output clock

6.4. Noise injection performance

The rms jitter performance is worsened to 15.8ps under noise injection via a 500mV pk-pk 70MHz sine wave coupled into the power supply as illustrated in Fig. 6.7. This results in a supply sensitivity of 0.18ps/mV, comparable to the reported results in [8,13], which uses the same architecture for the delay cell. The test setup is given by simple circuit in Fig. 6.8, where a waveform generator is used to perturb the power supply directly. Sine waves of different frequencies are later used for injection and the supply sensitivity is plotted against the injected noise frequency in Fig. 6.9. While low frequency noise has less impact on the clock jitter, high frequency noise however causes the largest deterioration of jitter. We can infer somewhat that the technique of noise shaping and filtering is effective in curbing high frequency noise in the circuit since the ΔΣ DLL exhibits relatively low jitter under normal operation. In the case where the ΔΣ DLL is incorporated into other larger silicon-on-chip (SOC) circuits consisting of extensive digital blocks where noise sources do not just come from the ΔΣ DLL, the use of an on-chip voltage regulator can help in rejecting noise from power supply to improve the jitter performance.

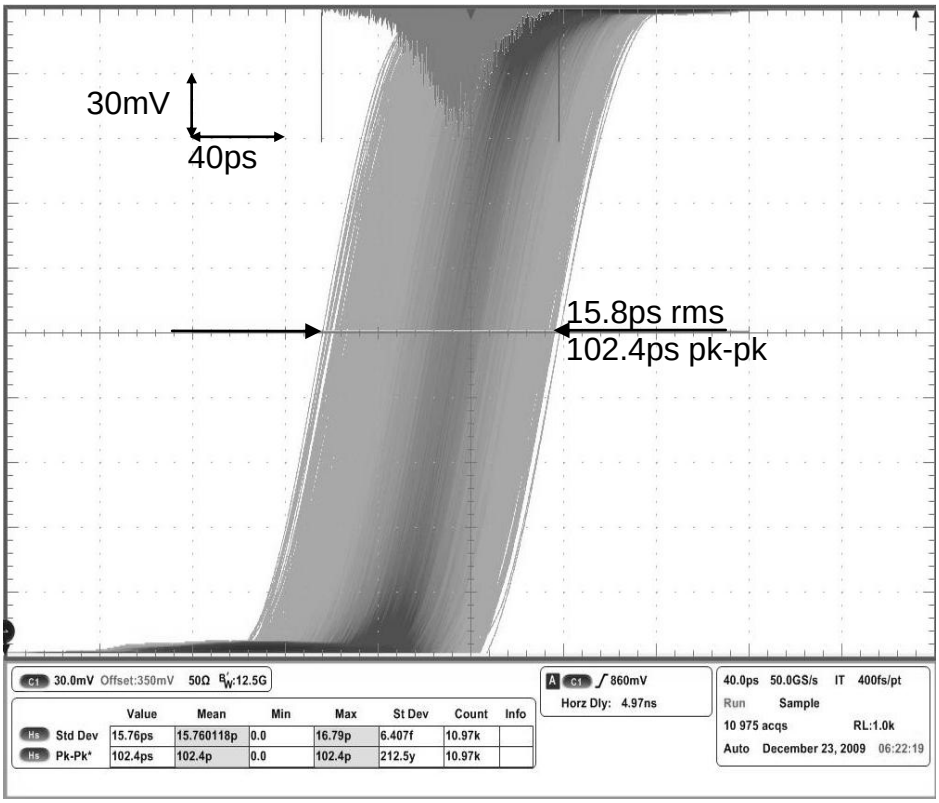


Figure 6.7: Measured jitter performance with noise injection

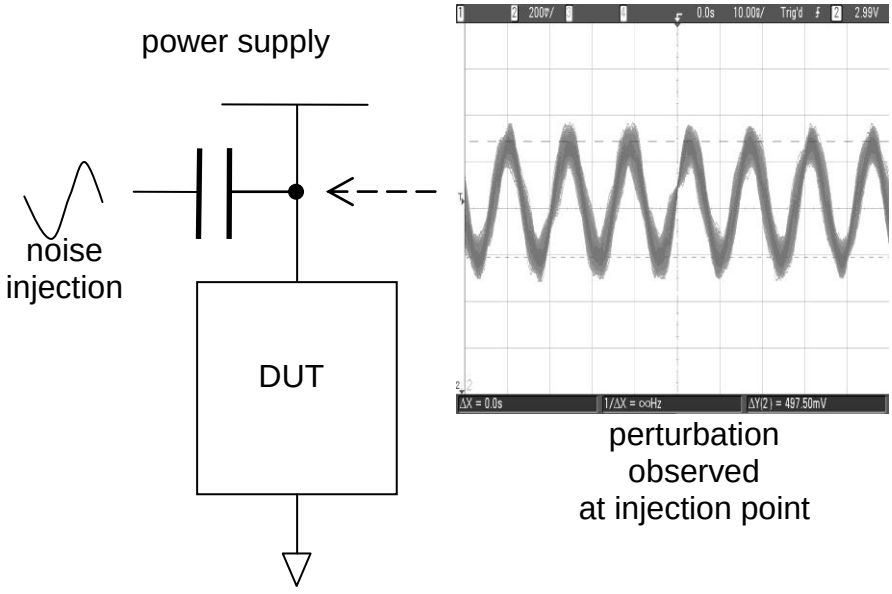


Figure 6.8: Test setup for noise injection

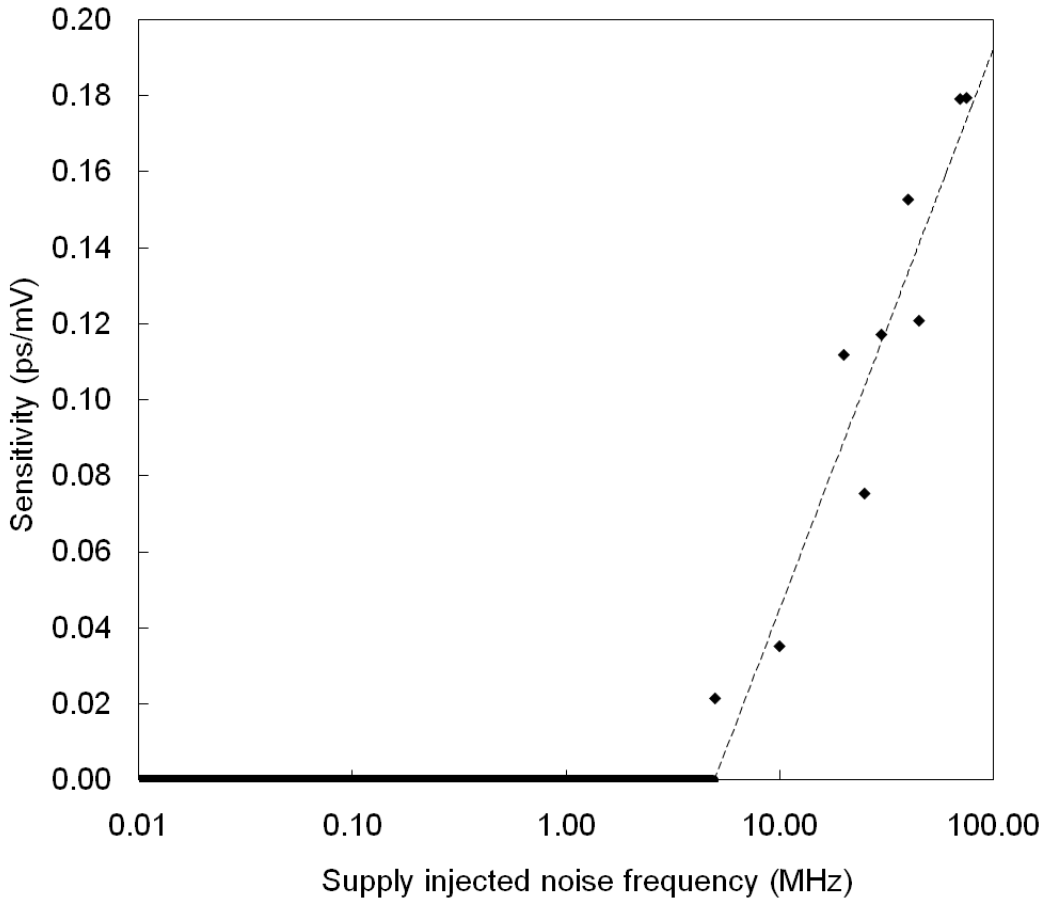


Figure 6.9: Effect on supply sensitivity from noise of various frequencies

6.5. Initial transient step response

The loop filter step response for 50MHz, 100MHz and 200MHz shown in Fig. 6.10 is normalized with respect to both the respective step voltage values and settling time for 200MHz input clock for ease of comparison. From Table I, it is clear that the deduced PM is not far from the ideal case where both poles shift if the adaptive bandwidth feature worked. If the second pole was not adjusted adaptively, it would have resulted in decreasing PM with increasing frequency, which is not the case of the derived PM. The settling time, which gives an indication of loop bandwidth, varies quite proportionally with the input frequency and it proves the workability of the adaptive bandwidth feature of the circuit.

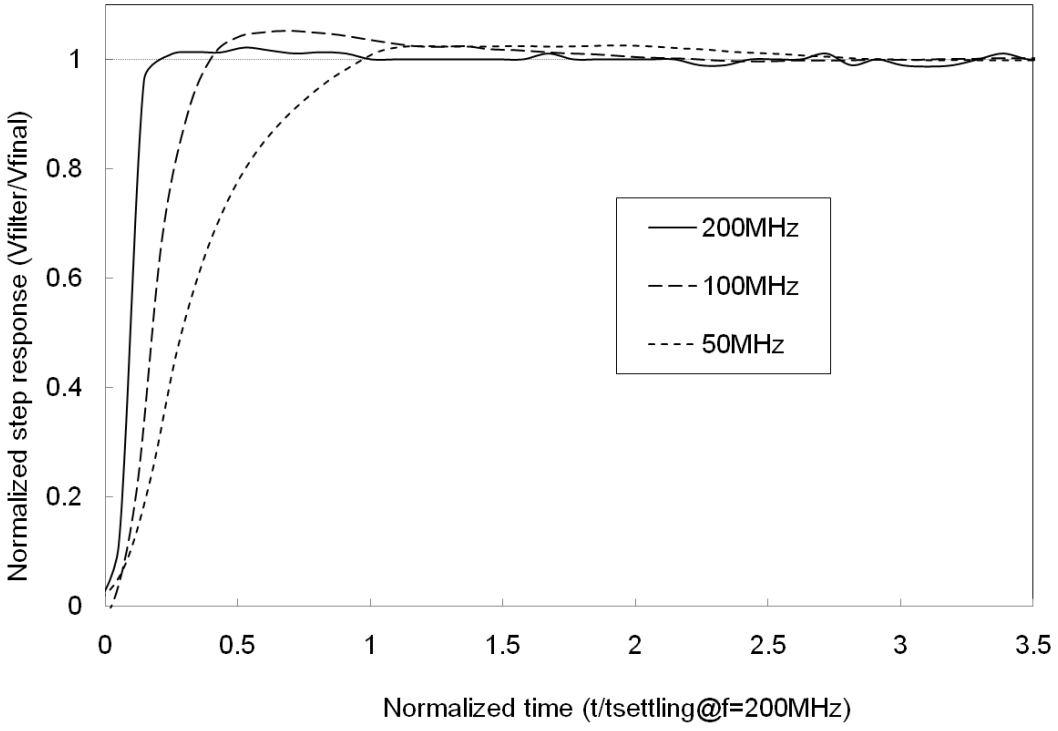


Figure 6.10: Normalized transient loop filter voltages at 50MHz, 100MHz and 200MHz.

Table 1: Phase margin (PM) and settling time comparison

frequency (MHz)	peak voltage (V)	final voltage (V)	overshoot (%)	deduced PM (^{o})	PM$^{\#}$ (^{o})	PM* (^{o})	normalized $t_{settling}$ ($t/t_{settling@f=200MHz}$)
50	1.903	1.893	2.49	68.2	65.4	65.4	4.81
100	1.796	1.782	5.14	64.5	65.4	51.7	2.21
200	1.548	1.547	2.17	68.7	65.4	38.6	1

resultant PM if both close loop poles adjust adaptively
* resultant PM if 2nd close loop pole did not adjust adaptively

The architecture can provide clock synchronization for input frequency ranging from 50MHz to 250MHz with the ΔΣ DLL core consuming only 6.9mA under 3V supply excluding the test buffers and clock synchronization circuit at 200MHz. The power consumption breakdown is presented in Table II. Note that the power consumption of the FSM is not included because they contribute quite insignificant power compared to analog blocks.

Table 2: Power consumption breakdown at input frequency=200MHz

ΔΣ DLL composition

block name	number of blocks	current per block (μA)	total current (μA)
delay cell	13	147.54	1918.02
5-to-1 MUX	1	100	100
charge pump	1	1677.86	1677.86
programmable charge pump current interface	1	196.72	196.72
phase detector	1	56	56
ΔΣ modulator	1	1518	1518
FSM	1	negligible	0
differential to single buffer	2	258	516
single to differential buffer	1	135	135
master bias	1	500	500
total current			6887.6

Significant amount of power is being consumed by the differential delay cells which offer better supply noise immunity. Significant power saving can be achieved if simple current-starved inverter type delay cell is used in the design, similar to [10-11], using a pseudo-differential type of configuration. However, that will be done at the expense of jitter performance.

The performance is summarized and compared with other ΔΣ DLL in Table III. The most cited reference for semi-digital DLL is also included in the comparison as a benchmark. Due to the difference in technology and operating frequencies compared to other ΔΣ DLL, it is difficult to give a fair comparison.

Table 3: Summary and comparison of performance

Reference	This work	[8]	[10]	[11]
Technology	0.35µm	0.8µm	0.13µm	0.18µm
Power Supply	3V	3.3V	1.2V	1.8V
Operating Frequency	50MHz to 250MHz	80kHz to 400MHz	0.5GHz to 1.5GHz	0.4GHz to 1.6GHz
ΔΣ Resolution	5 to 7 bits	4 bit[#]	11 bits	20 bits
Jitter	$2.1ps_{rms}$ @200MHz $0.15°_{rms}$	$11ps_{rms}$ @250MHz $0.99°_{rms}$	$4.1ps_{rms}$ @1GHz $1.5°_{rms}$	$0.4ps_{rms}$ @1.2GHz[*] $0.17°_{rms}$
Phase Span	2π	2π	2π	2π
Current Drawn	6.9mA @ 200MHz	30.9mA @ 250MHz	12.5mA @ 1GHz	2.93mA @ 1.2GHz
Core DLL Area Dual Loop Area	$0.4mm^2$ $0.62mm^2$	- $0.8mm^2$	$0.48mm^2$ -	$0.21mm^2$ -

[#] Inferred from 16 bit thermometer code used in fine tuning
[*] Measured as integrated rms phase noise from 10kHz to10MHz.

For example, the time step resolution of the proposed design is mainly limited by the running speed of the ΔΣ modulator, and should improve with more advanced CMOS technology.

Despite the older technology employed, the proposed design achieves better jitter performance and smaller power consumption and area compared to [10] by eliminating the PLL based MPG. It should be pointed out that the reported jitter in [11] is obtained by integrating the measured phase noise through limited bandwidth and is expected to be up to 12.7% worse than actual jitter [17]. By converting the time jitter into phase domain to remove the frequency dependency, our design achieves the best rms phase jitter of 0.15^{o}. We also include the performance from [8] for comparison due to its similar clock synchronization architecture, delay cell, operating frequencies and technology. As illustrated, the elimination of the analog interpolator helps achieving better jitter performance, smaller power consumption. Despite a technology node leap, our design is only ~25% smaller in area than [8] due to the large filter capacitors used in the adaptive bandwidth feature. However, the area ratio of the core loop DLL to peripheral loop of our design is 2:1 compared to about 1:3 for [8], which implies that the peripheral loop in our design require less additional blocks and complexity compared to [8].

CHAPTER 7. CONCLUSION

With the popularity of DLLs in clock synchronization systems, clock and data recovery and other wireline operations, demands for higher operating frequency has pushed for better performance requirements for DLLs in terms of clock jitter and fine timing resolution. In order to meet these needs, our research has led to the exploration a new class of semi-digital DLL, ΔΣ DLL. While there are not many variants of ΔΣ DLL [10-11] in the existing literature, most of these architectures could achieve sub-ps resolution while maintaining good jitter performance. Despite having eliminated the analog phase interpolator that is required in conventional semi-digital DLL, it has introduced an additional block in the form of a multi-phase generator (MPG). By making use of existing multi-phases in the feedback path, not only MPG is rendered unnecessary, noise performance does not suffer from the additional MPG jitter and power overhead.

A Matlab linearized system model is presented in chapter 4 to show the intuitive relationship of the core loop parameters with respect to the secondary loop parameters in terms of loop stability and its transient characteristics. A full functional model is later described to illustrate its actual clock synchronization operation.

Its circuit implementation in CMOS technology is described in detail in chapter 5. An extension of Maneatis [13] adaptive loop filter control idea is highlighted and a novel anti-harmonic lock detector is shown to deal with the non-50% duty cycle nature of the dithered feedback clock and the wide varying delay gain of the voltage controlled delay cell. The coarse and fine loop tuning is also explained in the implementation of the finite state machine. Finally,

the measurement results of the fabricated chip are documented in chapter 6, proving the functioning of proposed clock synchronization architecture and its sub-blocks.

A ΔΣ DLL capable of generating fractional delay of 15ps has been successfully fabricated in 0.35μm CMOS technology as a proof of concept. The proposed architecture is able to synchronize to clock frequency ranging from 50MHz to 250MHz and exhibit low jitter and relatively fine delay tuning resolution. It consumes only 20.7mW and exhibits rms jitter of 2.1ps.

Compared with the existing ΔΣ DLLs [10-11], no MPG is required. Delay resolution is not in the sub-picosecond range like the other ΔΣ DLLs due to the technology limitation on the operating speed of the ΔΣ modulator. If implemented in more advanced technology nodes, this ΔΣ DLL would show greater potential, and should be able to offer better performance in terms of operating frequency range and tuning resolution. Area savings are also expected. Despite usage of older technology, better absolute jitter of $2.1ps_{rms}$ is obtained compared to [10]. In terms of rms degrees, it is comparable to the state-of-the-art [11]. Moreover in [11], in order to achieve low jitter, a high frequency reference is required, due to the high frequency division ratio to push down quantization noise. Parallel structures of MUXs and phase detectors, and a multi-bit charge-pump are also introduced for additional FIR filtering, increasing the complexity of the architecture.

Significant progress has been made since the arrival of the semi-digital DLL [8]. Significant power savings of more than 4 times and 5 times better jitter performance is obtained. While area savings of 25% is not impressive despite a technology node leap, some hint of reduction in the system complexity of the peripheral loop is evidenced by the area ratio of the peripheral loop to the core loop. The area ratio of core loop to its secondary loop in the conventional DLL is 1:3 while in this work, it is 2:1, further highlighting the area savings and

reduction in system complexity with the elimination of the analog phase interpolator. The research in this thesis has clearly demonstrated advancement in the work of semi-digital and ΔΣ DLL in its application in clock synchronization.

REFERENCES

[1] W. Garlepp et al., "A portable digital DLL for high-speed CMOS interface circuits", *IEEE J. of Solid-State Circuits*, vol. 34, No. 5, pp. 632-635, May. 1999.

[2] J. H. Kim, Y. H. Kwak, M. Kim, S. W. Kim and C. Kim, "A 120-MHz-1.8-GHz CMOS DLL-based clock generator for dynamic frequency scaling," *IEEE J. of Solid-State Circuits*, vol. 41, no. 9, pp. 2077-2082, Sep. 2006.

[3] L. Wu and W. C. Black Jr., "A low-jitter skew-calibrated multiphase clock generator for time-interleaved applications," *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pp. 396-397, San Francisco, CA, Feb. 2001.

[4] X. Maillard, F. Devisch and M. Kuijk, "A 900-Mb/s CMOS data recovery DLL using half-frequency clock," *IEEE J. of Solid-State Circuits*, vol. 37, no. 6, pp. 711-715, Jun. 2002.

[5] B. Kim, T. C. Weigandt and P. R. Gray, "PLL/DLL system noise analysis for low jitter clock synthesizer design," *IEEE Proc. of Int. Symp. on Circuits and Systems (ISCAS)*, vol. 4, pp. 31-34, Jun. 1994.

[6] Y. Moon, J. Choi, K. Lee, D. K. Jeong and M. K. Kim, "An all-analog multiphase delay-locked loop using a replica delay line for wide-range operation and low-jitter performance", *IEEE J. of Solid-State Circuits*, vol. 35, No. 3, pp. 377-384, Mar. 2000.

[7] A. Efendovich, Y. Afek, C. Sella, and Z. Bikowsky, "Multifrequency zero-jitter delay-locked loop," *IEEE J. Solid-State Circuits*, vol. 29, no. 1, pp. 67–70, Jan. 1994.

[8] S. Sidiropoulos and M. A. Horowitz, "A semi-digital dual delay-locked loop", *IEEE J. of Solid-State Circuits*, vol. 32, No. 11, pp. 1683-1692, Nov. 1997.

[9] R. Kreienkamp, U. Langmann, C. Zimmermann, T. Aoyama, H. Siedhoff, "A 10-gb/s CMOS clock and data recovery circuit with an analog phase interpolator", *IEEE J. of Solid-State Circuits*, vol. 40, No. 3, pp. 736-743, Mar. 2005.

[10] P. K. Hanumolu, V. Kratyuk, G. Y. Wei, and U. K. Moon, "A sub-picosecond resolution 0.5-1.5GHz digital-to-phase converter," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 414-424, Feb. 2008.

[11] X. Yu, W. Rhee, Z. Wang, J. B. Lee and C. Kim, "A 0.4-to-1.6GHz low OSR with self-referenced multiphase generation," *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pp. 398-400, San Francisco, CA, Feb. 2009.

[12] K. Ogata, "Modern control engineering", Prentice-Hall, pp. 283-288, 1990.

[13] J. G. Maneatis, “Low-jitter process-independent DLL and PLL based on self-biased techniques”, *IEEE J. of Solid-State Circuits*, vol. 31, No. 11, pp. 1723-1725, Nov. 1996.

[14] D. J. Foley and M. P. Flynn, “CMOS DLL-based 2-V 3.2-ps jitter 1-GHz clock synthesizer and temperature-compensated tunable oscillator”, *IEEE J. of Solid-State Circuits*, vol. 36, No. 3, pp. 417-423, Mar. 2001.

[15] S. R. Norsworthy, “Effective dithering of sigma-delta modulators,” *IEEE Proc. of Int. Symp. on Circuits and Systems (ISCAS)*, vol. 3, pp. 1304-1307, 1992.

[16] S. J. Bae, H. J. Chi, Y. S. Sohn and H. J. Park, “A VCDL-based 60-760-MHz dual-loop DLL with infinite phase-shift capability and adaptive-bandwidth scheme”, *IEEE J. of Solid-State Circuits*, vol. 40, No. 5, pp. 1119-1129, May. 2005.

[17] M. Ishida, K. Ichiyama, T.J. Yamaguchi, M. Soma, M. Suda, T. Okayasu, D. Watanabe, K. Yamamoto, “A programmable on-chip picosecond jitter measurement circuit without reference clock input,” *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pp.512-513, San Francisco, CA, Feb., 2005.

PAPERS RELATED TO DISSERTATION

[1] S.-J. Cheng, L. Qiu, Y.Zheng, and C.-H. Heng, “50-250MHz ΔΣ DLL for Clock Synchronization”, *IEEE J. of Solid-State Circuits*, vol. 45, No.115, pp. 2445-2456, Nov. 2010.