Cairo University
Faculty of Engineering
Electronics and Electrical Communications Engineering

A Massive MIMO Digital Beamforming mmWave Transmitter for 5G

A Graduation Project Report Submitted to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree of

Bachelor of Electronics and Electrical Communications Engineering.

Submitted By

Ahmed Mohammed Mohammed

Ahmad Muhammad Sayed

Ahmed Mohammed Sherif

Radwa Abdelazim Abdelfatah

Yasmin Yosri Sadek

Zeyad Tarek Elsefsafi

Under Supervision of

Dr. Hassan Mostafa   Dr. Hassan Abou Shady

July 2023

Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

Abbreviation	**Definition**
BW	Bandwidth
CIFB	Cascaded Integrators with Distributed Feedback
DAC	Digital to Analog converter
DUC	Digital Up Converter
EFB	Error Feedback
EVM	Error Vector Magnitude
FIR	Finite Impulse Response
MASH	Multi-stage Noise shaping
MIMO	Multiple Input Multiple Output
NTF	Noise Transfer Function
OSR	Over Sampling Ratio
PVT	Process, Voltage and Temperature
RF	Radio Frequency
SNR	Signal to Noise Ratio
STF	Signal Transfer Function

Acknowledgments

We would like to express our sincere gratitude and appreciation to Dr. Hassan Mostafa, Dr. Hassan Abo Shady, and Dr. Islam Ishra for their valuable support and guidance throughout the course of our project. Their expertise and mentorship have been instrumental in helping us achieve our project objectives and overcome various challenges.

We would also like to thank Eng. Ahmed Ghoneim, Eng Bahy Yahia and Eng Ziad Saeed for their insightful feedback, encouragement, and constructive criticism which have greatly enriched our work and inspired us to strive for excellence. I am also grateful for their generosity in sharing their knowledge, resources, and time with us, which have been crucial to the success of our project.

Abstract

This thesis provides the complete work on a massive MIMO digital beamforming mmWave transmitter for 5G to be interfaced to one-bit DAC. The architecture implementation is done through three main stages. The Simulink implementation, RTL implementation and ASIC implementation using 130nm CMOS technology through two flows open-source and proprietary software to build the multi-chain transmitter. The final product is two chains transmitter with output sampling rate of 400MHz, bandwidth of 6.25MHz, area of 0.1387 mm^2 and power of 2.245 mW.

Chapter 1: Introduction

The transmission of signals with high capacity, high coverage, reduced interference and directed beam to specific target with narrow band are important characteristics of 5G. Unlike the traditional cellular networks which send the signals in all directions which can lead to signal interference, reduced coverage and inefficient use of available spectrum as shown in figure 1.1.[1]

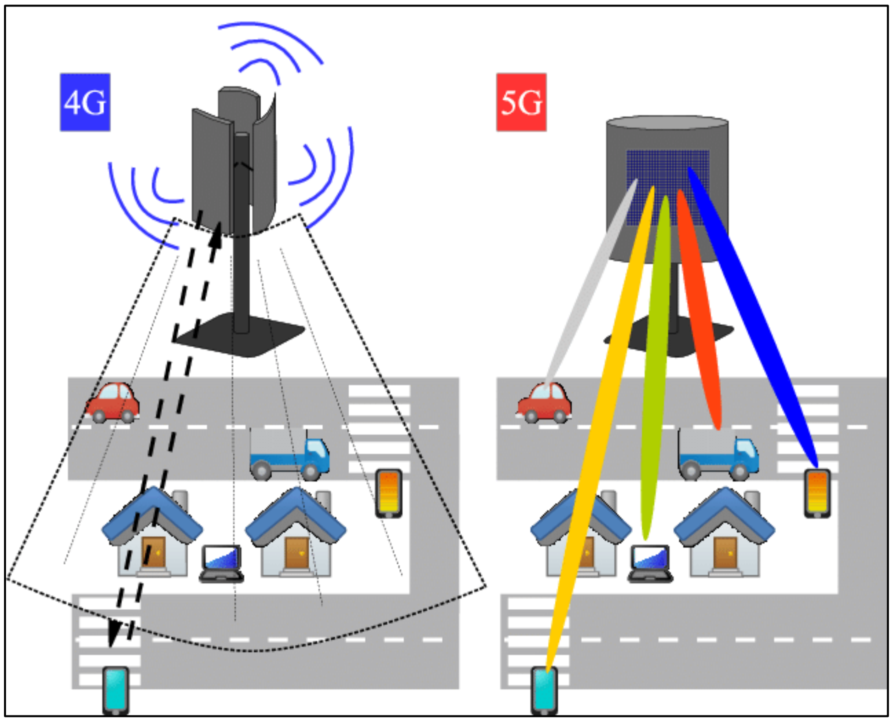


Figure 1.1 4G vs 5G Transmission Operation [1]

5G aims to overcome these limitations by directing the energy accurately in the desired direction to improve data rate, reliability and the utilization of resources.

To implement a design for 5G transmitter it is a must to perform the operation of transmitting several beams with high directivity. Beamforming technique for transmission provide these characteristics for the transmission operation. Beamforming uses the MIMO technology using multiple antennas in the transmitter in order to control the directivity.

This thesis provides a detailed explanation of the implementation of a massive MIMO digital beamforming mmWave Transmitter for 5G through three stages:

- System Modeling Using Simulink
- RTL Implementation
- ASIC Implementation using two flows (open source and proprietary software)

Chapter 2:Beamforming

2.1 Introduction

Beamforming is a particular processing technique for signals that allow for directional transmission or reception. 5G beamforming allows Verizon (wireless) to make 5G connection more focused toward a receiving device. For example, a typical 5G small cell that does not employ beamforming during its multiple-input multiple-output (MIMO) transmission will not be able to narrowly concentrate or focus its transmitted beams to a particular area. With beamforming, the small cell can focus the transmission on a particular direction towards a mobile device such as a cell phone, laptop, autonomous car or IoT node. This improves the efficiency overall of the network and saves energy.

The operation of beamforming is done by sending a certain signal several times through different antennas. Each antenna chain contains a certain phase shift so that a constructive interference between these signals happens only in the direction of the steering angle. Steering angle is the angle of the desired maximum transmission power. In other directions destructive interference occurs between the signals leading to low transmission power in these directions.[2]

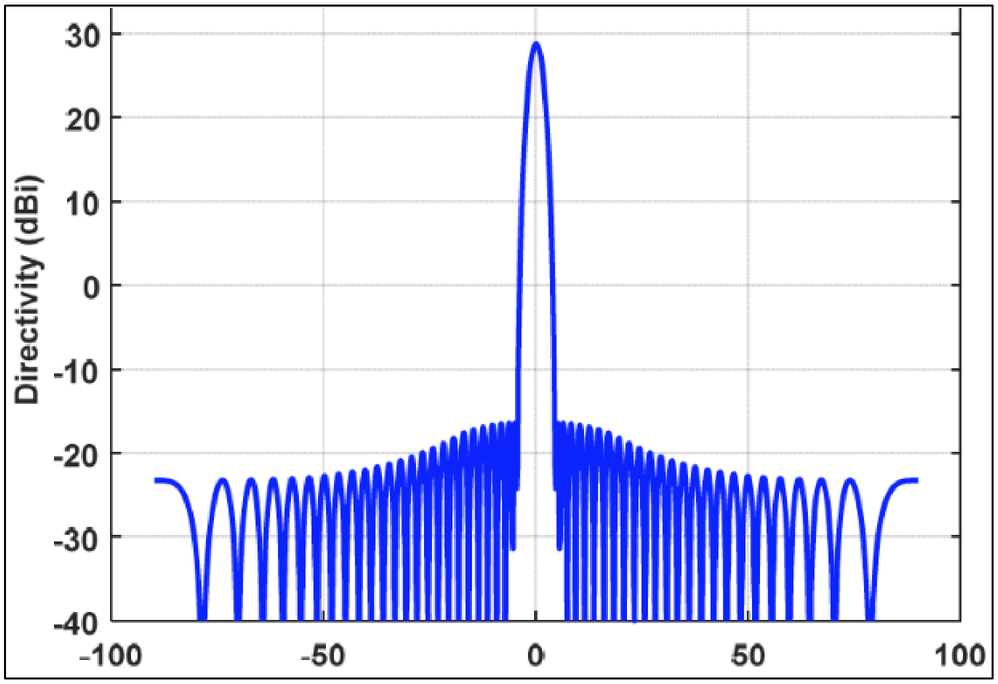


Figure 2.1 Beamforming Output Directivity Corresponding to Steering Angle

It can be noticed from figure 2.1 that the signal has high gain in the direction of steering angle which is zero here (broadside).

The electrical phase shift in each antenna is determined depending on steering angle and continuously updated based on receiver location.

In following equation, the phase shift for each antenna chain is computed depending on the desired steering angle and the distance between each two adjacent antennas.

$$x = \mathrm{d} * \sin\theta_s$$

This distance x can be set in relation to the wavelength:

$$\frac{360^0}{\Delta\varphi} = \frac{\lambda}{x}$$

Both equations together are the solution. We get phase different between each two adjacent antennas as following:

$$\Delta\varphi = \frac{360^0\, d\sin\theta_s}{\lambda}$$

Equation 2.1 Phase Shift Calculations

Where $\Delta\varphi$ refers to phase shift between two successive elements, d refers to distance between the radiating elements and θ_s is the beam steering.[3]

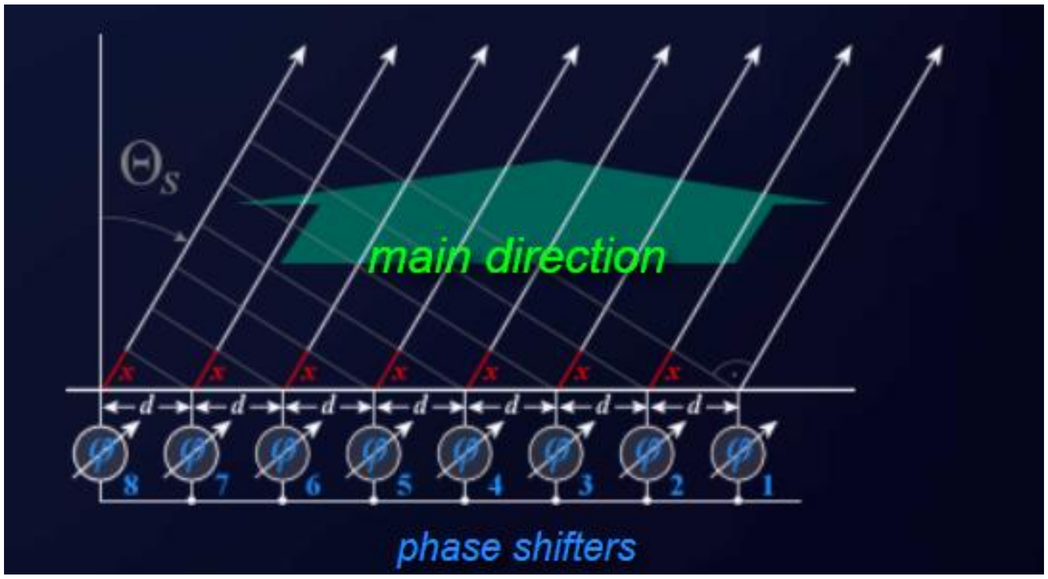


Figure 2.2 The Beamforming Operation Transmission [3]

Figure 2.2 Shows graphically the parameters used in equation 2.1.

2.2 Types of Beamforming

Beamforming has more than one type. In this section, two types of beamforming are taken into consideration analog and digital beamforming.

2.2.1 Analog Beamforming

Analog beamforming is done by applying modulated RF single input signal at the transmitter to a set of different phase shifters leading to different antennas. The difference in phase between the signals determines the directivity of the beam.

The phase shift in analog beamforming is done in the RF domain at the end of the chain before transmission. So, only one processing chain is used in the system feeding the multiple phase shifters and antennas. As there is only one chain in this system, the analog beamforming has low area, cost and power consumption. On the other hand, analog beamforming is less adaptive to changing conditions than digital beamforming which would be covered later in this chapter leading to lower accuracy as a result of phase shifting at RF domain.

The architecture in figure 2.3 Shows a block diagram for analog beamforming.

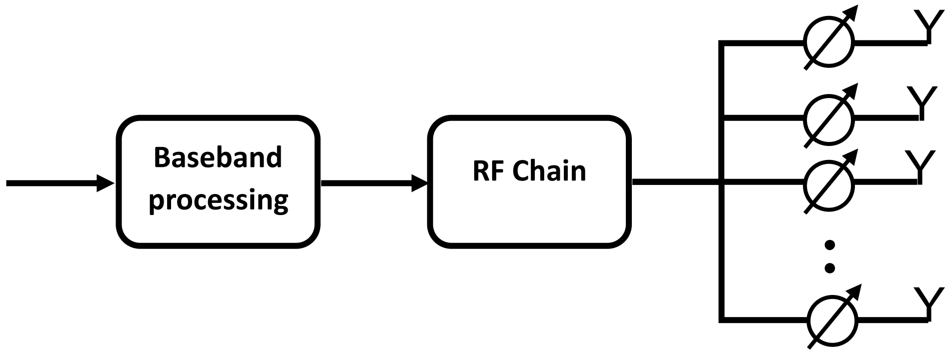


Figure 2.3 Analog Beamforming Architecture

2.2.2 Digital Beamforming

Digital beamforming is done by applying baseband single input signal at the transmitter to a set of different phase shifters leading to different processing chains and different antennas. The difference in phase between the signals determines the directivity of the beam.

Digital phase shifters are used to achieve the phase shift in digital beamforming which is done in the digital domain before the DACs in each chain. So, multiple chains are used in the system feeding the multiple phase shifters and antennas. As there are multiple chains in this system, the digital beamforming has high area, cost and power consumption comparing to analog beamforming. As phase shift is done in digital domain, higher accuracy can be achieved than that of analog beamforming.

The architecture in figure 2.4 Shows a block diagram for digital beamforming.

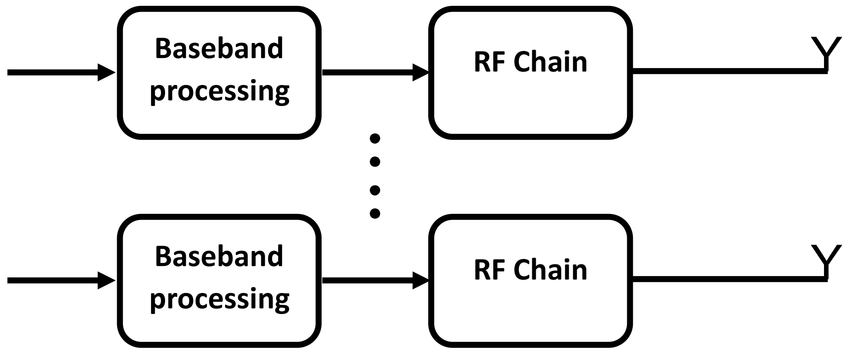


Figure 2.4 Digital Beamforming Architecture

Chapter 3: System Modeling

3.1 Introduction

In this chapter, an architecture for the digital beamforming transmitter is implemented using MATLAB Simulink. For digital beamforming system as discussed above it consists of multiple phase shifters and multiple antennas for the transmitting process. But for this transmitter more blocks are implemented in the multi-chain transmitter architecture. Each of the chains consists of a digital phase shifter, two interpolation filters, two sigma delta blocks and a digital up converter to be interfaced later with one bit digital to analog converter (one bit DAC) which is out of our scope for now. The single chain block diagram is shown in figure 3.2. Each block would be discussed in this chapter showing its role in the system and its design details.

MATLAB Simulink implementation of the transmitter architecture is considered the first step in the design flow. Which is an important step before entering the digital design flow as in this step the details of each block are determined. The goal of this step to convert the black boxes of our blocks shown in the architecture to well defined blocks that fulfil the specifications required in this work by testing different types of phase shifters, filters, sigma delta and DUC architectures determining their order and way of implementations that would achieve the best outcomes.

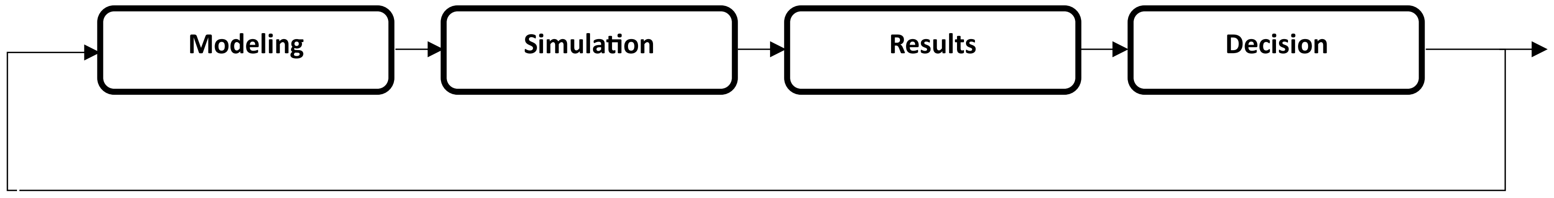


Figure 3.1 System Modeling Flow

In next sections the flow of choosing the final architecture is shown.

3.1.1 System Constrains

For this implementation for the transmitter architecture some specifications are required for the output signal to noise ratio (SNR) and the output bits width. These specifications would be shown in this section.

The output signal to noise ratio (SNR) it is required to reach a value exceeds 50 dB. Also, for the output signal it should be a one-bit signal as the system would be interfaced with one bit DAC in the future work for its linearity and simplicity which would greatly reduce analog complexity. These two constrains on the output signal would be important in making some design decisions for the over sampling ratio for the filter, the filter order and the digital sigma delta type and order.

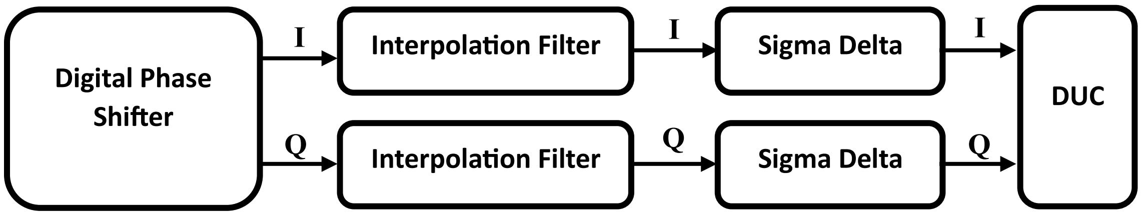


Figure 3.2 Single Chain Architecture

The next sections explain each of the single chain architecture blocks in details.

3.2 Phase Shifter

Phase shifters are considered the backbone of any beamforming system. Transmitting the signal through multiple antennas with different phase shifts to control the directivity and perform the beamforming operation as explained in chapter 2 is the role of the phase shifters in our system. Complex weight multiplier (CWM) digital phase shifters are used in this work to achieve the phase shift in each chain.

3.2.1 Complex Weight Multiplier

Complex weight multiplier technique is the phase shifting technique used in this work. CWM is mainly built on the equation 3.1[2]

$$\begin{pmatrix} {I_k}' \\ {Q_k}' \end{pmatrix} = \begin{pmatrix} \cos(k\varphi) & \sin(k\varphi) \\ -\sin(k\varphi) & \cos(k\varphi) \end{pmatrix} \begin{pmatrix} I \\ Q \end{pmatrix}$$

Which corresponds to

$$I_k' = I\cos(k\varphi) + Q\sin(k\varphi)$$

$$Q_k' = -I\sin(k\varphi) + Q\cos(k\varphi)$$

Equation 3.1 CWM phase shifting equations

Where I and Q refer to the inputs to the transmitter system modulated in any modulation technique (e.g., QPSK, QAM, …. Etc.), k refers to the chain number in the multi chain system, I_k' and Q_k' refers to the phase shifter outputs and φ is the phase shift angle between two adjacent chains in the architecture computed in chapter 2.

The output of the phase shifter I_k' and Q_k' are phase shifted version of the input I and Q with phase difference equals to kφ

3.2.2 Phase Shifter Simulink Implementation

In this section a Simulink design is implemented for the CWM phase shifting technique. CWM equation can be described in the next block diagram that would be implemented in Simulink.

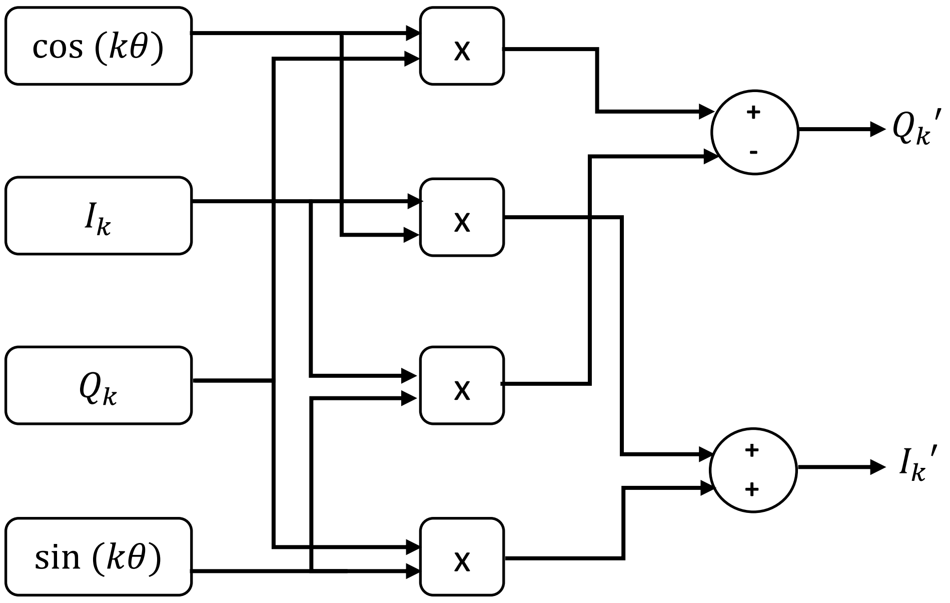


Figure 3.3 CWM phase shifter block diagram

The block diagram in figure 3.3 is implemented in MATLAB Simulink. The two plots in figure 3.4 show the operation result of the phase shift block where the first plot is input cosine signal with amplitude 0.6 and the outputs are a 90-degree and 180-degree phase shifted versions of input as shown in figure 3.4.

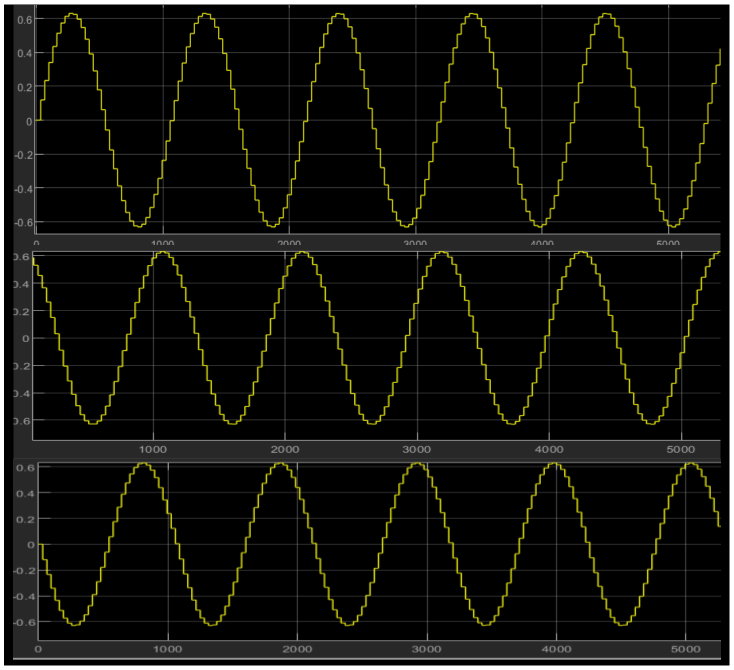


Figure 3.4 Phase Shifters Input 90-Degrees and 180-Degrees Outputs Plots

3.3 Interpolation Filter

3.3.1 Interpolation (Oversampling)

Oversampling refers to the process of sampling a signal at a rate that is significantly higher than the Nyquist rate, which is the minimum sampling rate required to accurately capture the bandwidth of the signal. By oversampling, the sample rate (f_s) is much greater than the Nyquist rate (f_N), which is twice the bandwidth of the signal ($2BW$).

Advantages of using oversampling:

1. Improved Resolution: Oversampling allows for higher resolution in analog-to-digital conversion (ADC). By sampling the signal at a higher rate, more samples are taken within each cycle of the signal, providing finer granularity in capturing the signal's amplitude. This can result in increased effective bit depth and improved accuracy in digitizing the signal.

2. Simplified Anti-Aliasing Filtering: Oversampling can simplify the design of the anti-aliasing filter. The high oversampling ratio reduces the required filter's complexity and steepness of the cutoff, which can ease the design constraints and reduce the cost of implementing the filter.

3. Increased Signal-to-Noise Ratio (SNR): Oversampling can improve the SNR of the system. Since the quantization noise in an ADC is spread over a larger frequency range due to oversampling, the noise power within the bandwidth of interest is reduced. This effectively increases the SNR, resulting in improved signal fidelity.

4. Flexibility in Post-Processing: Oversampling provides flexibility in digital signal processing (DSP) techniques applied to the sampled data. The higher sample rate allows for more sophisticated digital filters, interpolation, decimation, and other DSP algorithms to be applied to the oversampled data, enabling enhanced signal processing capabilities.

However, it's important to note that oversampling also has some drawbacks, such as increased digital power consumption. Processing a higher sample rate requires more computational resources, which can result in increased power consumption in digital circuits. Additionally, oversampling may not be suitable for all applications, particularly when the signal of interest has a very limited bandwidth or when there are stringent power constraints.

Applying over sampling ratio:

To digitally increase the sampling rate of a discrete-time signal y(n), you can use a process called interpolation. In this case, the goal is to increase the sampling rate by a factor of L, where L is an integer greater than 1.

The interpolation process involves inserting L-1 zero-valued samples between adjacent samples of y(n). For example, if the original signal y(n) is sampled at a rate of fs, the interpolated signal will have a new sampling rate of L * fs.

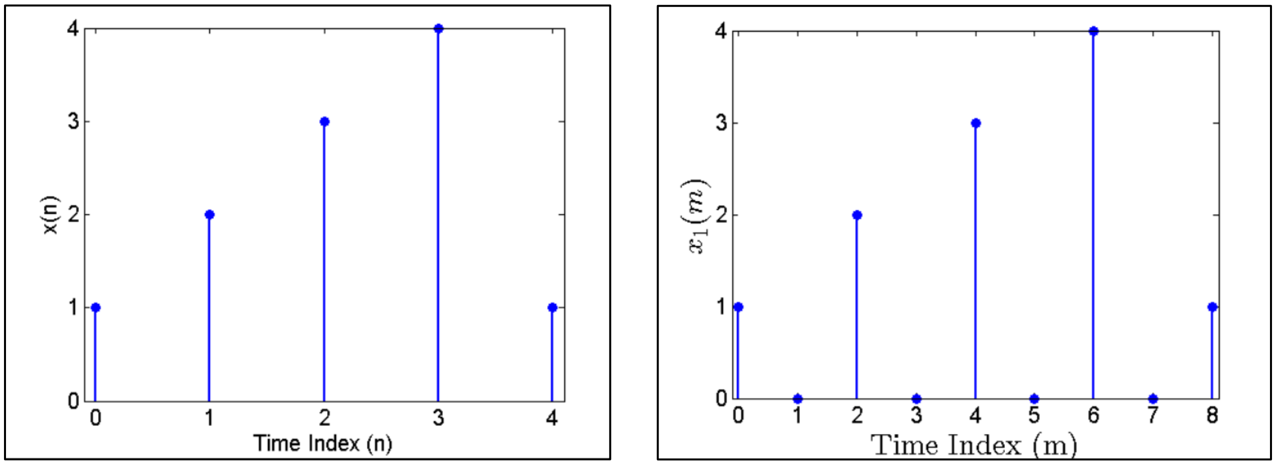


Figure 3.5 Interpolation Process

3.3.2 FIR Filter Implementation

In oversampling, when we increase the sampling rate of a signal, the spectral replicas (images) are shifted to higher frequencies. These replicas are caused by the periodic nature of the discrete-time signal resulting from sampling. The replicas can overlap with the desired frequency band and cause aliasing if not properly addressed.

To prevent aliasing and remove the spectral replicas, an anti-aliasing filter is typically used after oversampling. This filter attenuates or eliminates the frequency components that are outside the desired frequency range, ensuring that only the desired signal content is preserved during oversampling.

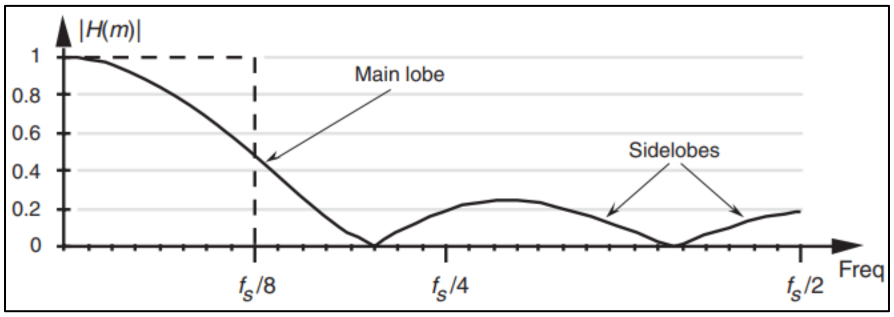


Figure 3.6 Anti-Aliasing Filter Response

The FIR filter is implemented as a series of adders and shifters between the signal and the filter coefficients which is the polyphase method.

$$Y[m] = \sum_{k=0}^{N} b_k x[m - k]$$

Equation 3.2 FIR Filter Equation

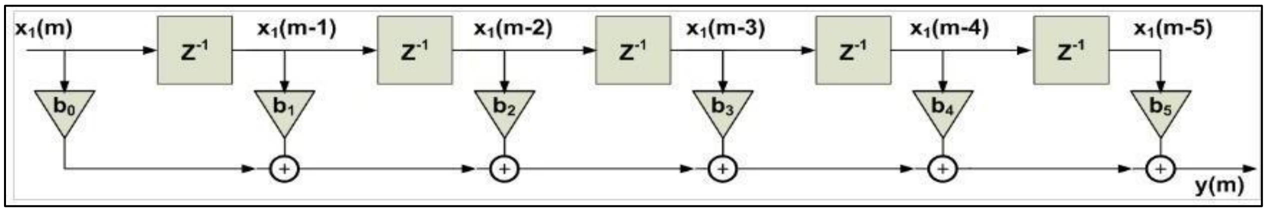


Figure 3.7 FIR Filter Architecture

3.3.2.1 Filter Decomposition

With these assumptions (L=2), let’s examine the straightforward implementation of the interpolation filter. At time index m=5, the FIR filter will be as shown in Figure 3.8:

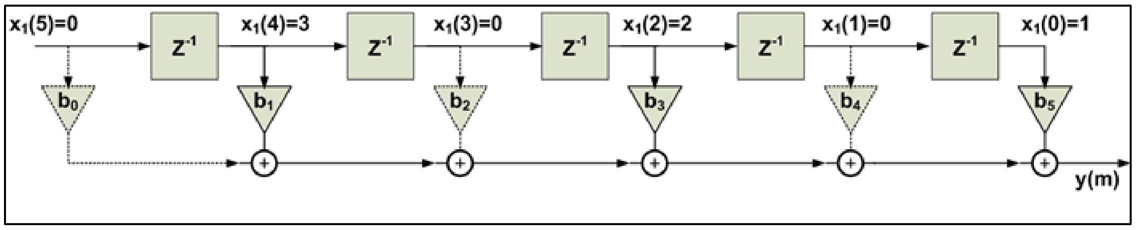


Figure 3.8 The FIR Filter at m=5

At m=5 half of the multiplications of the FIR filter has a zero-valued input. The branches corresponding to these multiplications are shown by the dashed lines. At the next time index, m=6, Figure 3.9 can be obtained:

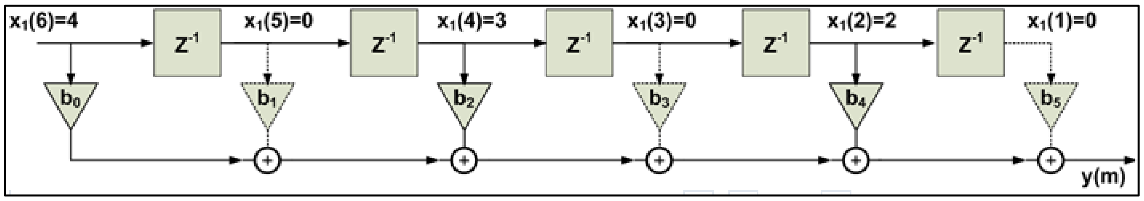


Figure 3.9 The FIR Filter at m=6

Again, those branches which incorporate a zero-valued input are shown by dashed lines. It is observed that for an even time index, the coefficients (i.e., b0, b2, and b4) are important and the sum of the products for the rest of the coefficients becomes zero. This means that at any instant, half of multiplications can be dispensable.

If two different filters are used after the up sampler: one with the odd coefficients and the other one with the even coefficients and add the output of these two filters together to get y(m).

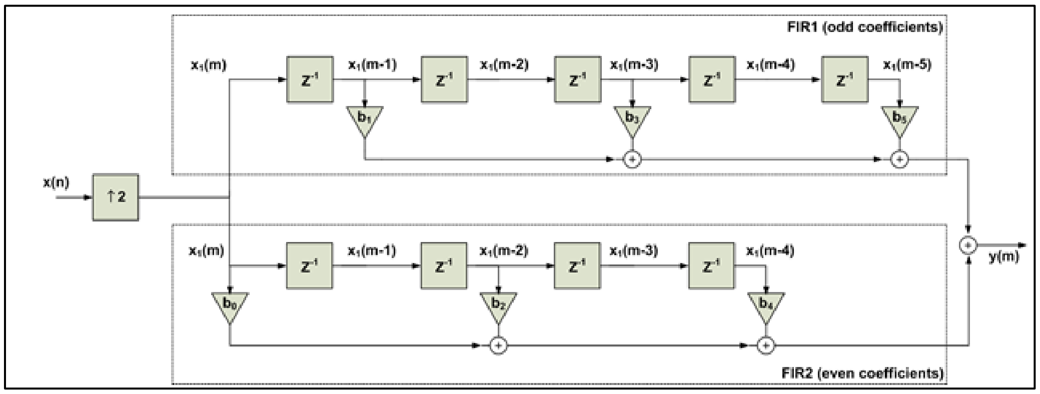


Figure 3.10 Breaking The Filter Difference Equation into Two Sets of Coefficients

The figure 3.10 can be easily obtained by manipulating Equation 3.3 as:

$$y(n) = \big(b_0x(n) + b_2x(n-2) + b_4x(n-4)\big) + \big(b_1x(n-1) + b_3x(n-3) + b_5x(n-5)\big)$$

Equation 3.3 Filter Equation with Two Sets of Coefficients Even and Odd

In figure 3.10, there are two paths shown. The purpose of this decomposition is to simplify the processing based on the properties of the filters involved. At each time index, only one of the filters produces a non-zero output while the other outputs zero. By taking advantage of this property, the processing can be simplified. In figure 3.11, a switch is introduced after the filter FIR2 to be equivalent to the other one on figure 3.10. To match the previous behavior in figure 3.10 in the equivalent circuit of Figure 3.11, the output needs to be forced to zero for odd values of m. Interestingly, the operation of this switch is equivalent to that of an up sampler by a factor of two. An up sampler increases the sampling rate of a signal by inserting zero-valued samples between the original samples. In this case, the switch acts as an up sampler by inserting zero values for odd values of m, effectively matching the behavior of FIR2.

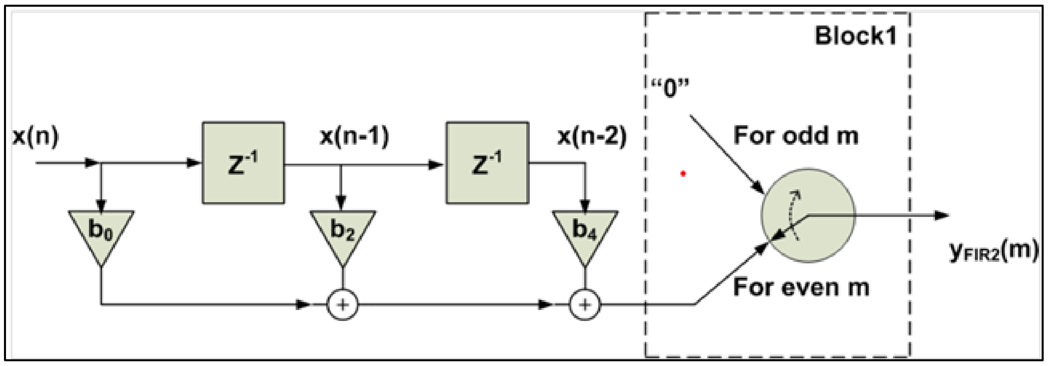


Figure 3.11 The Schematic Is Equivalent to The Cascade of The Up Sampler and FIR2 in Figure 3.10.

3.3.2.2 Noble Identity

When implementing a system that involves both filtering and upsampling, there is a choice in the order of these operations. One approach is to perform upsampling first and then apply the filter, while the other approach is to filter the signal first and then upsample. Both implementations can be equivalent in terms of the final output, but there are some considerations that might make one approach preferable over the other.

Performing upsampling after filtering has the advantage of reducing the overall computational load and power consumption. In this approach, the filter operates at the lower sample rate, which means fewer computations are required compared to upsampling first. This results in reduced complexity and power consumption in the filtering stage.

Additionally, when using an N-tap FIR filter for interpolation, there is no need for "multiply by zero" computations. This is because, in upsampling, the additional zero-valued samples inserted between the original samples do not contribute to the filter output. Therefore, these computations can be avoided, saving computational resources and further reducing power consumption.

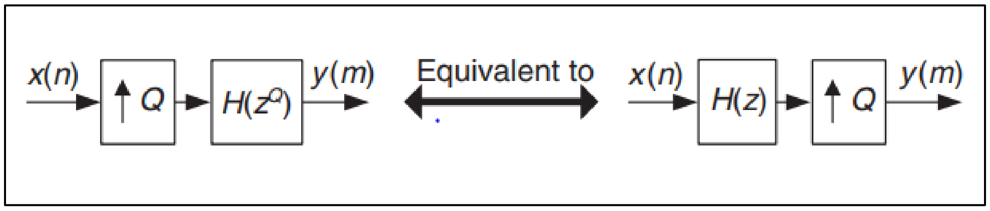


Figure 3.12 Noble Identity

Figure 3.13 shows example for noble identity.

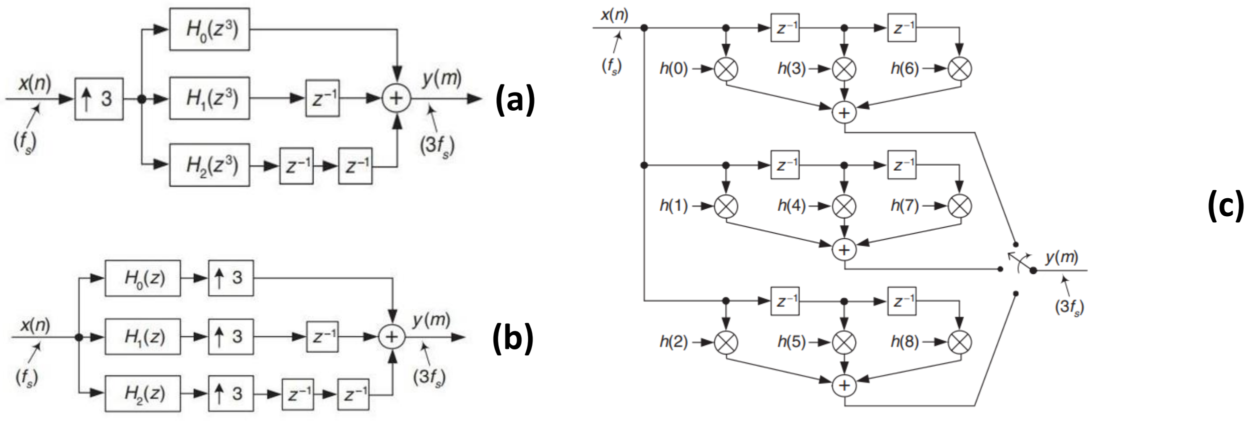


Figure 3.13 Noble Identity Example

For figure 3.13, Three Polyphase decomposition of h(k), for interpolation by L = 3: (a) simple depiction; (b) reduced-length subfilters; (c) final structure.

3.3.2.3 Two Stages Filter

In general, in communication systems, a high over sampling ratio is needed. When a desired interpolation factor *L* is large, say *L* > 20, significant interpolation filter computational savings may be had by implementing the interpolation in Figure 3.14 in two stages as shown in Figure 3.15.

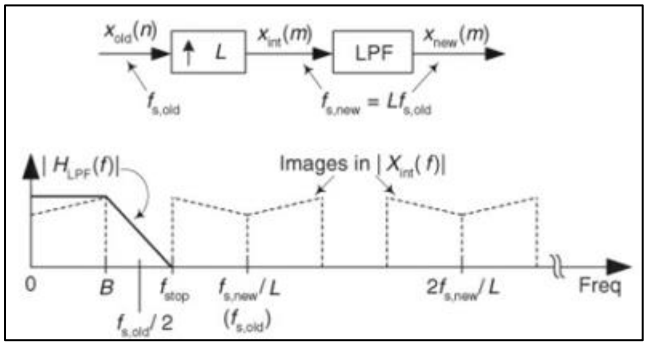


Figure 3.14 One Stage Interpolation

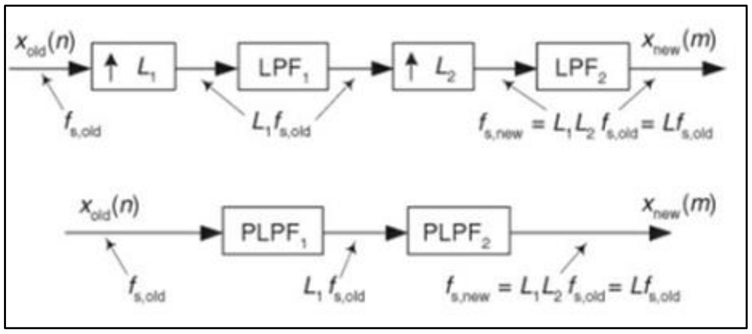


Figure 3.15 Two Stages Interpolation

To achieve overall interpolation by a factor of L using a two-stage interpolation approach, it is possible to reduce the computational workload compared to using a single filter implemented before the interpolation. The optimal interpolation factors L1 and L2 for the two-stage interpolation can be determined using the following equation:

$$L_{2,\text{opt}} \approx 2L \cdot \frac{1-\sqrt{LF/(2-F)}}{2-F(L+1)},$$

Equation 3.4 Optimal Interpolation Factors Equation

Where F is the ratio of the LPF filter's transition region width over the filter's stopband frequency, as such, the filter LPF's transition region extends from B Hz to f_{stop}. B is the 3-dB cutoff frequency, fstop is the beginning of the lowpass filter's stopband frequency

$$F = \frac{f_{\text{stop}} - B}{f_{\text{stop}}} = \frac{f_{\text{s,old}} - 2B}{f_{\text{stop}}}.$$

Equation 3.5 LPF Filter Transition Region to Stopband Frequency Ratio

Upon using equation 3.4 to compute $L_{2,opt}$ and setting L_2 equal to the integer which is sub-multiple of L that is closest to $L_{2,opt}$ the first interpolation factor L_1 is found using:

$$L_1 = \frac{L}{L_2}.$$

Equation 3.6 First Stage Interpolation Factor

A simple example is that when OSR=32 and using equation 3.3 we need 100 taps. but when splitting the filter into two stages 4 and 8 we only need 32 taps.

3.3.2.4 Filter Coefficients

The filter is needed to be optimized for multi-rate applications so a comb-interpolation filter was used with the following transfer function shown in equation 3.7. Where L is the interpolation factor and K is filter order.

Using a comb-interpolation filter to get the coefficients because its coefficients are integers which is easier to be applied using shifters.

Then by using this equation we can get the filter coefficients that will be later implemented.

$$H(z) = \left(\frac{1 - z^{-L}}{1 - z^{-1}}\right)^k$$

Equation 3.7 Comb-Interpolation Filter Transfer Function

3.3.3 Filter Results

Applying the previous theoretical background, the final implementation of the used filter is as following:

By using third order filter with OSR=32 splitting it into two stages and using equation 3.4 to calculate the optimum values for OSR of first and second stage. It is found that $L_1 = 4$ and $L_2 = 8$. Also, Noble identity is applied then the final structure of the filter will be as shown in figures 3.16, 3.17, 3.18 and 3.19

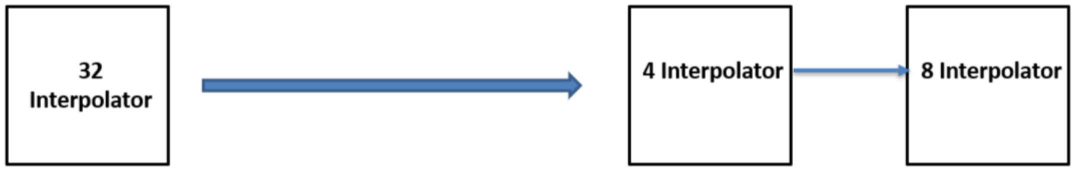


Figure 3.16 Two Stages Filter

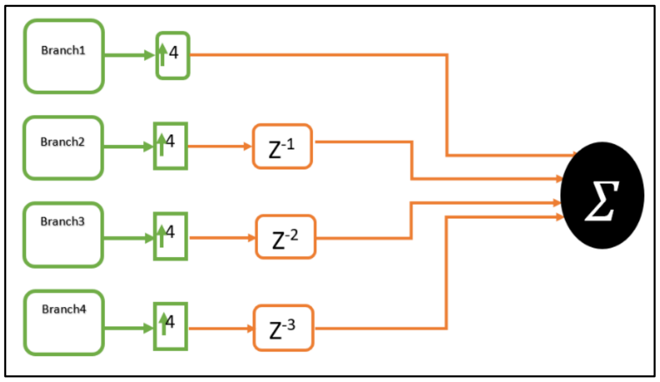


Figure 3.17 First Stage Implementation

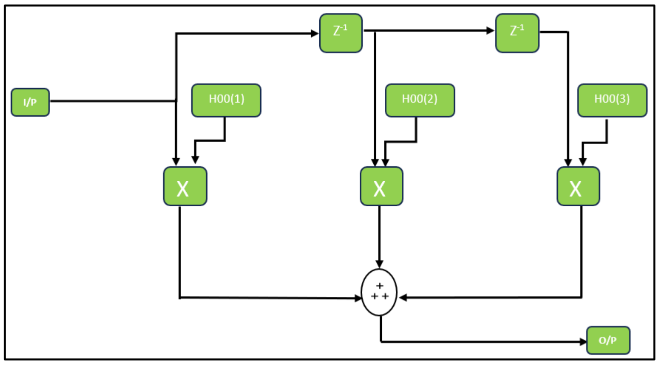


Figure 3.18 H0 Implementation

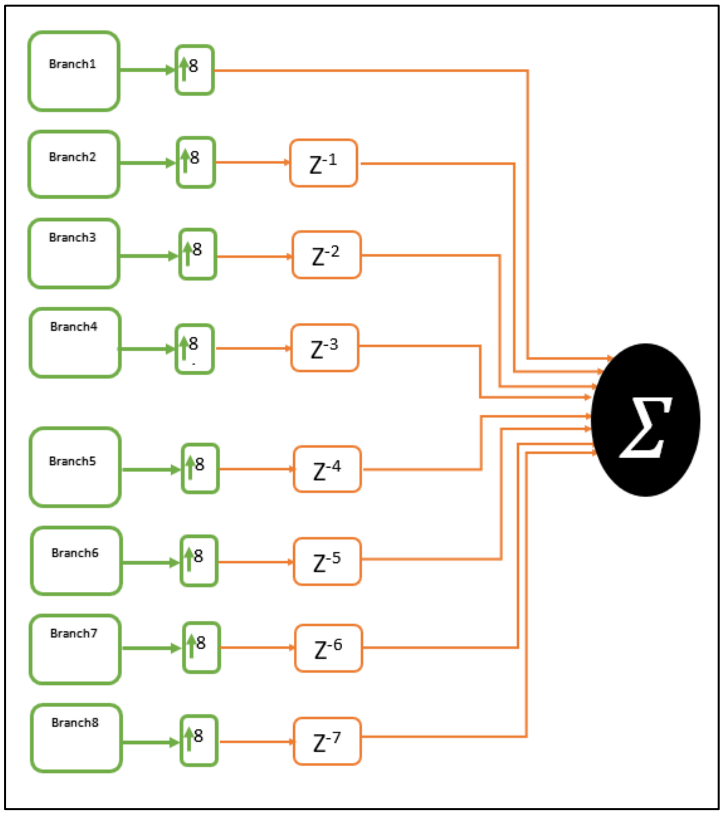


Figure 3.19 Second Stage Implementation

The figure 3.20 shows the filter response of 32-UP sampler **after decomposition** at BW= and first null =fs/32=0.03125

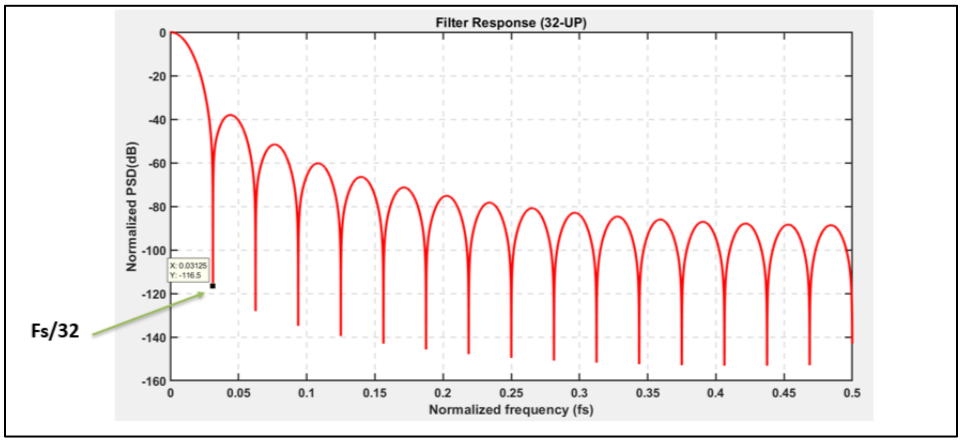


Figure 3.20 32-Upsampler Filter Response

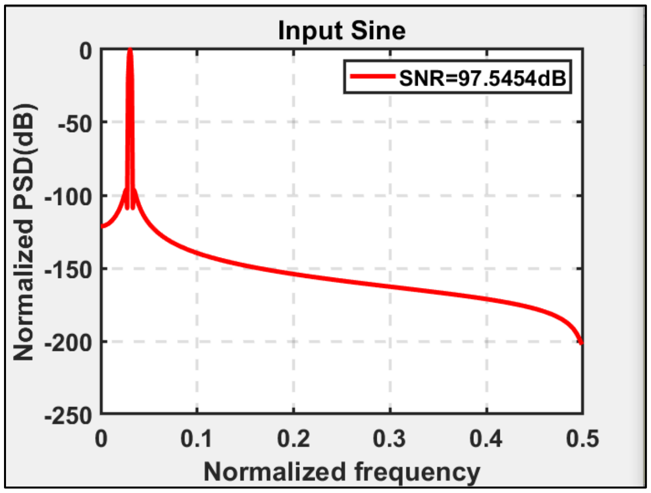


Figure 3.22 Filter Input Spectrum

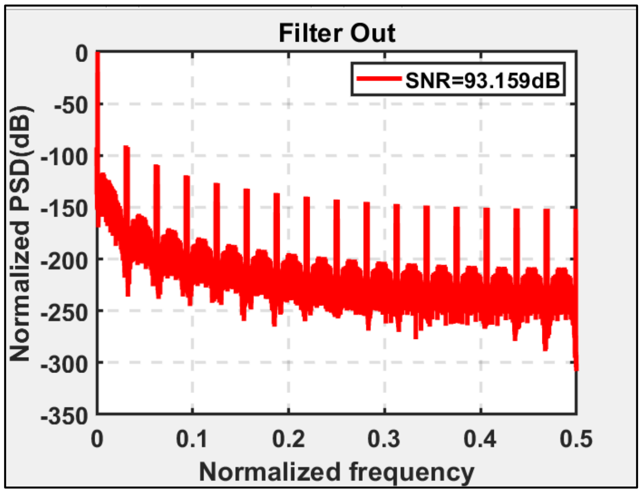


Figure 3.21 Filter Output Spectrum

This output is resulted from an input sine wave and as shown the images are suppressed with an input **SNR = 97.54 dB** and output **SNR = 93.159 dB.**

3.4 Digital Sigma Delta

In the transmitter architecture the output of the interpolation filter is taken as an input to sigma delta block. Sigma delta is an essential block in this transmitter system in which it makes modulations to the signal to convert it from high bit count to low bit count. It has the ability to decrease the complexity of analog circuits dramatically especially when a single bit structure is used. Sigma delta is used in this system for two main functions. First as it is mentioned above the interface of the transmitter would be a one-bit DAC. So, it is a must to modulate the transmitter output signal into a one-bit signal and that is delta sigma mission. Second function of sigma delta is that has the ability to make noise shaping to the quantization noise of the signal to make most of the noise outside the signal bandwidth to have a cleaner signal after passing by a low pass filter in RF domain later.[7][8]

3.4.1 Over Sampling and Noise Shaping

Over sampling is a very important concept related to sigma delta systems and noise shaping. Sampling the signal with sampling frequency much higher than Nyquist frequency leads to relaxing the conditions on the low pass filter in the analog domain leading to simpler design as there is no need for the high sharpness in the filter response as shown in figure 3.23.

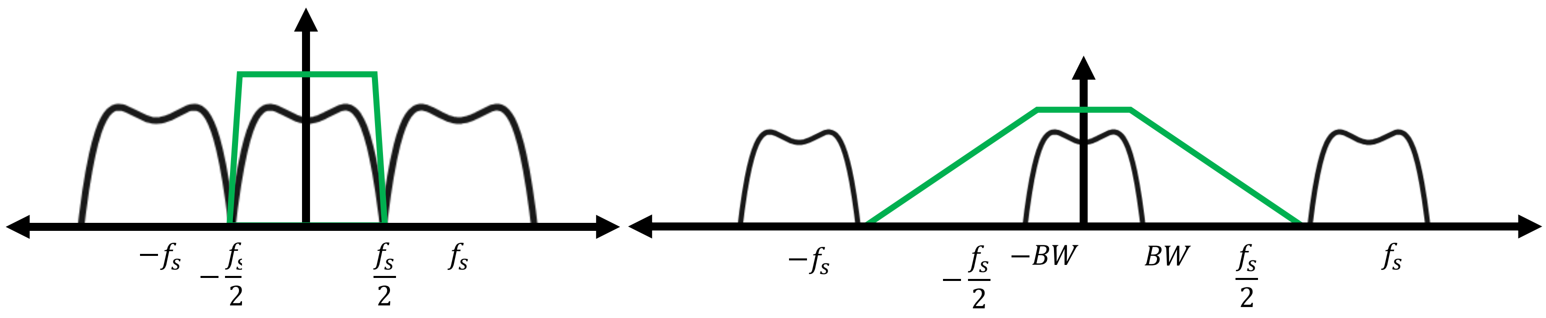


Figure 3.23 Filter Relaxation After Increasing Sampling Frequency

The OSR is taken as the ratio between the sampling frequency and Nyquist frequency.

$$OSR = \frac{f_s}{f_n} = \frac{f_s}{2BW}$$

Equation 3.8 OSR Equation

In addition, using high OSR (i.e., increasing sampling frequency) increases the resolution of the signal as the quantization noise decreases so that SNR increases. Quantization noise can be considered as white noise if there is no correlation between input signal and sampling frequency. Quantization noise power can be computed as shown in equation 3.9.

$$P_q = \frac{\Delta^2}{12}$$

Equation 3.9 Quantization Noise Power Equation

Where P_q refers to the quantization noise and Δ refers to the step size.

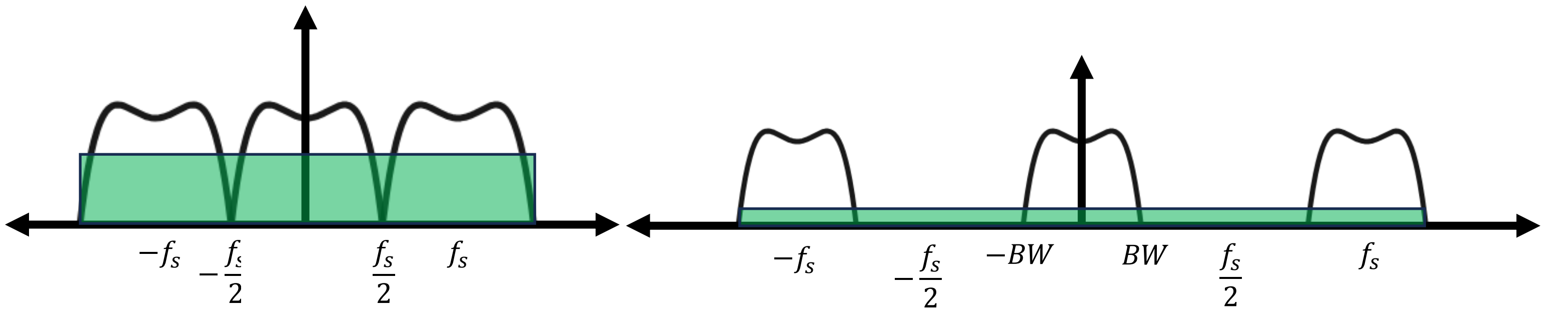


Figure 3.24 Quantization Noise Before and After Over Sampling

In signal bandwidth noise can be calculated as shown in equation 3.10 in case of sampling at Nyquist frequency. (i.e., OSR=1)

$$IBN = \frac{\Delta^2}{12}$$

Equation 3.10 Noise Power for OSR=1

Where IBN refers to in-band noise

In case of oversampling (i.e., OSR > 1) the noise in signal bandwidth is decreasing by OSR factor as the noise becomes distributed on a higher bandwidth as shown in figure 3.24. So, the in-band noise can be computed as shown in equation 3.11.

$$IBN = \frac{\Delta^2}{12 * OSR}$$

Equation 3.11 Noise Power for Oversampled Signal

The oversampling concept is highly related to the noise shaping concept. As the noise shaping mission now is to reshape the noise so that most of the in-band noise is shifted outside the signal bandwidth which is the band of interest in the systems. The noise shaping graphically can be shown as the plot in figure 3.25.

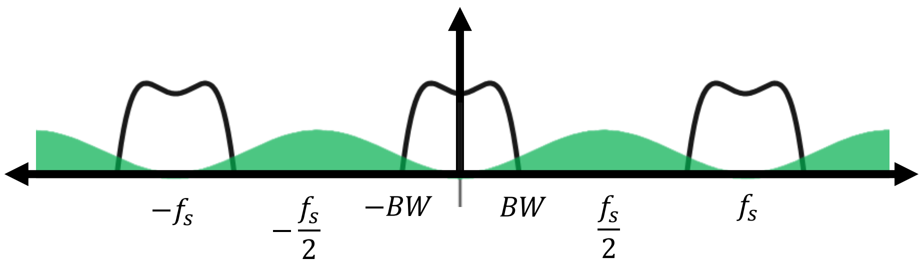


Figure 3.25 Noise Shaping phenomenon

So, what is needed here is to make the quantization noise in the system to pass by a differentiator to take the shape of high pass filter response while the signal should have no changes in its frequency domain and that what the sigma delta would do.

Having the feedback loop in figure 3.26 into consideration. It is required is to make the quantization noise which is represented by signal e(n) have a noise transfer function (NTF) which represents a differentiator. So, NTF is required to be as shown in equation 3.12.

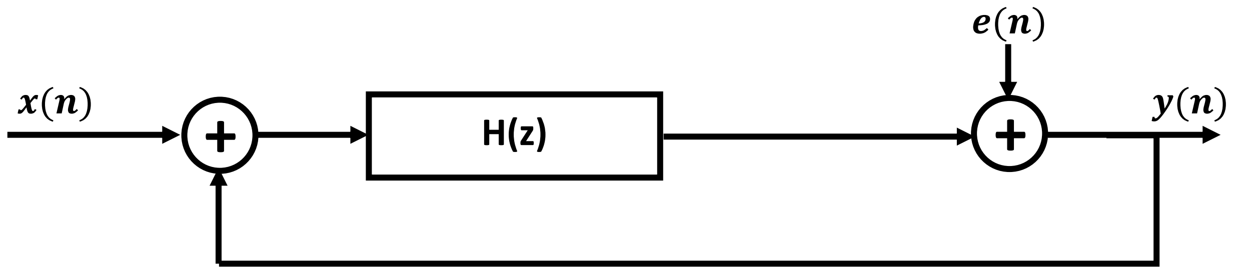


Figure 3.26 Sigma Delta Feedback Loop

$$NTF = (1 - z^{-1})^{L}$$

Equation 3.12 Noise Transfer Function equation

Where L is the order of sigma delta

For the signal transfer function (STF) and NTF

$$STF = \frac{y}{x} = \frac{H(z)}{1 + H(z)}$$

$$NTF = \frac{1}{1 + H(z)}$$

If H(z) is taken as follows

$$H(z) = \frac{z^{-1}}{1 - z^{-1}}$$

STF and NTF can be computed as shown in equation 3.13.

$$|STF| \approx z^{-1}$$

$$|NTF| \approx 1 - z^{-1}$$

Equation 3.13 Sigma Delta loop Transfer Functions

The two equations for the transfer function fulfil the requirement. As for STF, it is corresponding to a delay so the signal just gets delayed but no change in frequency domain. For NTF which is a differentiator transfer function. So, the quantization noise gets the high pass filter shape and can be eliminated from the signal by using a low pass filter in RF domain.

As it was mentioned, NTF is a differentiator that has an order of the sigma delta L. So, increasing sigma delta order leads to higher noise shaping effect as shown in figure 3.27.

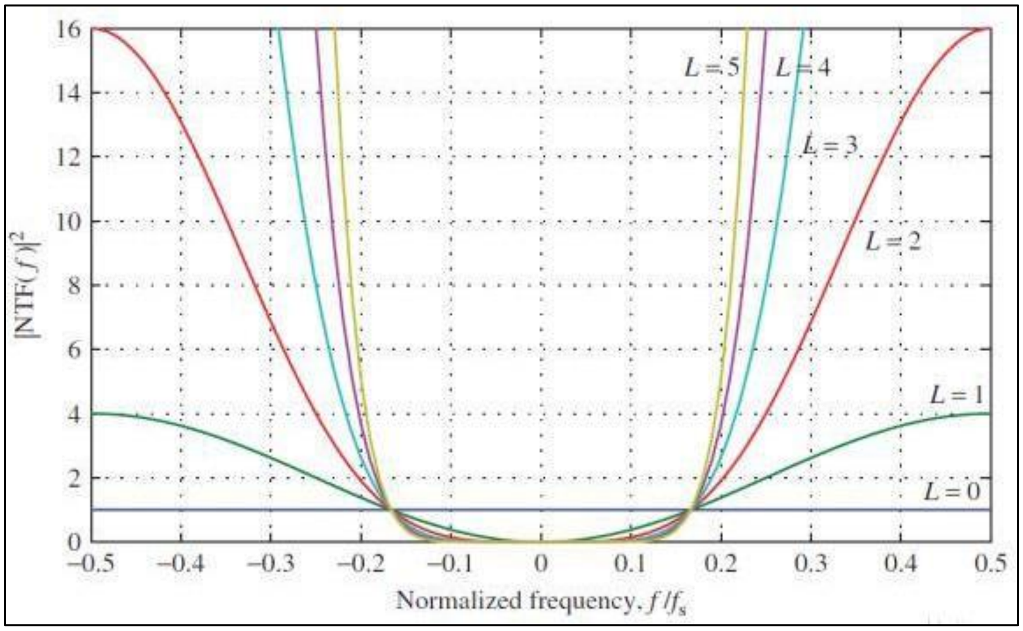


Figure 3.27 Noise Shaping Effect for Different Sigma Delta Orders

But increasing sigma delta order more than two in most cases complicates the loop stability.

3.4.2 Sigma Delta Architectures

In this work six different architectures of sigma delta were designed. In this section the six architectures would be taken into consideration comparing their functionality and output SNR to decide the one with best outcome to be used in the digital beamforming transmitter system. As the transmitter system output would be interfaced with a one-bit DAC later, one-bit output sigma delta is needed to achieve the output width constrain. The output is one or zero depending on the output of the quantizer in the block.

Three types of sigma delta were taken into consideration during this work:

- Cascaded Integrators with Distributed Feedback Sigma Delta (CIFB)
- Error Feedback Sigma Delta (EFB)
- Multi-stage Noise Shaping Sigma Delta (MASH)

The three types work is explained in more details in the next sections.

3.4.2.1 Cascaded Integrators with Distributed Feedback Sigma Delta

Three architectures are implemented for CIFB. First, second and third order are built to test their functionality and SNR. CIFB is consists of a cascade of L integrators where L refers to the order of sigma delta with output feedback signal weighted by a_i. as well as the input signal is being fed to each integrator input terminal with different weight factors b_i as shown in figure 3.28. Where U is the input for the CIFB block and V is the output of the block.

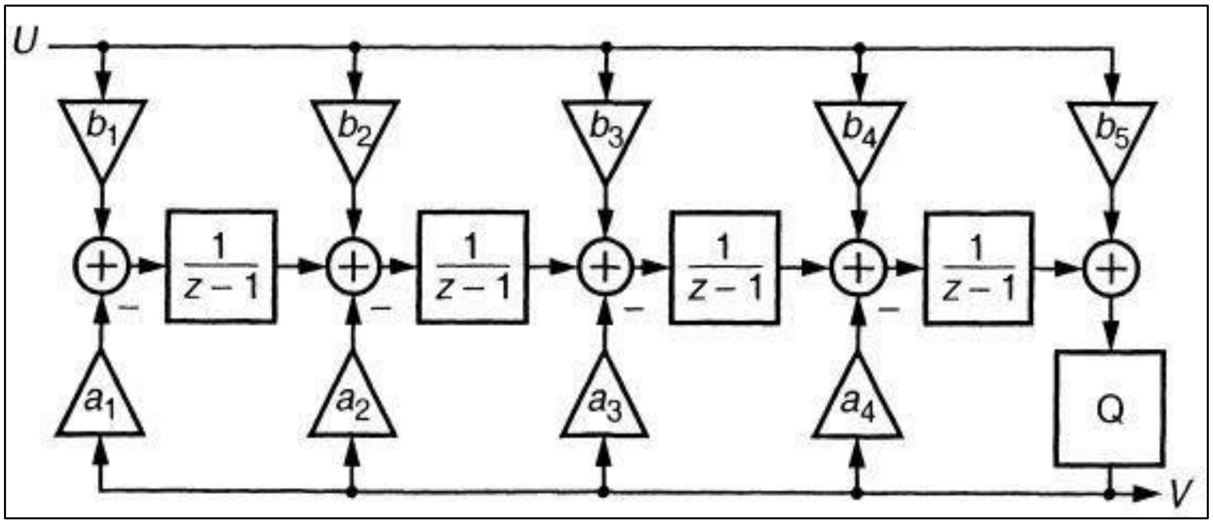


Figure 3.28 CIFB Sigma Delta Architecture [7]

3.4.2.1.1 CIFB First Order

First order CIFB sigma delta is implemented in simulink as shown in figure 3.29. Using Richard Schreier's MATLAB toolbox the output spectrum, STF, NTF and poles and zeros plot were plotted for the implementation feedback loop.[7]

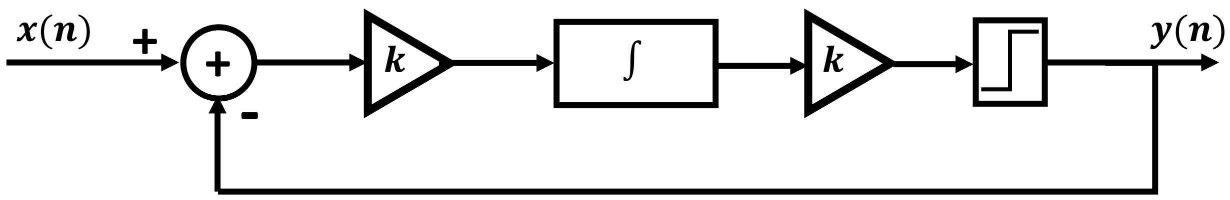


Figure 3.29 First Order CIFB Sigma Delta

The architecture consists of a feedback loop with one integrator and comparator to get the output bit for the sigma delta (1 or 0).

The output spectrum is as shown in figure 3.30 for OSR of 32 and input sinusoidal of amplitude of -4dB.

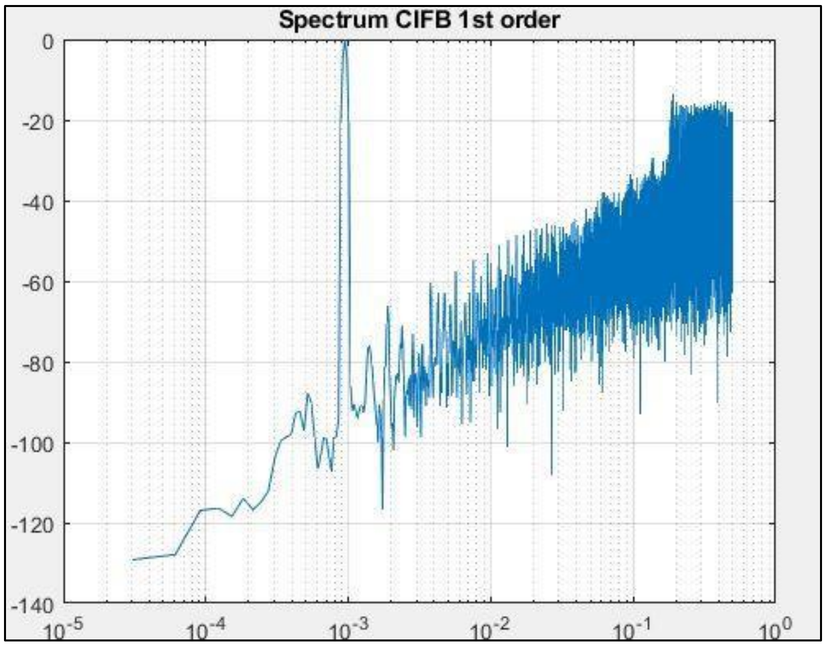


Figure 3.30 First Order CIFB Sigma Delta Output Spectrum

The output spectrum plot in figure 3.30 shows the noise shaping applied by the first order CIFB sigma delta in which the noise takes the high pass filter shape and the in-band noise decreases. Finally, first order CIFB sigma delta achieved output signal SNR about 40.84dB.

3.4.2.1.2 CIFB Second Order

CIFB second order sigma delta architecture is as shown in figure 3.31. The architecture was implemented on Simulink and tested to get output spectrum and SNR.

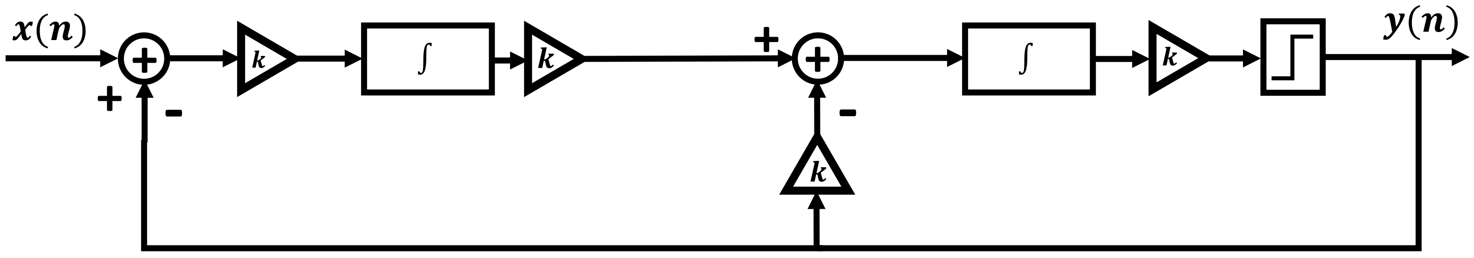


Figure 3.31 CIFB Second Order Sigma Delta Architecture

The design has two loops of two integrators cascaded together with output feedbacking with weights a_1 and a_2. The input signal is a sinusoidal wave with OSR of 32 and amplitude of -4dB with output y which is a one-bit signal.

The output spectrum is as shown in figure 3.32.

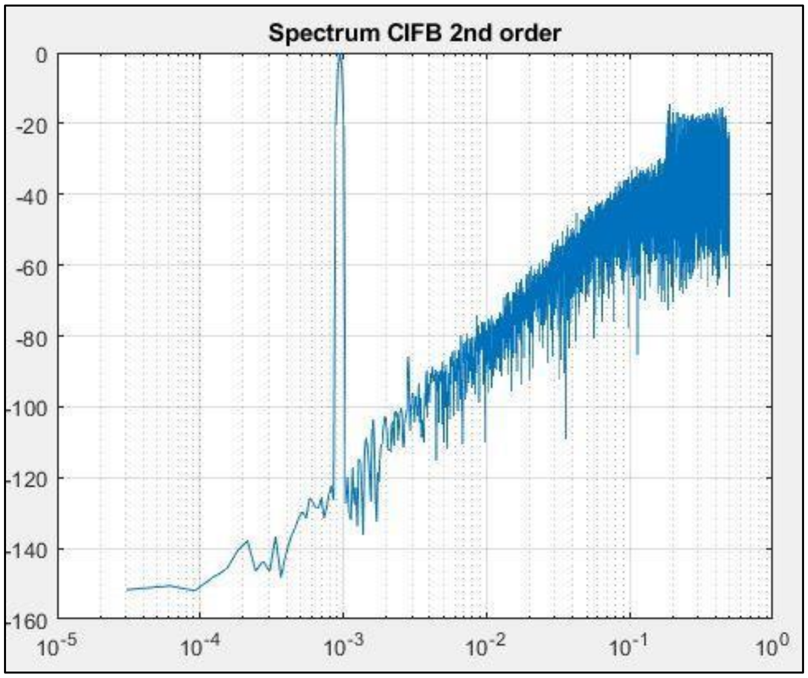


Figure 3.32 CIFB Sigma Delta Output Spectrum

The output signal SNR for the second order CIFB is about 55 dB for input signal of OSR 32 and amplitude of -4dB. Same as first order the output spectrum in figure 3.32 shows the noise shaping done to the signal.

3.4.2.1.3 CIFB Third Order

The third design for CIFB sigma delta is a third order CIFB. The architecture is shown in figure 3.33.

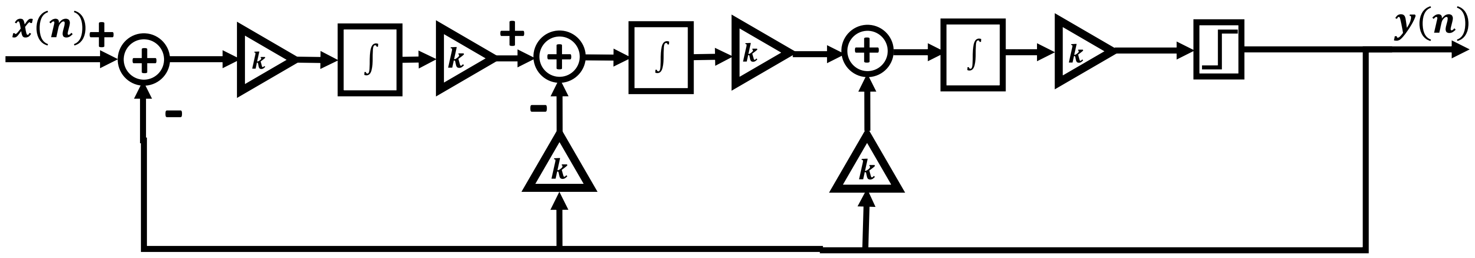


Figure 3.33 Third Order CIFB Sigma Delta Architecture

The third order architecture consists of the three cascaded loop of the integrators with the weighted output feedbacks. The input same as the first and second order case a sinusoidal of amplitude -4dB and OSR=32. Where the quantizer output is a one-bit signal.

The output spectrum is shown in figure 3.34.

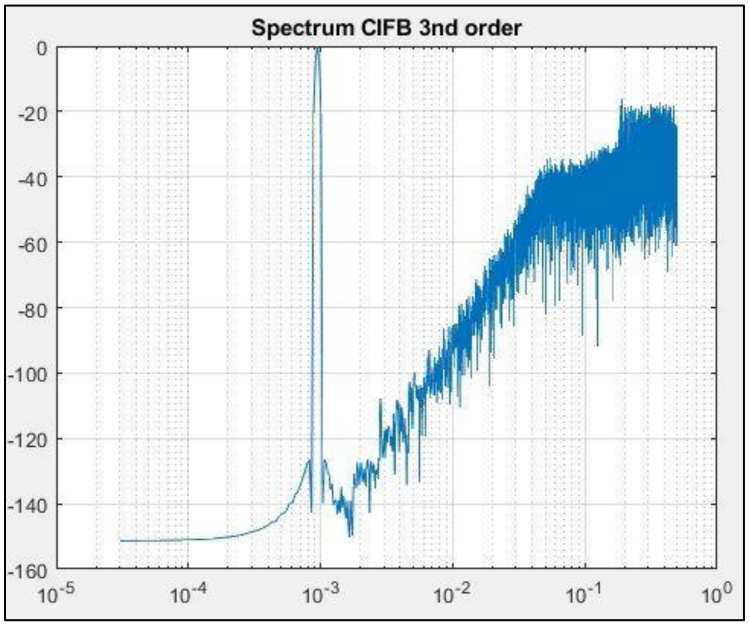


Figure 3.34 CIFB Third Order Output Spectrum

The noise shaping is done to the input signal with output SNR of 61.3dB.

As mentioned above sigma delta architecture with order greater than two have stability complexity so, CIFB third order poles were plotted to test the stability as shown in figure 3.35.

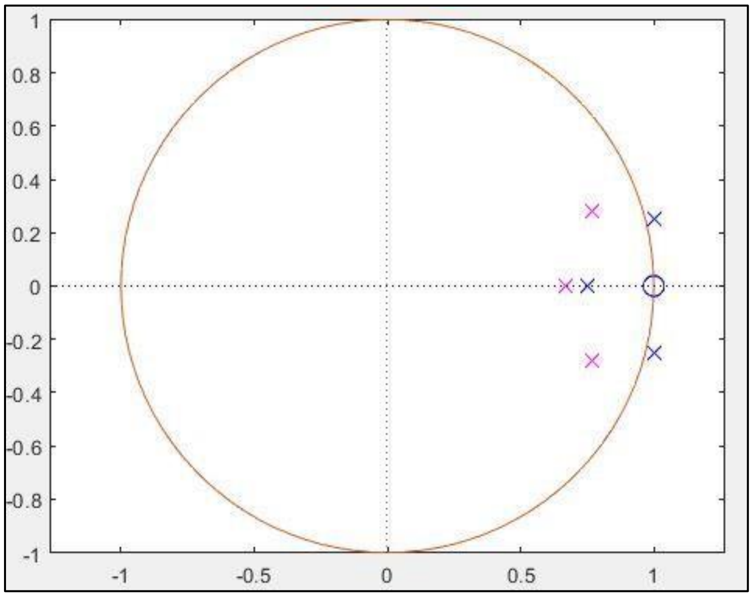


Figure 3.35 CIFB Third Order Poles and Zeros

The poles of the third order CIFB were plotted. And as shown in figure 3.35 the poles are outside the unity circle (i.e., more than one). So, the system is unstable and can’t be used in the transmitter system.

3.4.2.2 Error Feedback Sigma Delta

Second type of sigma delta tested for the transmitter system is the error feedback sigma delta. First and second order sigma delta architectures are implemented and tested functionally and for the output SNR result. The architecture of EFB is a feedback loop with the quantization error is fedback to the input by taking the difference between the signal before quantizer and after quantizer and fed it back to the input. Figure 3.36 show a first order EFB with input U and output V. EFB sigma delta design is simpler than CIFB as it doesn’t have integrators.

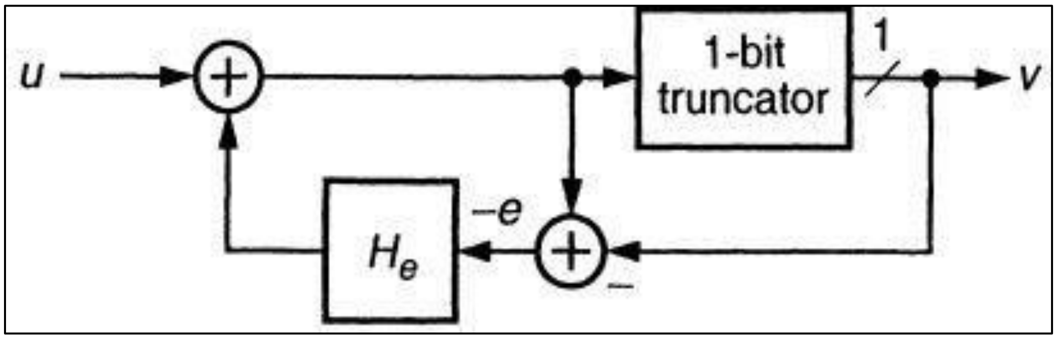


Figure 3.36 EFB Sigma Delta [7]

3.4.2.2.1 EFB First Order

The first design for the EFB sigma delta is a first order EFB as shown in figure 3.37. The architecture was made on Simulink system and tested to get its output spectrum and SNR.

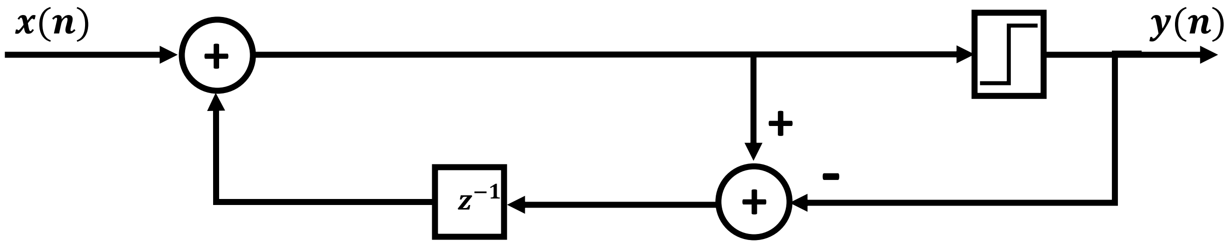


Figure 3.37 First Order EFB Implementation

The design consists of a feedback loop with the quantization error is fed back to the input with a delay element in its chain. The input is kept the same as CIFB case a sinusoidal wave with amplitude -4dB and OSR 32. The output signal for the architecture is a one-bit signal.

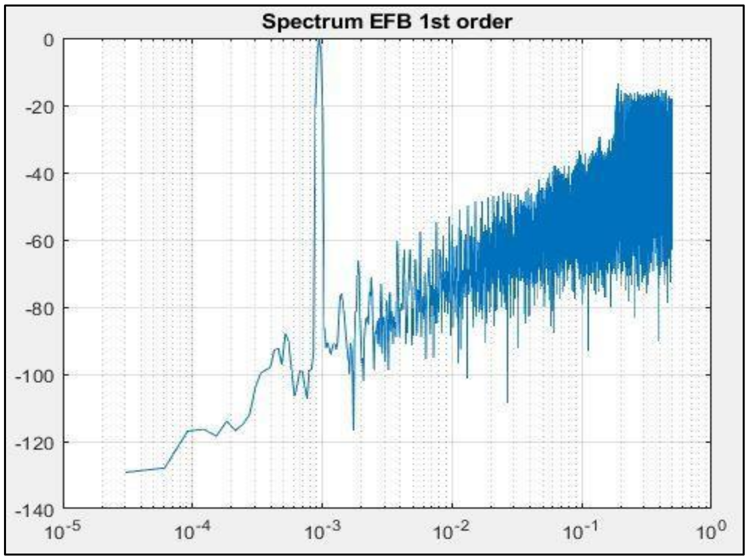


Figure 3.38 EFB First Order Output Spectrum

Figure 3.38 shows the output spectrum with the noise shaping done to the signal. The output SNR for the first order EFB sigma delta is 40.84dB.

3.4.2.2.2 EFB Second Order

The second design for EFB and the fifth for the sigma delta in this work is a second order error feedback sigma delta. The architecture implemented on Simulink is shown in figure 3.39.

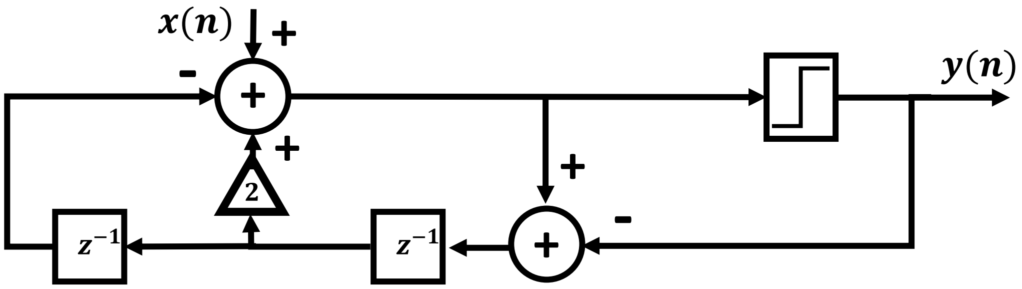


Figure 3.39 Second Order EFB Implementation

The design of EFB second order sigma delta consists of the quantization error feedback loop where the quantization error is fed back to the input twice one with gain 2 and the other delayed by a delay element. The input is same as above a sinusoidal wave with amplitude -4 dB and OSR 32. The signal is quantized to produce the output y which is a one-bit signal.

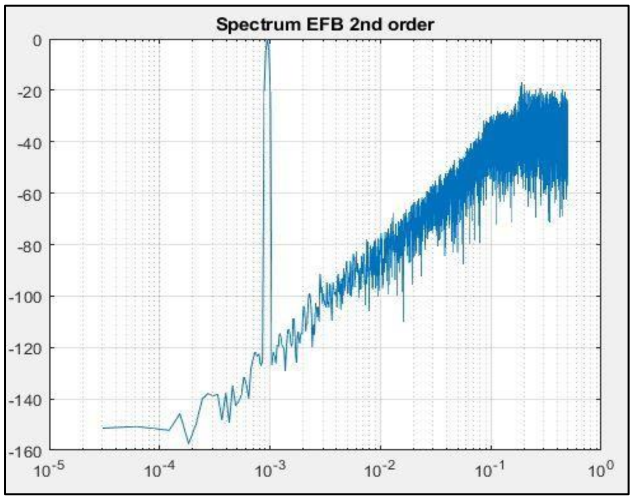


Figure 3.40 Second Order EFB Sigma Delta Output Spectrum

The output spectrum of EFB second order is as shown in figure 3.40. The output figure shows the noise shaping done to the signal producing an output snr of 56.5dB.

3.4.2.3 Multi-stage Noise Shaping Sigma Delta (MASH)

MASH is the third type of sigma delta implemented for this work. Multiple stages of noise shaping are employed to improve the overall performance of the converter. Each stage in the MASH architecture contributes to the noise shaping process and helps to enhance the resolution. Each stage of MASH could be another type of sigma delta like EFB discussed above. MASH sigma delta is a stable architecture for orders more than two which is not available easily for the other architectures.

3.4.2.3.1 Third Order MASH Sigma Delta

A third order MASH sigma delta is implemented to be tested functionally. For the MASH design as shown in figure 3.41. it consists of three stages of first order EFB sigma delta where the quantization noise of each stage is taken as an input for the next stage. The output of the architecture is a summation of the output of the three stages.

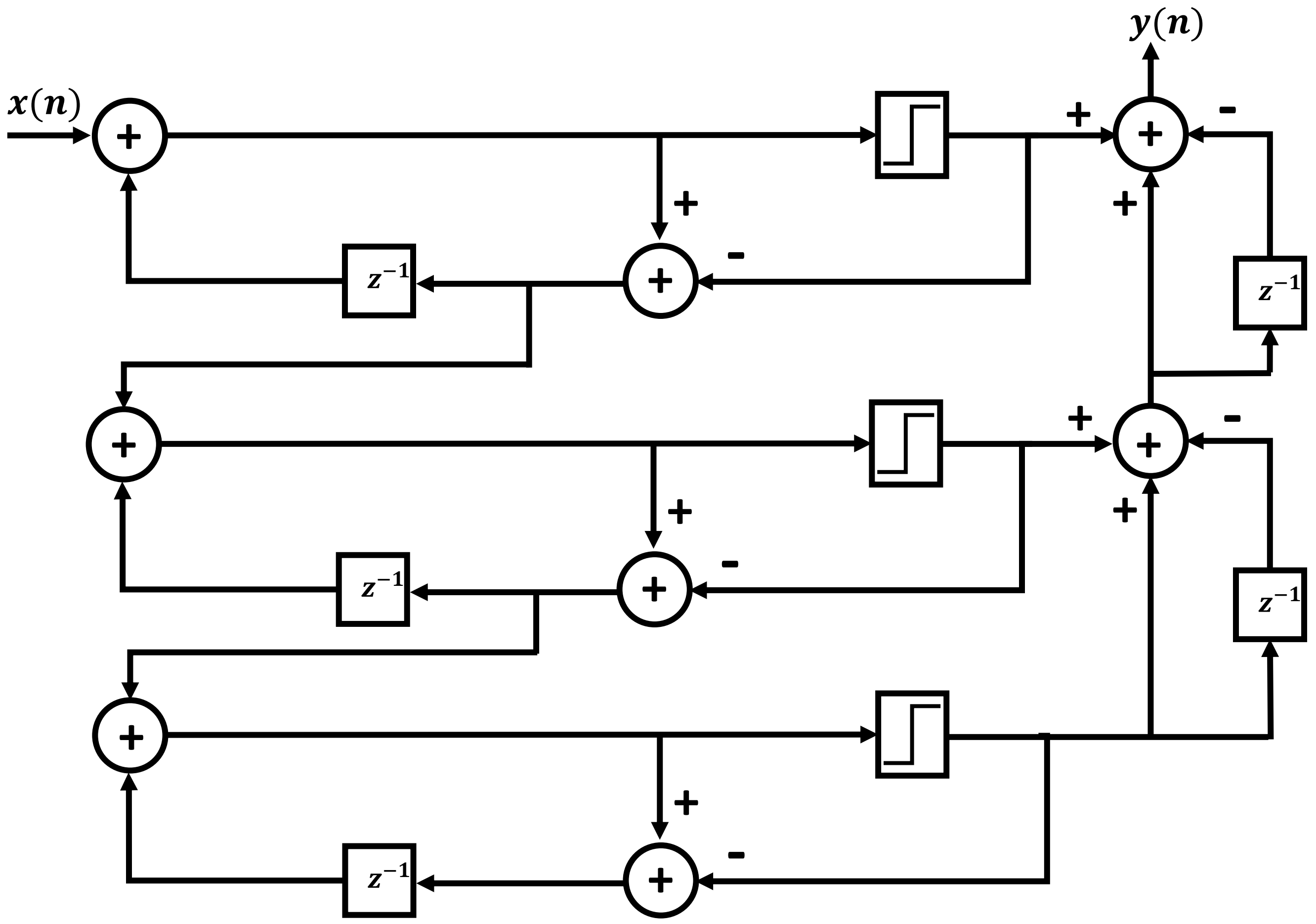


Figure 3.41 Third Order MASH Sigma Delta Architecture

The output spectrum is as shown in figure 3.42. the noise shaping of the output signal is done so that the noise can be removed using a filter to get the signal with high resolution.

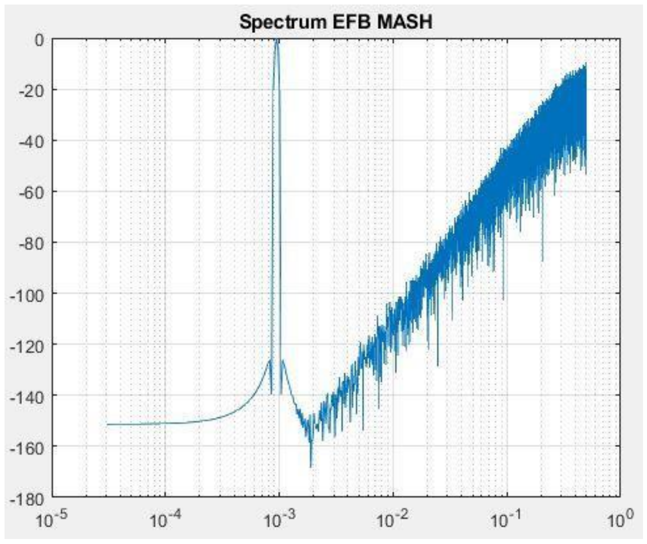


Figure 3.42 Third Order MASH Sigma Delta Output Spectrum

The output SNR for the third order MASH for input amplitude -4db and OSR 32 is 82.25dB. But for this transmitter the MASH implementation can't be used as the output signal produced from summation of three stages output has a width which is more than one bit. As mentioned above the transmitter is built to be interfaced with one-bit DAC so the MASH architecture can't be used here.

3.4.3 Sigma Delta Summary and Conclusion

Summarization for sigma delta part SNRs are shown in table 3.1.

Architecture	SNR (dB)
First Order CIFB	40.8
Second Order CIFB	55
Third Order CIFB	61
First Order EFB	40.8
Second Order EFB	56.5
Third Order MASH	82.2

Table 3.1 Sigma Delta Architectures SNRs

By comparing the output SNRs third order MASH has the highest SNR with 82.2dB followed by third order CIFB with 61dB. But both architectures are excluded from the transmitter architecture as third order MASH output is more than one bit while third order CIFB is unstable architecture. So, by excluding both of them second order EFB has the priority with the high SNR of 56.5dB. EFB output is one bit which is suitable for the one-bit DAC interface. Also, EFB has simpler design than CIFB as it has less blocks and no integrators. So, second order EFB shown in figure 3.43 would be the sigma delta architecture used in the system.

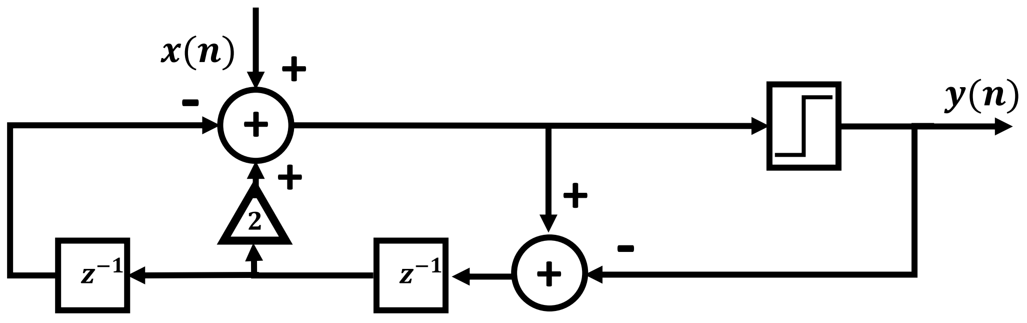


Figure 3.43 Sigma Delta Architecture Used in The System

The output signal of the EFB sigma delta would be taken as input for digital up converter block (DUC) that would be discussed in next section.

3.5 Digital Up Converter

The final block in the chain is digital up converter (DUC) block. The output bitstream of sigma delta is taken as an input to the DUC to be put on the carrier frequency to be transmitted. The DUC block is as shown in figure 3.44. The inputs I and Q are multiplied by sine and cosine waves with the carrier frequency then both signals are summed together. The summation doesn't affect the signals due to the orthogonality of sine and cosine.[2]

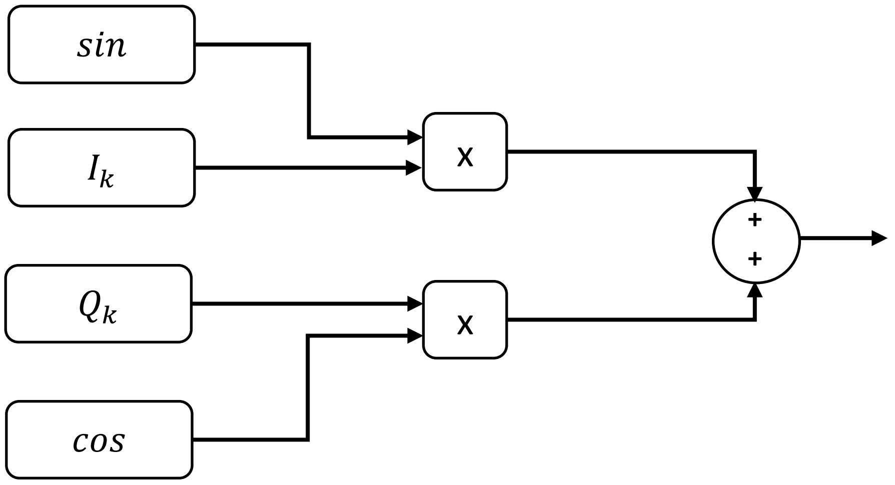


Figure 3.44 DUC Architecture

The carrier sine and cosine signals are sampled with sampling frequency four times the DUC output frequency which greatly simplifies the up-conversion process as the sine and cosine waves can be represented as sequence of 0, -1, 0, 1, …. The output of DUC is the output of the chain that would be interfaced to one-bit DAC.

DUC block is tested by applying sine and cosine waves to the inputs with using carriers of frequency $\frac{f_s}{4}$ and check the output. The input and output plots are shown in figures 3.45 and 3.46.

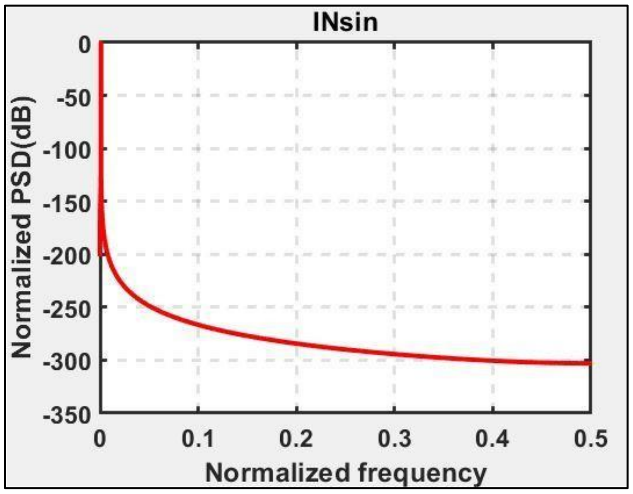


Figure 3.46 DUC Input Spectrum

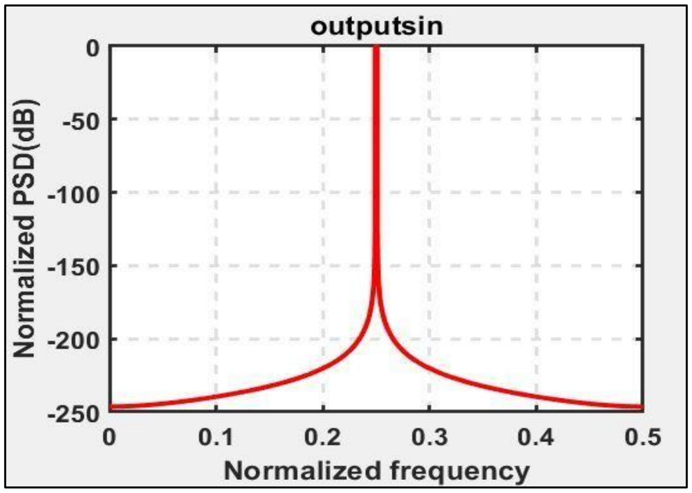


Figure 3.45 DUC Output Spectrum

The figures 3.45 and 3.46 shows the successful process of up-conversion. The signal frequency moved from baseband to the carrier frequency. By implementing the DUC block, all the blocks of the chain are implemented and tested so, the next step is the beamforming transmitter system integration. The integration and the results of single chain and multiple chains system would be discussed in the next section.

3.6 System Integration and Results

Throughout the chapter details were discussed for each block in our system. In this chapter single chain integration and beamforming system integration would be taken in consideration showing their results and functionality for the transmitter system.

3.6.1 Single Chain Integration

CWM digital phase shifter, two interpolation stages 32 OSR polyphase interpolation filter, second order EFB digital sigma delta and digital up converter blocks. Integration of these block is done to build the single transmitter chain and test the functionality of the chain as shown in figure 3.47.

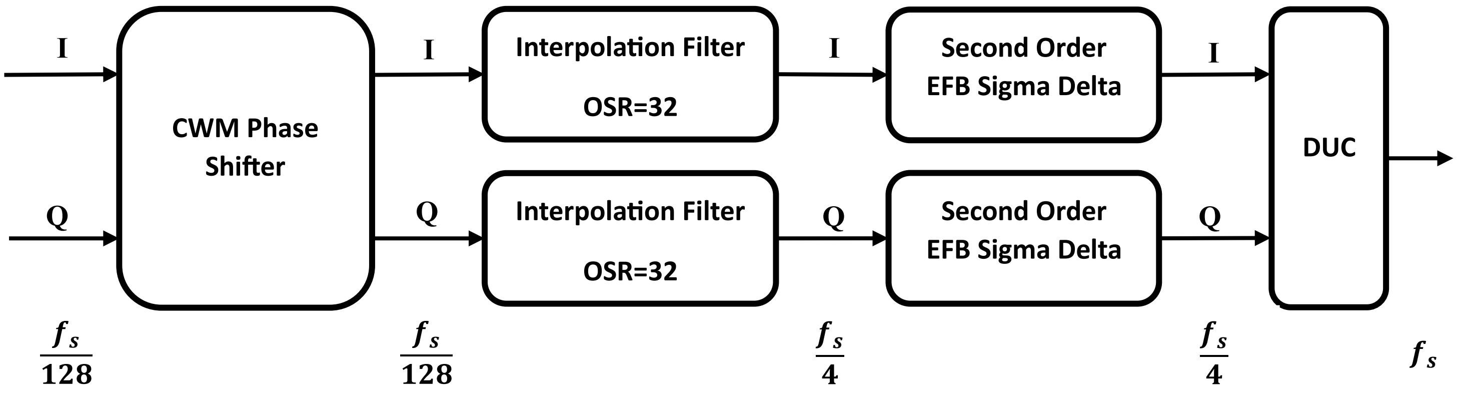

I
Q
CWM Phase Shifter
I
Q
Interpolation Filter OSR=32
Interpolation Filter OSR=32
I
Q
Second Order EFB Sigma Delta
Second Order EFB Sigma Delta
I
Q
DUC
$\frac{f_s}{128}$
$\frac{f_s}{128}$
$\frac{f_s}{4}$
$\frac{f_s}{4}$
f_s

Figure 3.47 Single Chain of The Beamforming Transmitter

Starting with input signals I and Q which are sine and cosine signals come from any communication modulation technique. I and Q are the inputs for the digital phase shifter block having sampling frequency of $\frac{f_s}{128}$. The output of phase shifter is $k\Delta\varphi$ phase shifted version of the input having the same sampling frequency of $\frac{f_s}{128}$. Then it's taken as input for the two stages interpolation filter to get the signal oversampled by OSR 32 so that taking the filter output sampled at sampling frequency $\frac{f_s}{4}$. The sigma delta does the noise shaping and decrease the signal width to one bit. The output signal of sigma delta is sampled at $\frac{f_s}{4}$ and taken as an input for the DUC block so that the signal is carried into the carrier frequency to be transmitted. The DUC block oversamples the signal by 4 to avoid interference between the signal images. The output of DUC is the chain output having sampling frequency at f_s. The sampling frequency for each signal is shown in table 3.2.

Signal	**Sampling Frequency**
Input	$\frac{f_s}{128}$
Phase Shifter Output	$\frac{f_s}{128}$
Interpolation Filter Output	$\frac{f_s}{4}$
Sigma Delta Output	$\frac{f_s}{4}$
DUC Output	f_s

Table 3.2 Sampling Frequency Table

To test the chain functionality inputs I and Q a sine signal and a cosine signal of amplitude -4dB and sampling frequency of $\frac{f_s}{128}$ are applied to the chain. The input spectrum is as shown in figure 3.48.

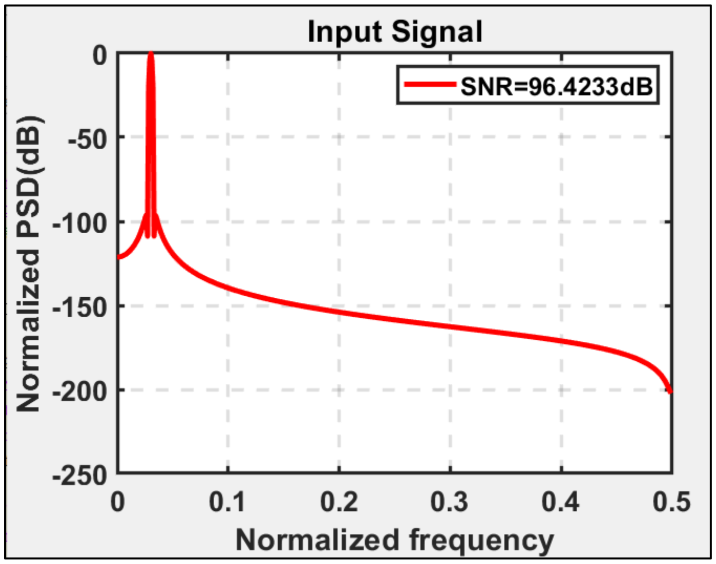


Figure 3.48 Input Spectrum

The input has SNR of 96.4dB and it then passes by the phase shifter giving a spectrum as shown at figure 3.49.

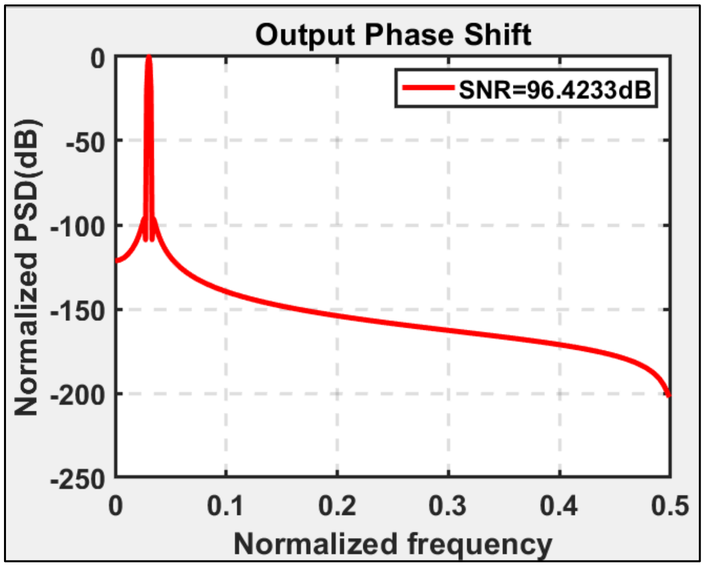


Figure 3.49 Phase Shifter Output Spectrum

Phase shifter output SNR and sampling frequency don't change from its input. This is expected as phase shifter doesn't change the signal spectrum. The phase shifter output is taken as input to the interpolation filter and produce the spectrum shown in figure 3.50.

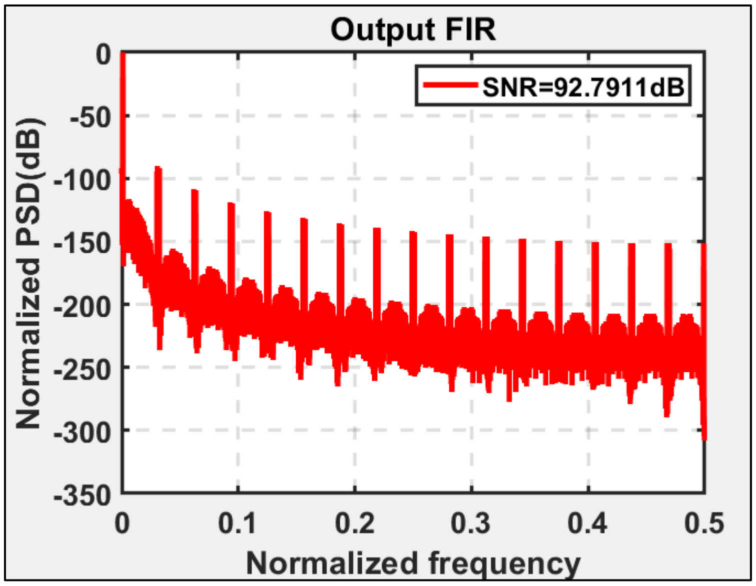


Figure 3.50 Interpolation Filter Output Spectrum

The interpolation filter output has SNR of 92.8dB and sampling frequency of $\frac{f_s}{4}$. the spectrum shows the suppressed images of the signal that are outside of our signal bandwidth. The filter output passes by the sigma delta block. The output of sigma delta is shown in figure 3.51.

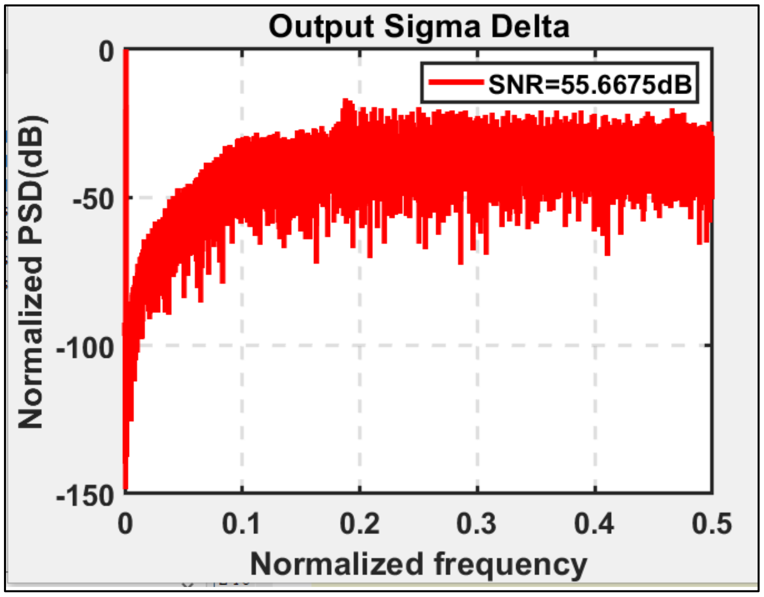


Figure 3.51 Sigma Delta Output Spectrum

The sigma delta output SNR is 55.7dB and sampling frequency same as its input $\frac{f_s}{4}$. The spectrum shows the noise shaping done to the input making most of the noise outside the band of interest of the signal. Now the signals of I and Q are ready to get multiplied by carriers of frequency $\frac{f_s}{4}$ in the DUC to be transmitted so the output of sigma delta is taken as DUC input. The output of DUC block is shown in figure 3.52.

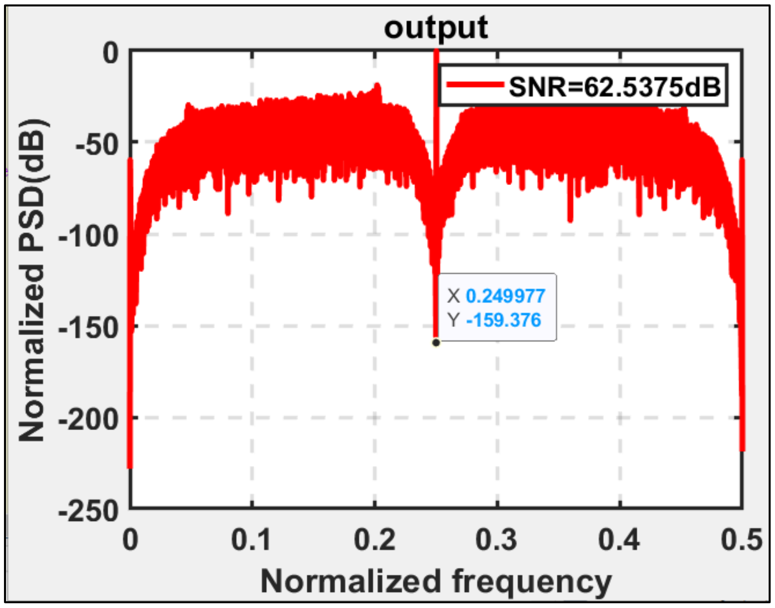


Figure 3.52 DUC Output Spectrum

The output of DUC has SNR of 62.2dB which fulfills the constrain of having output SNR more than 50dB mentioned above and it has a sampling frequency of f_s. The spectrum shows the multiplication of signal by carrier process where the pulse moved by the carrier frequency in the spectrum plot.

Table 3.3 summarizes the SNRs which were got from each stage.

Signal	**SNR (dB)**
Input	96.4
Phase Shifter Output	96.4
Interpolation Filter Output	92.8
Sigma Delta Output	55.7
DUC Output	62.5

Table 3.3 Blocks Output SNR

3.6.2 Multiple Chains Integration

After testing the single chain functionality and getting the SNRs and spectrums in this section the beamforming process would be tested by integrating multiple chains for the transmitter having different phase shifts. The phase shift for each chain as mentioned in chapter 2 is $(k-1)\Delta\varphi$ where k is the chain order and $\Delta\varphi$ is got from equation 3.14 which its details are mentioned in chapter 2.

$$\Delta\varphi = \frac{360^0\, d \sin\theta_s}{\lambda}$$

Equation 3.14 Phase different Between Adjacent Chains

The multiple chains architecture is as shown in figure 3.53 where the inputs I and Q enter to multiple chains of the single chain discussed in section 3.6.1 to be fed to multiple antennas to perform the digital beamforming concept and plot the directivity to test the achievement of the desired steering angle.

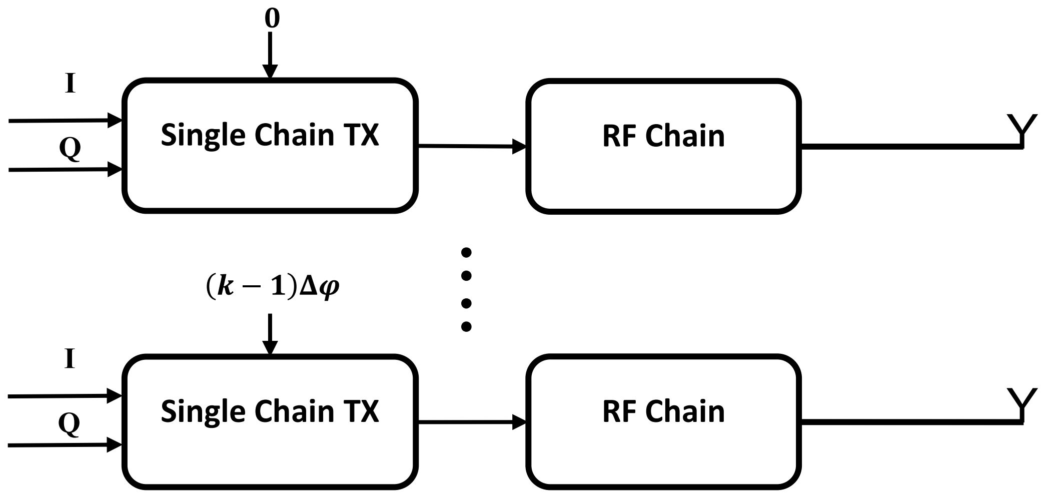


Figure 3.53 Multiple Chains Architecture

The test is done by transmitting the signal at different angles and plot the directivity. Figure 3.54 shows the directivity upon transmitting at steering angle 0

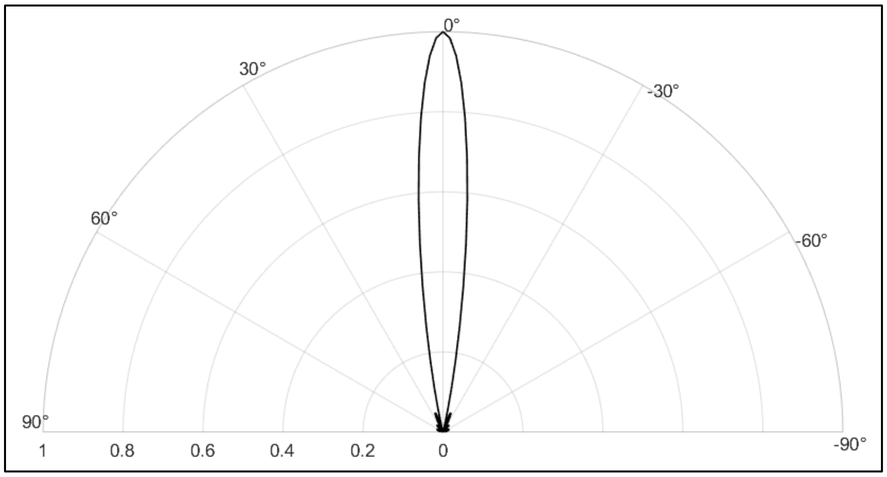


Figure 3.54 Directivity at Steering Angle 0-degree Using 8-Chains Transmitter

Figure 3.54 shows how the signal is transmitted at steering angle 0-degree. The signal power is directed in the direction of steering angle while destructive interference is done in the other direction making the signal directed in a narrow beam.

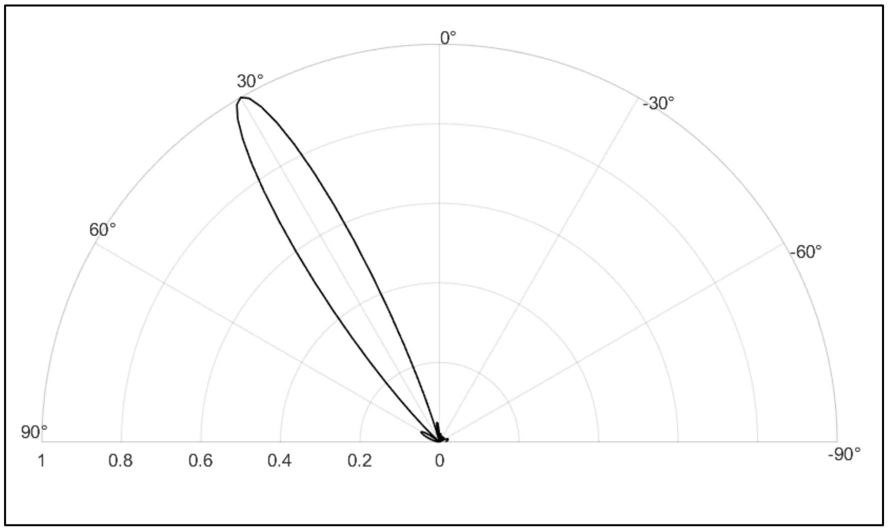


Figure 3.55 Directivity at Steering Angle 30-degrees Using 8-Chains Transmitter

Figure 3.55 shows the effect of changing the steering angles to 30-degrees. The beam now is directed in the new direction having constructive interference between signals in 30-degrees direction and destructive in other directions.

Figure 3.56 shows the effect of changing the steering angles to negative steering angle of -30-degrees.

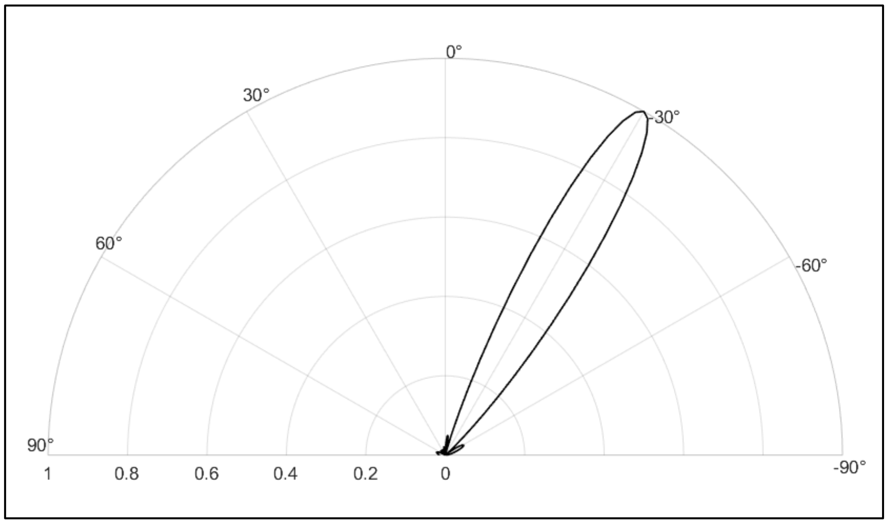


Figure 3.56 Directivity at Steering Angle -30-degrees Using 8-Chains Transmitter

Figure 3.57 show multiple beams of different steering angles.

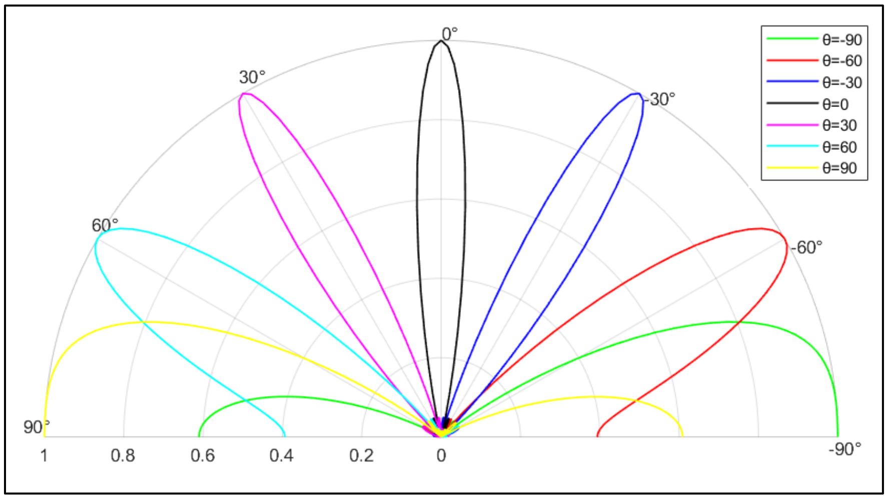


Figure 3.57 Multiple Beams of Multiple Steering Angles

After discussing the effect of changing steering angle on the transmitted beam. Figure 3.58 shows a comparison of transmission at steering angle of 0-degree at two architecture 2-chains beamforming transmitter and 8-chains beamforming transmitter.

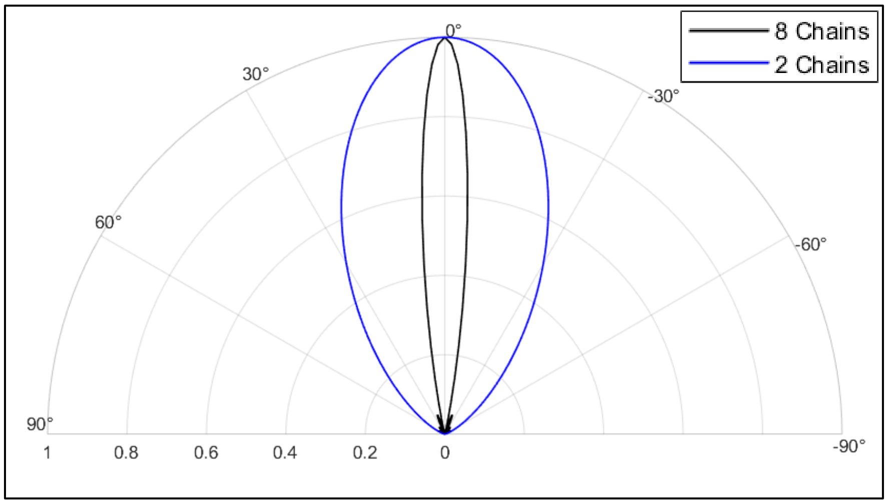


Figure 3.58 Beam at 2-chains and 8-chains for Steering Angle = 0

Figure 3.58 shows the effect of changing the number of chains used in the beamforming transmitter architecture. Using more chains increase the directivity of the beam. In case of using 8 chains at 0-degree steering angle the beam is narrower and more directed to the 0-degree direction. The attenuation at in other directions increases too leading to higher directivity comparing to the two chains architecture.

As the system modeling phase is done by implementing each block, making all design choices and doing chain and system integration, the next step that would be discussed in next chapter is the conversion of the system to fixed points system and do the RTL implementation to the beamforming transmitter system.

Chapter 4: RTL Implementation

4.1 Full system & HDL

This chapter mainly focuses on translating the abstract modelled system into a behavioral model describing the architecture of the design using synthesizable HDL constructs, which will then be verified using simulations to check if it is Functional and meets the system requirements.

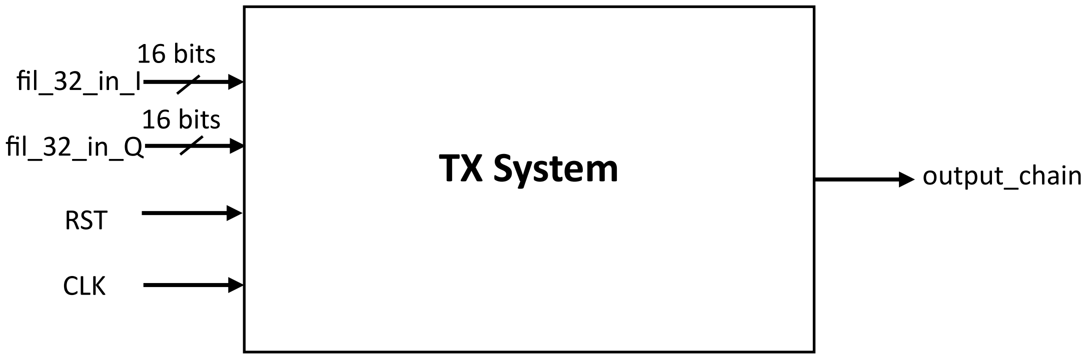


Figure 4.1 Full system ports

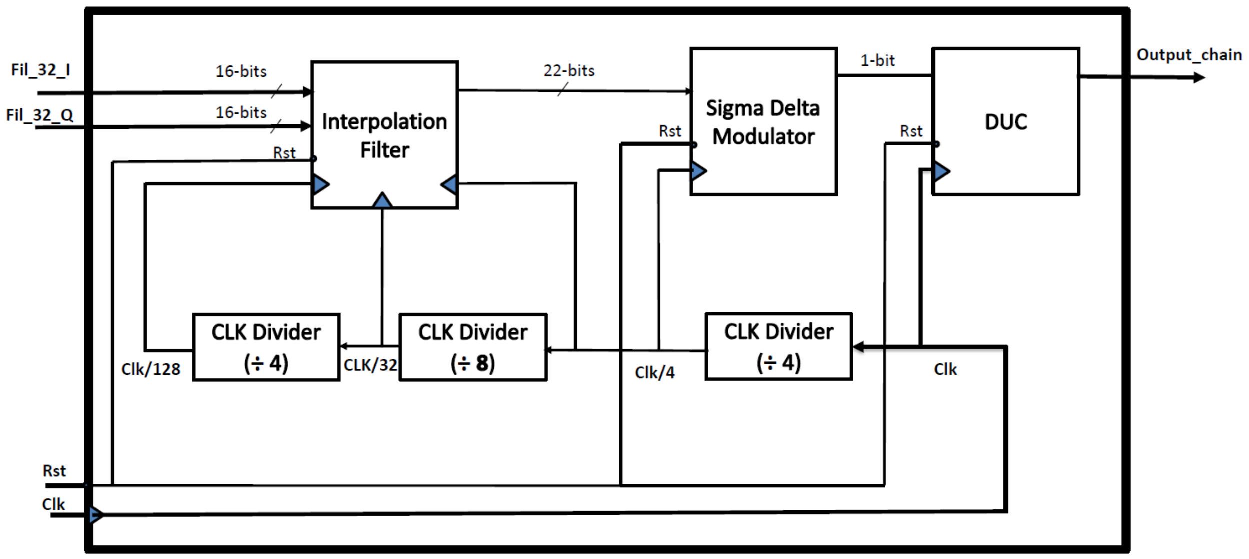


Figure 4.2 Full system internal Block diagram

Inputs to the System:

1. CLK: The main system clock that is connected to different flops and clock dividers inside the system.
2. RST: Hard reset signal of the system that resets the whole system asynchronously.
3. fil_32_in_I (16-bits): In phase signal input that passes through the 'I' data path in the system which is 16 bits in size.
4. fil_32_in_Q: Quadrature signal input that passes through the 'Q' data path in the system which is 16 bits in size.

Output of the System:

1. output_chain (1-bit): Output of the system which is produced from DUC block with size of 1-bit.

4.2 Phase Shifter

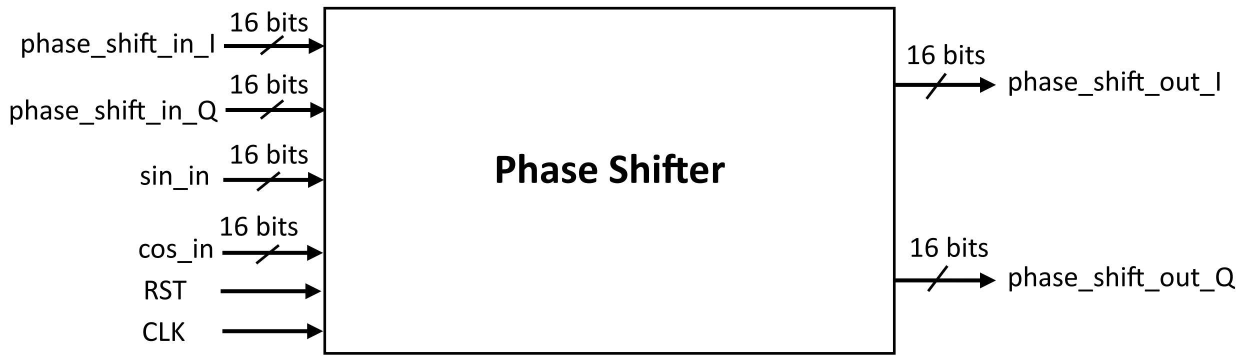


Figure 4.3 Phase Shifter Block

The implemented phase shifter block has 6 input and two output as shown in figure 4.3. The six inputs consist of two 16-bits inputs for the signals I and Q. Two 16 bits input for the sin_in and cos_in which correspond to the phase shift angle cosine and sin. Two inputs for the reset and clock signals in which the clock used in phase shifter is divided by 128 version of the master clock using the clock dividers blocks in the system. The block has two outputs represent the phase shifted versions of the input I and Q.

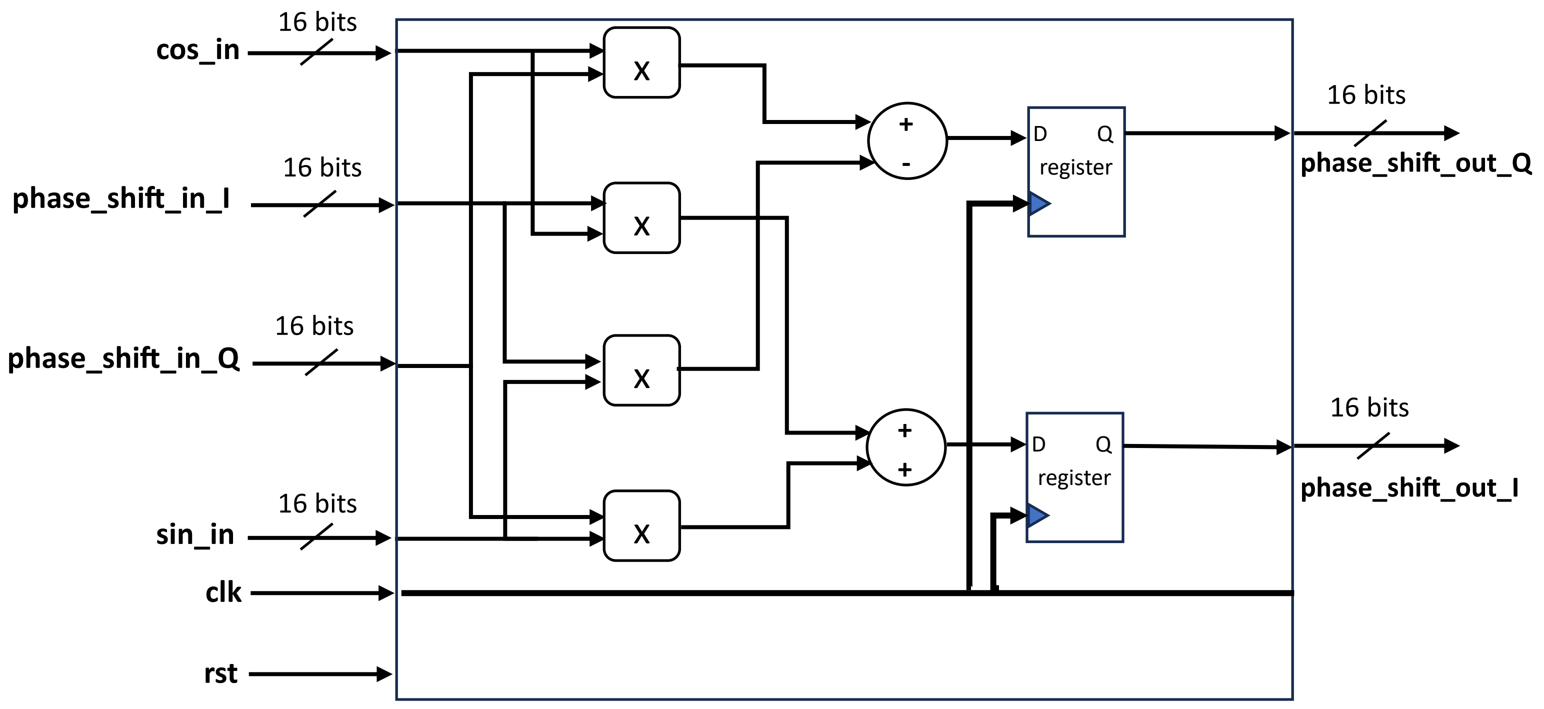


Figure 4.4 Phase Shifter Internal Architecture

RTL code were written for the architecture in figure 4.4 getting the result shown in figure 4.5 for the RTL phase shifter with shit angle of -30 degree.

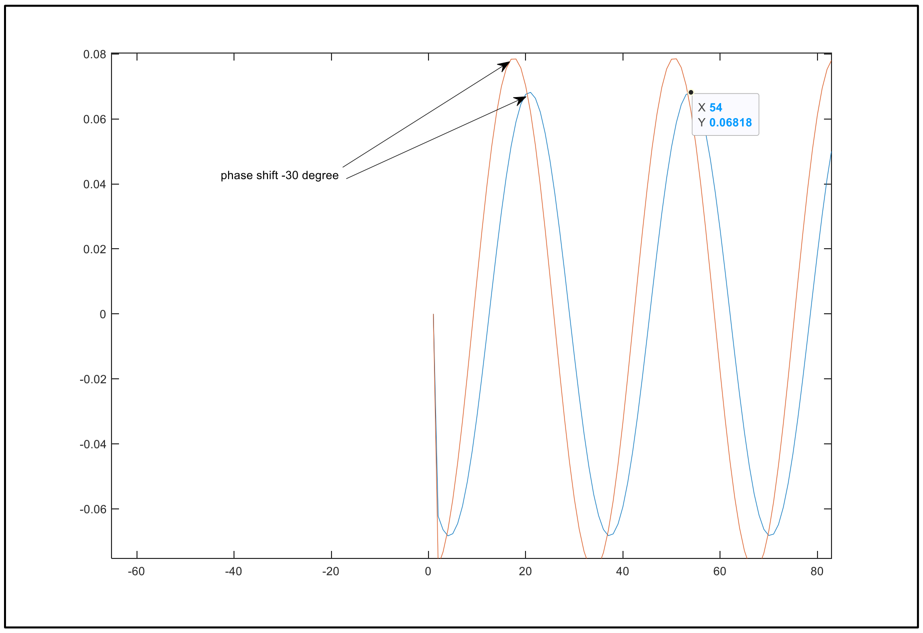


Figure 4.5 Phase Shifter RTL Result

4.3 Interpolation Filter

The interpolation filter consists of 2-stages of a comb filter which up samples the signal by ratio of 32 where first stage up samples by a ratio of 4 and the later up samples by a ratio of 8.The filter digital implementation will be different from the SIMULINK modeled one but with same functionality as the one that will be implemented digitally can be as the adders used inside the filter at the output port to sum all branches will be replaced with a mux operating at a speed equivalent to the up sampling ratio of that specific filter stage where in each clock cycle it samples output from a different branch rather than using the delay elements as in the model as this increases linearly with the filter up sampling ratio, so be replacing them it saves area and power consumption and this is illustrated more in the example.

This is an example of an interpolation filter with oversampling (OSR) ratio of 3 where first figure is the like the modelled design and second figure is the proposed model used which is equivalent to the first one.

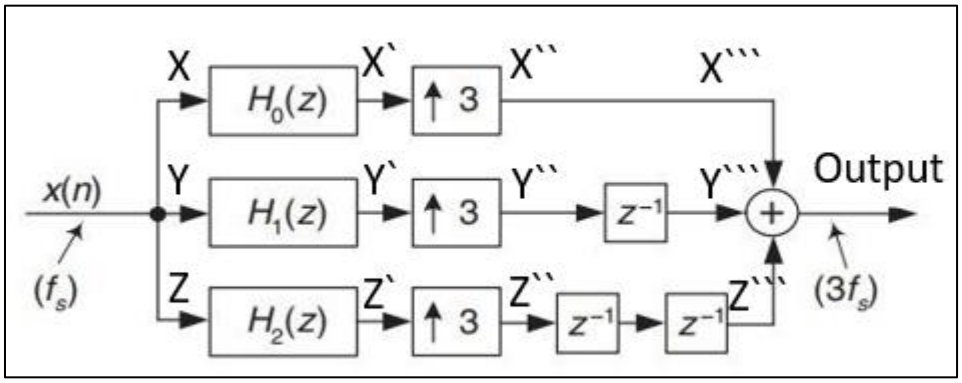


Figure 4.6 Modelled Architecture

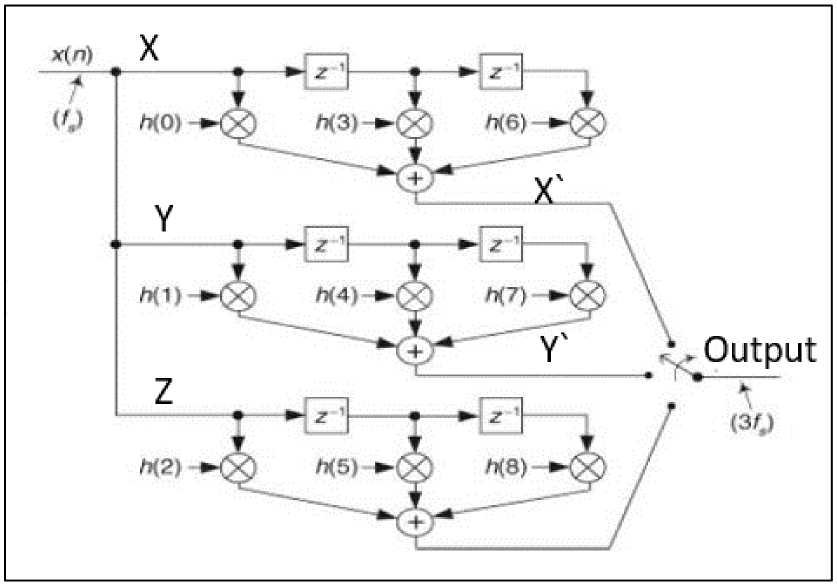


Figure 4.7 Optimized Equivalent Architecture

The idea behind this implementation is rather than up sampling by adding zeros to each branch in the filter and summing them each clock cycle but with different delays, you will just need to select a branch output in order every clock cycle but with a with a faster sampling frequency at the output than that at the input with a speed increase of the filter's up sampling ratio, so each filters' output branches will be connected to a switch which is a multiplexer with a selection line connected to a counter operating on that fast speed.

The equivalence between the two designs can be illustrated more through the following graphs and equations that proves the equivalence between both designs.

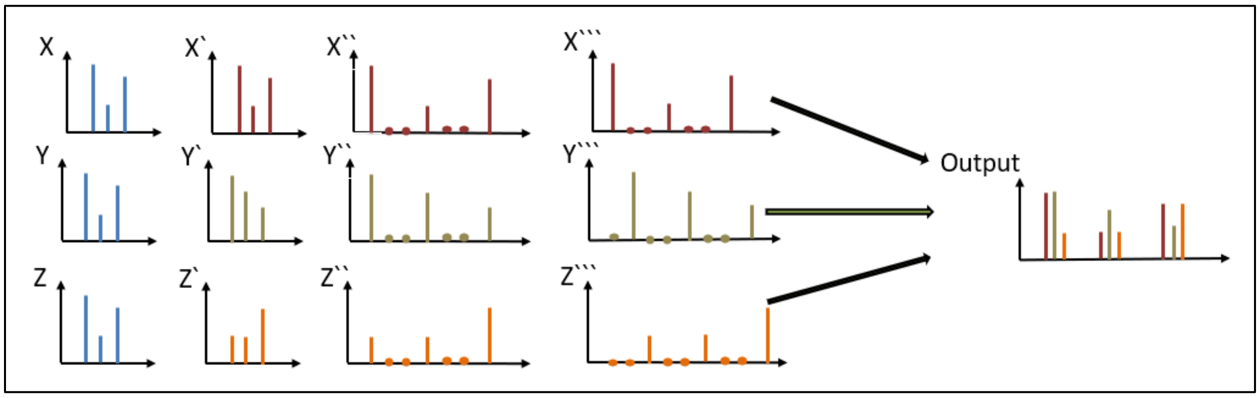


Figure 4.8 Optimized Equivalent Architecture

As shown above the branches after generating the sample it oversamples it by adding (L-1) zeros where L is the oversampling ratio and then add delay then summing these branches so as to avoid overlapping between the branches results, but you can select the output samples by selecting a different sample every clock cycle from the branch in order from the second column graphs (output of X', Y', Z').

The filter coefficients were chosen to give response of a comb filter due to its efficiency in multi-rate systems as it places null specifically on the images in the stopband region which are the main contributors in degrading the signal strength. Thus, after getting these coefficients polyphase implementation of the filter is done as discussed in previous chapters.

As shown in the following figure that 2-filters operate on different speeds (i.e., frequencies thus we need a clock divider to supply each block with its required speed). therefore, three clock speeds from the clock divider are needed.

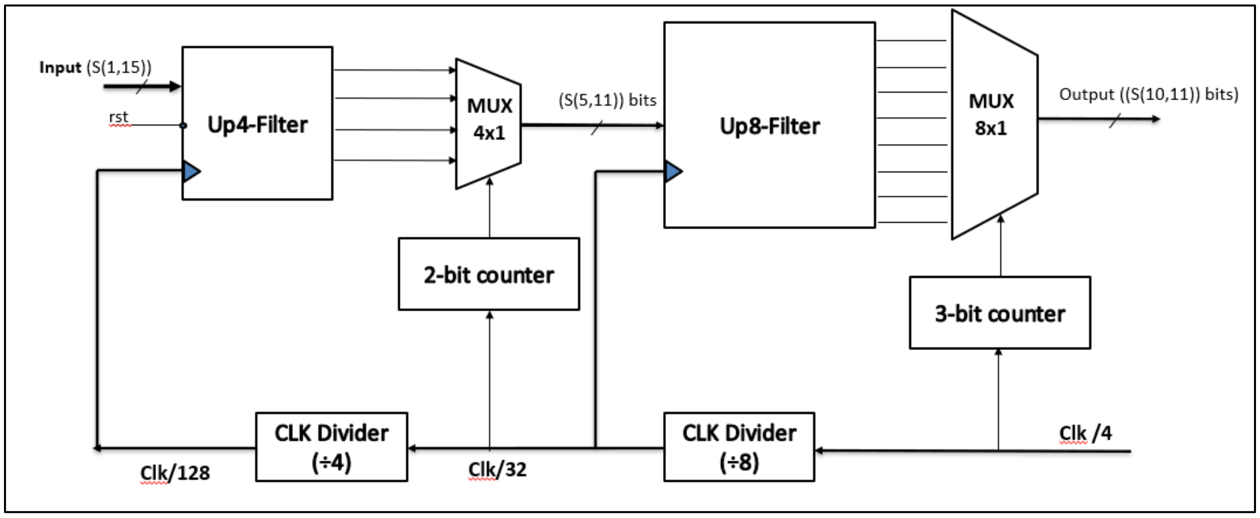


Figure 4.9 Filter Digital Implementation Block Diagram

The first filter stage up samples and filters the signal with OSR of '4' and needs the input to be sampled with a clock speed of CLK/128 and output to be sampled by quadrable the speed therefore CLK/32 speed is needed. The input data is sampled to filter branches and multiplied with the filters' coefficients, here the coefficients are integer constants so we don't need to use multiplier instead it can be replaced with shifters and adders therefore reducing area and power consumption. While the second filter stage up samples and filters the signal with OSR of '8' and needs the input to be sampled with a clock speed of CLK/32 and output to be sampled by eight times the speed therefore CLK/4 speed is needed as well.

4.4 Digital Sigma Delta

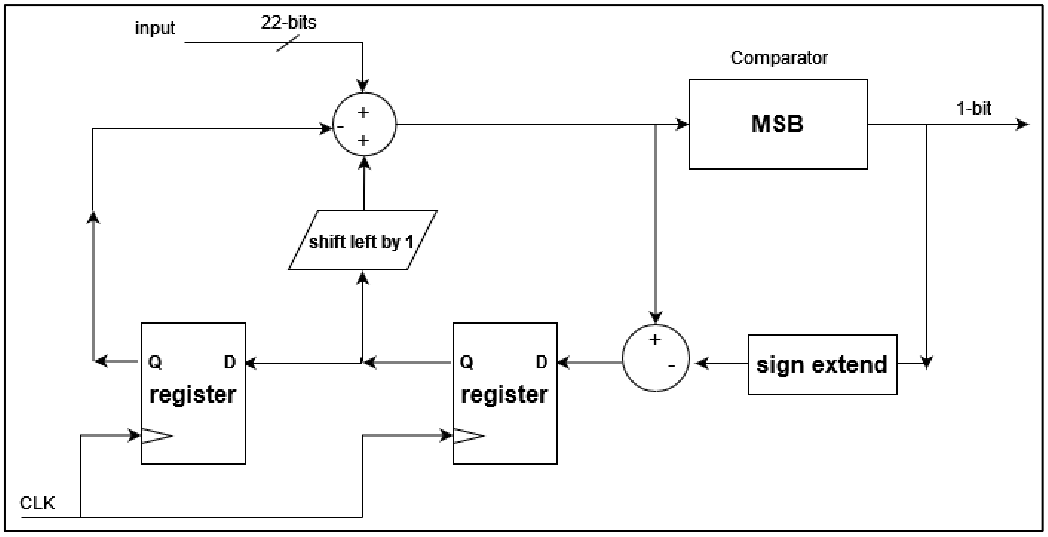


Figure 4.10 Digital architecture of EFB 2nd order sigma delta

The main design points that were taken into consideration during the RTL implementation phase of the sigma delta are the following:

First was to correctly represent the loop architecture of the sigma delta to achieve the main functionality like the system model.

The delay elements in the architecture are achieved through a register as it will delay the signal one clock cycle per register.

The comparator will simply be a bit extractor to extract the most significant bit (sign bit).

We replaced the multiply by 2 with a shift lift by 1 block.

One main point was to correctly feedback the signal with a sign extend block. In the implementation simply feedbacking the single bit won't do as we're dealing with a fixed-point representation of the data and to have a correct mathematical operation, we only will extend in the integer part of the bits and leave the fractions as zeros.

For more elaboration on the fixed-point representation of the data: -

Tha input of the sigma delta in our design is in 22-bits. As mentioned before in chapter 2 sigma delta can only deal with input dynamic range between 0:1 so for the sigma delta to operate correctly we have to deal with the input in a normalized manner and leave 1 sign bit and deal with the rest as a fraction.

However, this second order architecture needs 4-bits in the integer part for operation as it was demonstrated in the system modeling in chapter 3. So, we had two implementations' options. First, was to shift right the input by 4 and use these 4-bits as integer bits for operation of the second order sigma delta architecture. Second, was to extend the 22-bits to 26-bits for the operation of the sigma delta.

After analyzing both options we decided to use the first option as it produces same performance (i.e., SNR) and not need the extra 4-bits that would for sure add more area and power consumption in our design.

The output of the sigma delta is a single bit 0 representing -ve one and 1 representing +ve 1. So, when we feedback the single bit, the sign extend block will implicitly take the 0 and extend it to signed representation of the -ve one and the 1 extend it to signed representation of the +ve one for correct operation.

These were the main design considerations for the correct RTL implementation of the EFB 2nd order sigma delta architecture.

4.5 Digital Up Converter

For the Digital up conversion (DUC) block to be digitally designed like the architecture demonstrated in the figure 4.11 there were some design considerations which we will discuss:

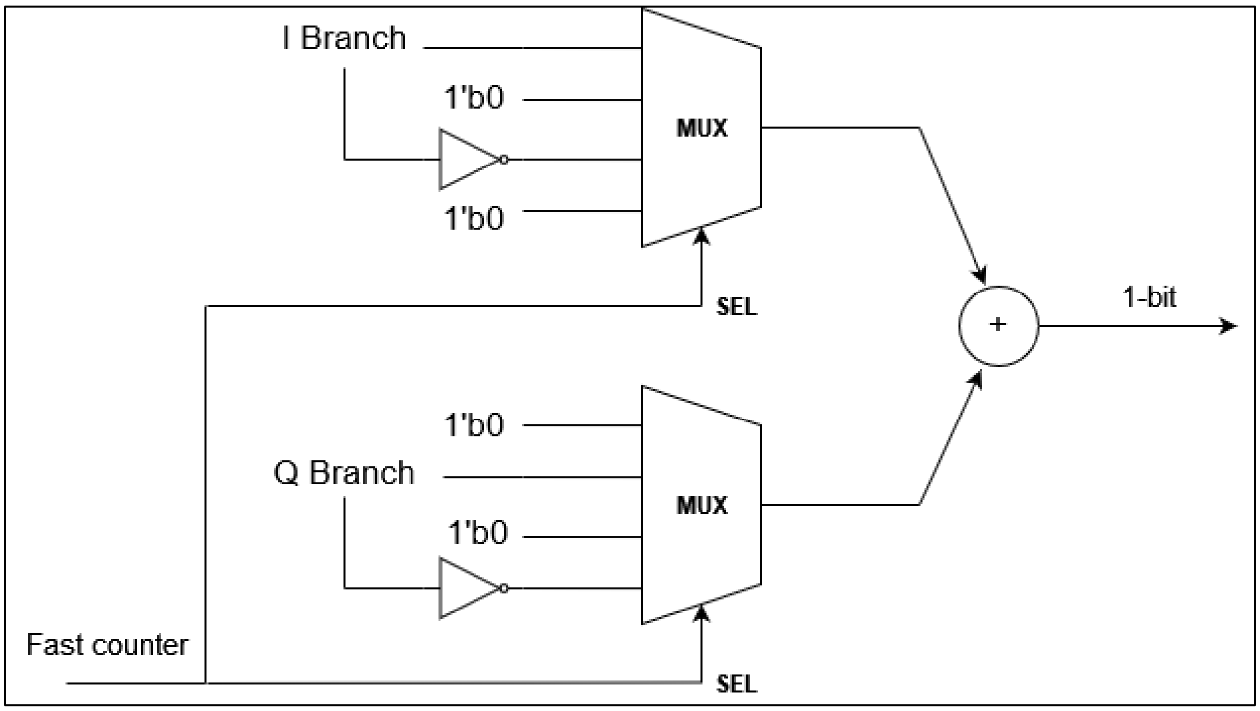


Figure 4.11 Digital architecture of DUC

First thing was using multiplexers as a replacement of the usual multiplications by both sin and cos carriers. By simply applying the following relation between the carrier frequency F_C and the sampling frequency F_S : -

$$F_c = 4\ F_s$$

We can guarantee that the carriers will only have 0, 1, -1 as their values. We can now use a multiplexer with a fast counter operating at the same frequency as the carrier frequency. The carrier value being 1 means that the signal will pass as it is. As the input is a single bit 0 representing -ve one and 1 representing +ve 1 then when the carrier value is -1 means that the signal is toggled and lastly when the carrier value is 0 just pass a 0.

You can see this algorithm implemented in the figure and if we take the first mux as an example it has 4 inputs and the carrier has a periodic sequence of {1, 0, -1, 0} which means that 4 inputs should be I-Branch then 0 then the inverted I-Branch value being the multiplied by -1 part and then 0 again.

As there is a 90-degree phase shift between the cos and sin the second mux is shifted in its periodic sequence which will be {0, 1, 0, -1}.

An important design point that could have introduced an issue in this design but was helped by the mere 90-degree phase shift between the two muxs and also adding both of them at the end is because we have three different values for both carriers that needs to be represented in bits a single bit shouldn't be enough and should cause a loss of information as two values will be represented with the same bit. Both {0, -1} will be 0.

The reason why we can ignore this potential issue is that when we add at the end the output of both muxs we can guarantee from the phase shift that every 0 at one of the muxs output will have a 1 or a -1 at the other mux output and adding both outputs will result either a 1 or a -1. So, the final output will only contain the values 1 or -1 which can be represented in just one bit.

4.6 Results

This section would cover the plots and results got from the RTL phase. The signal width for each stage is as shown in table 4.1.

Stages	**Number of bits**
Input signal	16 bits
Filter Input before up sampling	16 bits
Filter after up sampling by 8	22 bits
Sigma delta output	1 bit
DUC	1 bit

Table 4.1 RTL System Signals Width

The system works on master clock of 10ns that has generated clocks from it suitable for each block. For the single chain the output SNR for each block is calculated and shown in table 4.2 compared to Simulink system SNRs.

Signal	Simulink SNR (dB)	RTL SNR (dB)
Input	96.4	62.3
Phase Shifter Output	96.4	62.3
Interpolation Filter Output	92.8	57
Sigma Delta Output	55.7	55
DUC Output	62.5	60.5

Table 4.2 Simulink and RTL SNR

The output is one bit with SNR is more than 50 so the requirements are met. The output spectrum is as shown in figure 4.12.

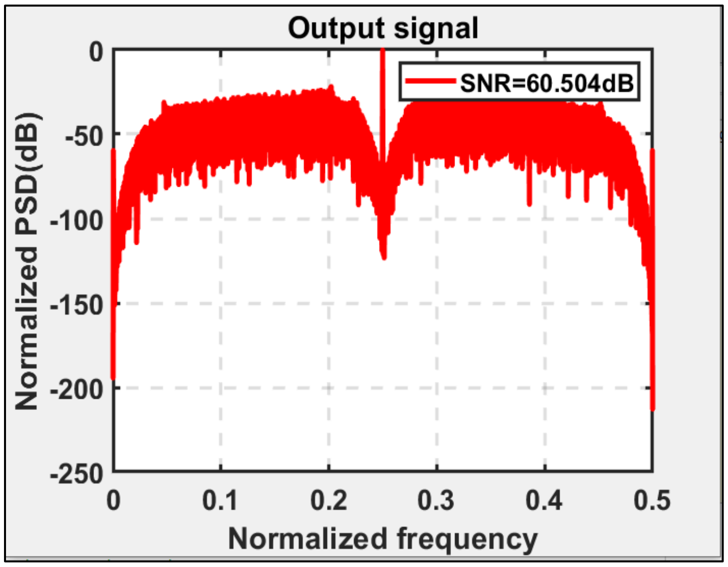


Figure 4.12 RTL Output Spectrum

The output spectrum shows the noise shaping effect done by sigma delta and the up conversion done by DUC. The next two chapter discusses the ASIC implementation of the digital beamforming transmitter.

Chapter 5: ASIC Flow

5.1 Introduction

ASICs are integrated circuits designed to perform specific functions tailored to a particular application. Unlike general-purpose processors, ASICs are customized to execute a dedicated set of tasks efficiently. They are optimized for performance, power consumption, area, and cost, making them an ideal choice for applications that demand high efficiency and reliability. The ASIC design flow represents a structured methodology that encompasses all the stages involved in creating a custom ASIC. It involves a series of steps from concept to production, ensuring the final product meets the desired specifications. While the specific flow may vary depending on the design requirements and the design team's preferences, the general ASIC flow typically includes the following stages shown in figure 6.1:

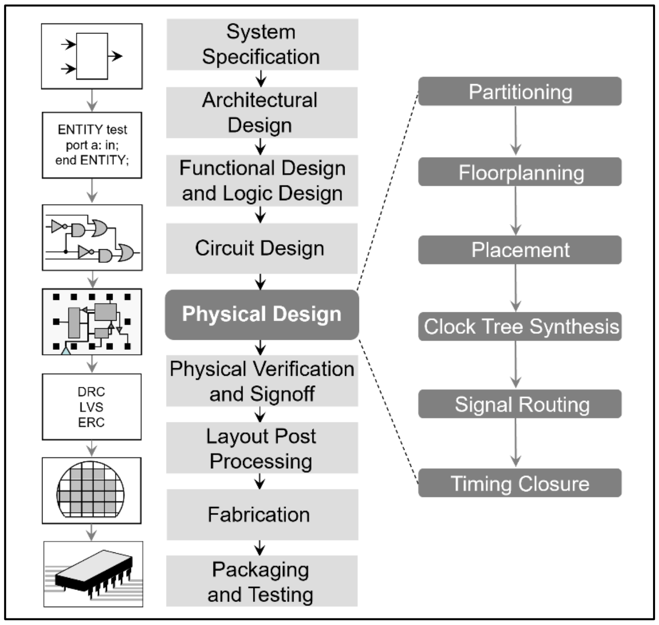


Figure 5.1 ASIC Flow Stages

Specification and Architectural Design: The initial stage involves defining the system requirements and functional specifications. The design team collaborates with the client or project stakeholders to determine the desired features, performance targets, and constraints. This phase also includes architectural design decisions that establish the overall structure and functionality of the ASIC.

a. RTL Design and Functional Verification: Once the specifications are finalized, the team proceeds to register-transfer level (RTL) design. RTL represents the behavior and functionality of the ASIC using hardware description languages (HDLs) such as Verilog or VHDL. Functional verification ensures that the RTL design accurately implements the desired functionality and is free from design flaws or errors.

b. Design Synthesis and Optimization: In this stage, the RTL code is transformed into a gate-level representation through design synthesis. Synthesis tools map the RTL description to a library of standard cells and optimize the design for power, performance, and area. The output of synthesis is a gate-level netlist that represents the logical structure of the ASIC.
c. Physical Design and Layout: Physical design involves converting the gate-level netlist into a physical layout that can be fabricated. It encompasses activities such as floor planning,

placement, clock tree synthesis, and routing. The goal is to optimize the layout for area, timing, power, and manufacturability.

d. Design Verification: After completing the physical design, extensive verification is conducted to ensure the design functions correctly in different scenarios and conditions. This includes simulation, formal verification, and other advanced verification techniques to detect and eliminate potential design flaws.

e. Design for Manufacturing (DFM): DFM considerations aim to enhance the manufacturability and yield of the ASIC. It involves analyzing the layout for potential manufacturing issues such as lithography, electromigration, and other process variations. DFM techniques are applied to optimize the design and minimize potential fabrication challenges.

f. Tapeout and Fabrication: Once the design is thoroughly verified and meets all the requirements, it is prepared for tapeout—the process of generating the final set of files required for fabrication. These files include the masks and photomasks used in the semiconductor fabrication process. The ASIC design is then sent to a semiconductor foundry for manufacturing.

g. Testing and Post-Silicon Validation: Once the fabricated ASIC chips are obtained, they undergo rigorous testing and validation to ensure they meet the performance and functionality requirements. This stage involves various testing methodologies, including functional testing, timing analysis, and power analysis.

5.2 Synthesis

In the context of ASIC design flow, synthesis involves transforming and mapping the high-level hardware description language (HDL) code, such as VHDL or Verilog, into a gate-level representation. This gate-level representation consists of individual gates, flip-flops, and their interconnections.

The synthesis tool analyzes the HDL code and performs various transformations and optimizations to generate an optimized gate-level netlist. This netlist represents the design in terms of specific gates available in the target technology library.

The transformations and optimizations performed during synthesis include logic restructuring, resource sharing, constant propagation, and technology mapping. These optimizations aim to improve the design's performance, area utilization, power consumption, and other parameters.

The technology mapping step involves selecting the appropriate gates or cells from the target technology library to implement the desired functionality. The synthesis tool maps the design elements in the HDL code to specific gates or cells that best fit the design requirements, considering factors such as timing, area, power and other constraints.

Input files to the synthesis stage typically include:

1. Verilog or VHDL Code File: This file contains the hardware description language (HDL) code that represents the design at a high level. The Verilog or VHDL code describes the functionality and behavior of the digital circuit to be implemented.
2. Technology Library: The technology library consists of a collection of pre-characterized cells or gates provided by the semiconductor foundry or technology provider. The library includes various logic gates, flip-flops, arithmetic units, and other components that can be used to implement the design. The library contains information about the timing, power, area, and other characteristics of each cell.
3. Constraints File: The constraints file contains additional design constraints that guide the synthesis tool during the optimization and mapping process. These constraints include timing constraints, area constraints, power constraints, and any specific requirements or limitations for the design. Constraints help ensure that the resulting design meets the desired specifications.

Output files of the synthesis stage typically include:

1. Gate-Level Netlist: The gate-level netlist is the primary output of the synthesis process. It represents the design in terms of individual gates, flip-flops, and their interconnections. The gate-level netlist specifies the logical structure of the circuit and serves as input for subsequent stages of the design flow, such as placement and routing.
2. Timing Reports: Synthesis tools provide timing analysis reports that give insights into the timing characteristics of the design. These reports include information such as critical paths, setup and hold times, and slack values. Timing reports help identify any timing violations or areas where the design may not meet the required timing constraints.
3. Area Reports: Area reports provide information on the estimated area utilization of the design after synthesis. They include details about the number and types of gates used, flip-flops, and other components. Area reports help assess the design's resource utilization and can be used to optimize for area if required.
4. Power Reports: Power reports provide estimates of power consumption for the synthesized design. They include information on dynamic power, leakage power, and switching activity. Power reports are useful for power optimization and analyzing the power characteristics of the design.
5. Constraints Check Reports: These reports provide feedback on whether the design meets the specified constraints. They include details about any violations or warnings related to timing, area, or other constraints defined in the constraints file.
6. Design Representation Files: Synthesis tools may also generate additional files that represent the design graphically or in other formats. These files can include schematic

representations, RTL (Register Transfer Level) diagrams, or other visual representations that help in understanding the synthesized design.

The RTL code of a digital system was synthesized using the Design Compiler synopsis tool with the TSMC 130 nm process technology PDK.

Design Compiler Mega Commands [11]

1. Read_file -format Verilog: import RTL Files and translate the intermediate design into a technology independent logic design using generic technology (GTECH).
2. Target_library: variable specifies the library that Design Compiler uses to select cells for optimization and mapping.
3. Link: Linking All the Design Parts.
4. check_design: checking design consistency.
5. Compile_ultra: this command will perform timing-driven optimization with high effort for the design hierarchy rooted at the specified.
6. Write_file: Generate the gate level netlist.

Following the synthesis process, log file analysis was conducted to detect critical paths and identify potential timing violations. Slack, area, and power consumption were also optimized using techniques such as logic restructuring and pipelining.

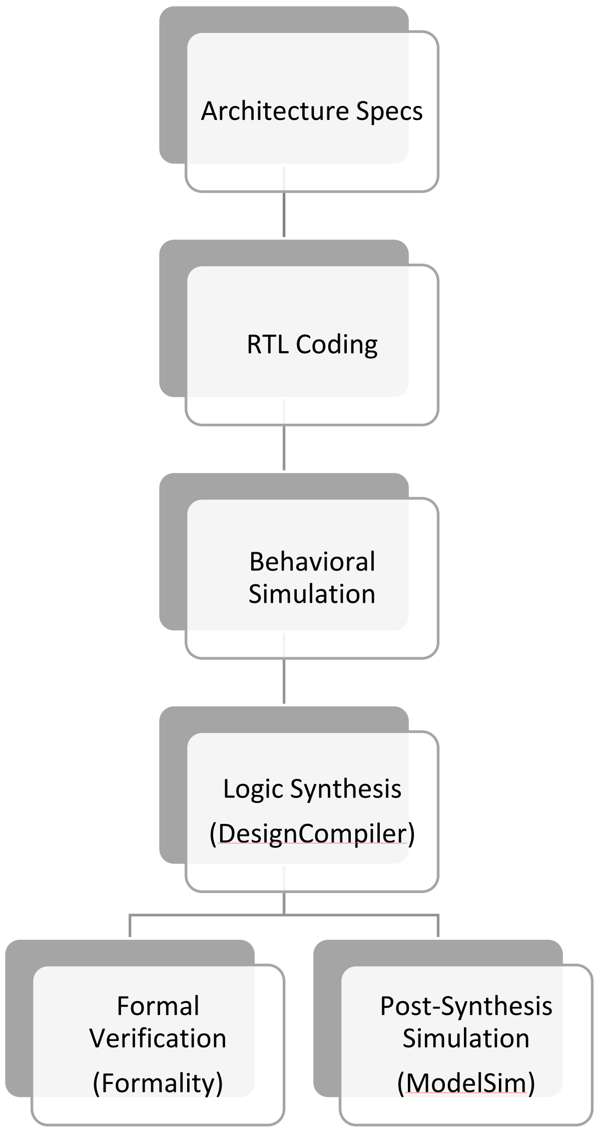

Architecture Specs
RTL Coding
Behavioral Simulation
Logic Synthesis (DesignCompiler)
Formal Verification (Formality)
Post-Synthesis Simulation (ModelSim)

Figure 5.2 ASIC Design Flow

5.2.1 Design Constraints

5.2.1.1 Clock Definitions

Master clock	
Clock period	2.5 ns
Clock uncertainty setup	0.02 ns
Clock uncertainty hold	0.01 ns
Clock transition rise	0.05 ns
Clock uncertainty fall	0.05 ns

Table 5.1 Master Clock Definition

5.2.1.2 Input/Output Delay

Input delay	0.2 * 2.5 ns
Output delay	0.2 * 2.5 ns

Table 5.2 Input/Output Delay on Ports

5.2.1.3 Driving Cell and Cap Load

Driving cell	BUFX2M
Cap Load	0.8 PF

Table 5.3 Driving Cell and Cap Load

5.2.2 Reports

5.2.2.1 Area Report

Area	
Total Cell Area	143,954 μm^2
Total Area	2,507,162 μm^2

Table 5.4 Area Report

5.2.2.2 Power Report

Power	
Total Power	17.412 mW

Table 5.5 Power Report

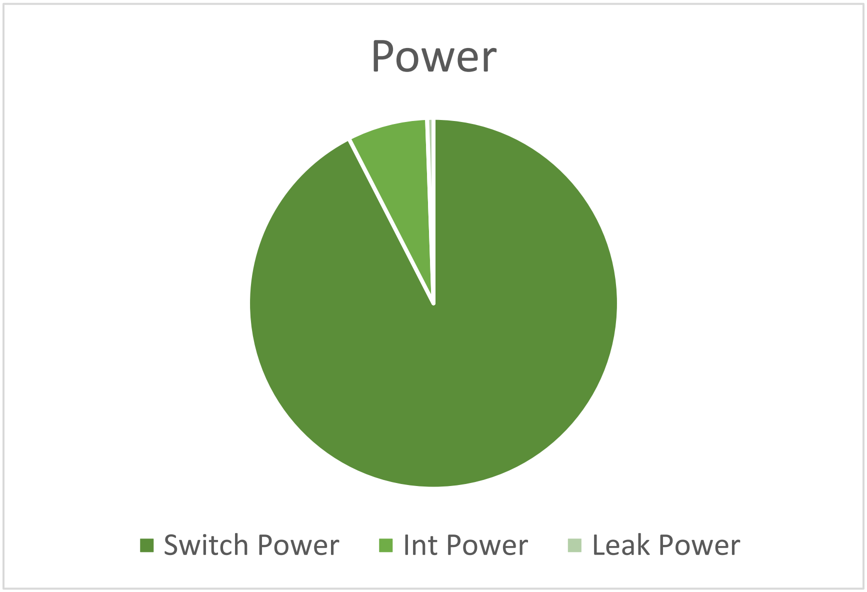


Figure 5.3 Power Distribution

5.2.2.3 Setup Report

	Start point	End point	Slack
Critical path	SigmadeltaQ/feed_back_reg[0]	SigmadeltaQ/feed_back_reg[0]	0.04 ns (met)

Table 5.6 Setup Report

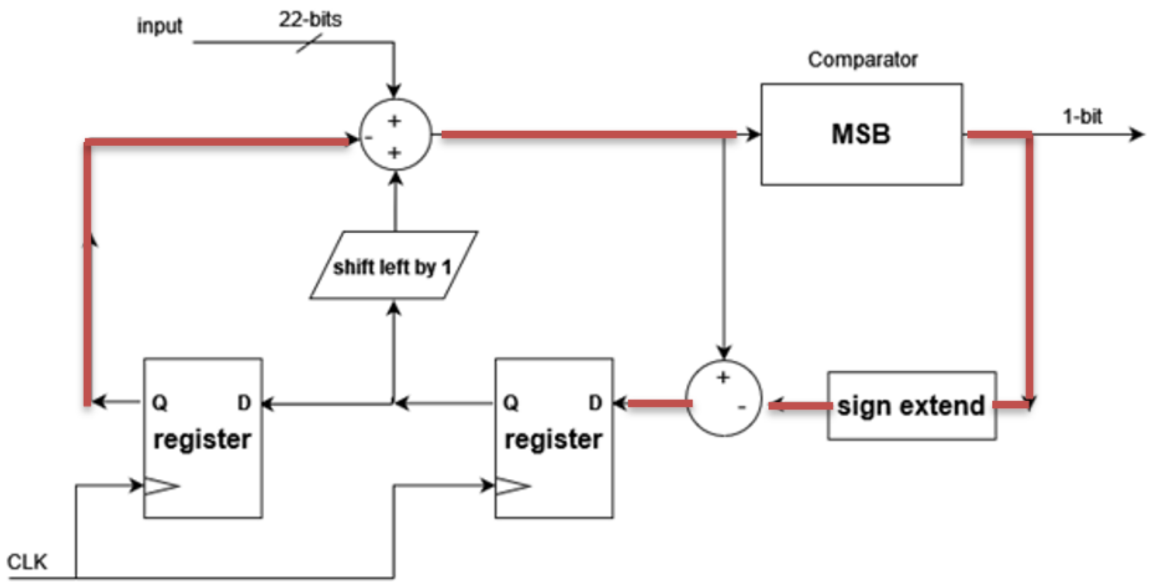


Figure 5.4 Critical path (setup)

5.2.2.4 Hold Report

	Start point	End point	Slack
Critical path	Fil_32_In_I[15]	Up_32_top_1/F4_I/b1_I/convert_reg[15]	0.31 ns (met)

Table 5.7 Hold Report

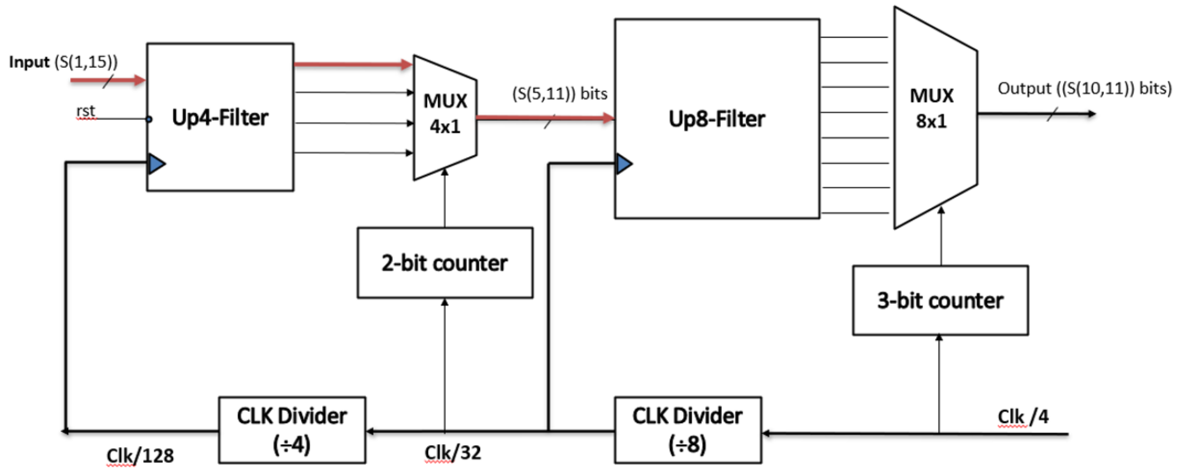


Figure 5.5 Critical path (setup)

5.2.3 Netlist

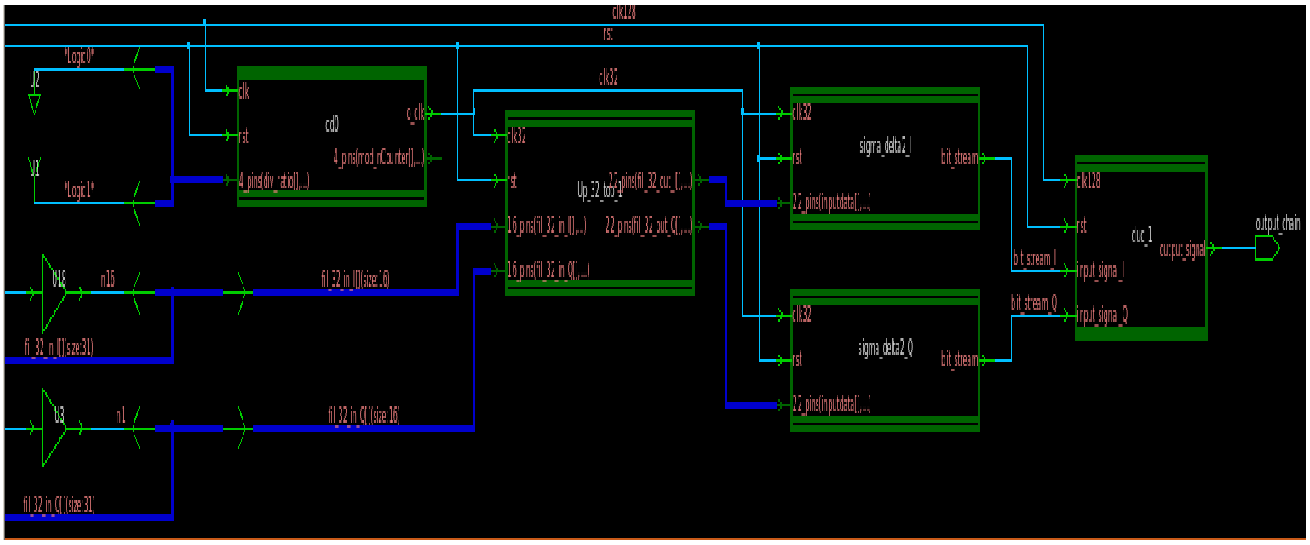
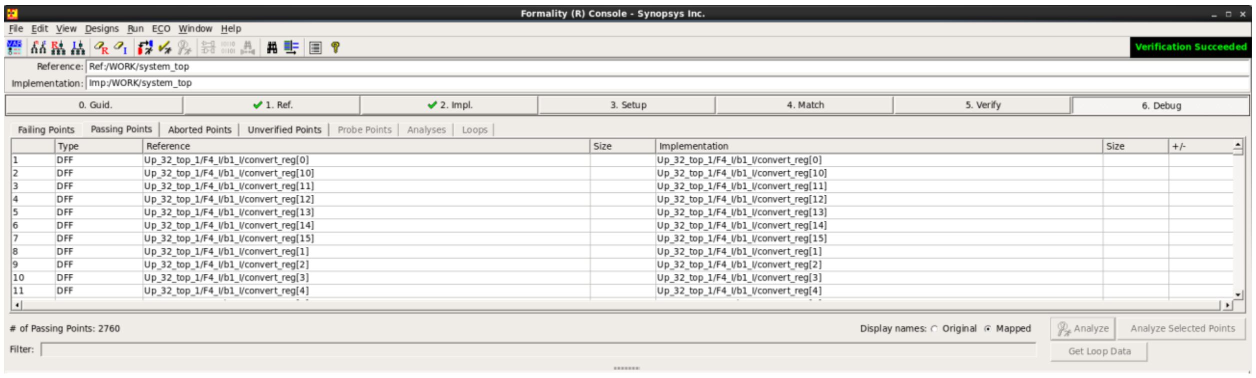


Figure 5.6 Single Chain Netlist

5.3 Formal Verification (Formality tool)

This chapter discusses the Formal verification done on the post-synthesis netlist generated, to check whether the synthesis tool had mistakenly changed the design architecture during synthesis process or not. The formal verification is done by comparing between the initial design (reference design) and the generated netlist of the design (implementation) to ensure that both of them are the same, where the tool makes a mathematical model for each design such as binary decision trees (BDD) to generate logic cones for every design to prove or disprove the equivalence between those two models through these cones. This checking is done statically, not depending on any timing constraints just functionality testing through equivalence checking between the designs.

The formal verification was done using a TCL script by assigning the initial design top module to reference model as well as adding the rest of the design modules, then assigning generated netlist to implementation model as well as adding the SVF file as a guidance file for the tool to ease its checking and finally let the tool do the checking and verification on the design. The following figure shows that the design successfully passed the formal verification process.



Verification Succeeded

Reference: Ref:/WORK/system_top
Implementation: Imp:/WORK/system_top

	Type	Reference	Size	Implementation	Size	+/-
1	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[0]		Up_32_top_1/F4_I/b1_I/convert_reg[0]		
2	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[10]		Up_32_top_1/F4_I/b1_I/convert_reg[10]		
3	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[11]		Up_32_top_1/F4_I/b1_I/convert_reg[11]		
4	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[12]		Up_32_top_1/F4_I/b1_I/convert_reg[12]		
5	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[13]		Up_32_top_1/F4_I/b1_I/convert_reg[13]		
6	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[14]		Up_32_top_1/F4_I/b1_I/convert_reg[14]		
7	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[15]		Up_32_top_1/F4_I/b1_I/convert_reg[15]		
8	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[1]		Up_32_top_1/F4_I/b1_I/convert_reg[1]		
9	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[2]		Up_32_top_1/F4_I/b1_I/convert_reg[2]		
10	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[3]		Up_32_top_1/F4_I/b1_I/convert_reg[3]		
11	DFF	Up_32_top_1/F4_I/b1_I/convert_reg[4]		Up_32_top_1/F4_I/b1_I/convert_reg[4]		

# of Passing Points: 2760

Display names: Original [✓ Mapped]

Filter:

Figure 5.7 Result of Formal Verification

5.4 PNR

Importing Files:

1. Netlist File (Verilog file)
2. Physical Library Files (LEF Files):

   - Technology LEF File (tsmc13fsg_8lm_tech.lef): Contains information about the metal layers, vias, design rules, etc. specific to the technology used.

   - Standard Cell LEF File (tsmc13_m_macros.lef): Contains the physical view of the standard cells in terms of the current technology.

3. View Definition File:

This file defines the Multi-Mode Multi Corner (MMMC) view and includes the following:

- Timing library files: Contain information about cell delay, input and output delays, etc.

- Capacitance Tables: Provide information about the capacitance values of the design elements.

- SDC files: Specify the timing constraints for the design.

5.4.1 Floor planning

When it comes to floor planning in integrated circuit (IC) design, there are several key considerations, including the placement of I/O, specifying the core area, specifying macros, and accommodating multiple metal layers. Here's a general outline of the floor planning process for your target specifications:

1. Placement of I/O:
    a. Identify the input pins: Determine the input pins required for your design.
    b. Specify metal layer 2 for input pins: Assign the metal layer 2 as the preferred layer for the input ports.
    c. Identify the output pin: Determine the output pin or pins that need to be placed on metal layer 3.
    d. Specify metal layer 3 for the output pin: Assign metal layer 3 as the preferred layer for the output ports.
2. Core Area Specification:

    - Define the core area, which is the central region of the chip reserved for placing the functional blocks, such as digital and analog circuits, memories, and processors.

    - The size of the core area is 600 for width and 500 for length so aspect ratio is 600/500, also we made the chip absolute "the die area equal to the core area".

3. Metal Layer Planning:

    - Determine the number of metal layers available for routing signals and power distribution. In this case, you have eight metal layers.

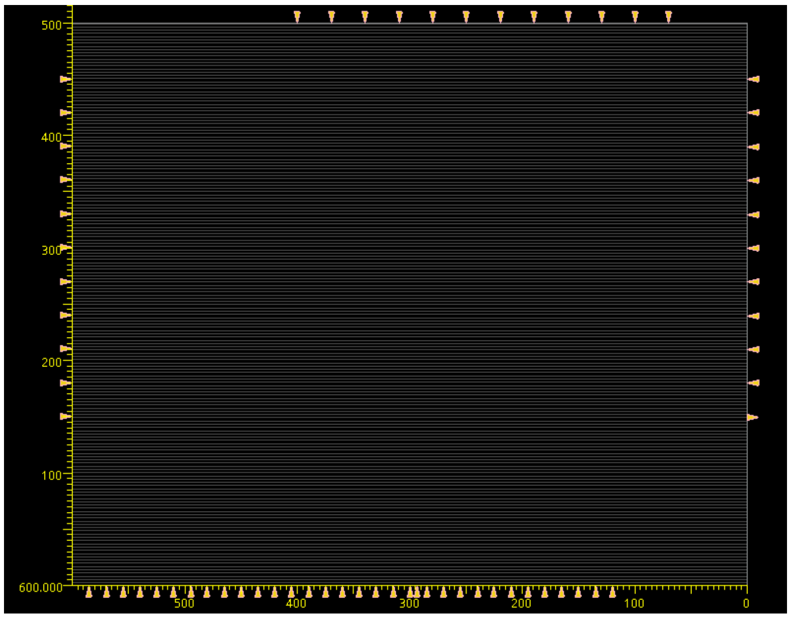


Figure 5.8 Floor Planning Core Area 600x500

5.4.2 Power planning

Power planning is an essential aspect of integrated circuit (IC) design to ensure proper distribution of power supply and efficient power delivery to various circuit components. Here's a general approach to power planning considering the specifications:

1. Rings on Layer 7:

   - Place power rings on the left and right sides of metal layer 7 to provide power supply to the circuitry. These rings are typically implemented as a continuous loop of metal, creating a power grid structure.

   - Ensure that the power rings cover the entire chip area and are connected to the power distribution network (PDN) for efficient power delivery.

2. Rings on Layer 8:

   - Add additional power rings on the top and bottom sides of metal layer 8. These rings serve the same purpose as the rings on layer 7, providing power supply to the circuitry.

   - Similar to the rings on layer 7, ensure that the rings on layer 8 cover the entire chip area and are connected to the PDN.

3. Stripes on Metal Layer 6:

   - Implement power stripes on metal layer 6 to enhance power distribution and reduce voltage drop across the chip.

4. Power Mesh:

   - Establish a power mesh network throughout the chip, connecting the power rings, stripes, and other power distribution structures.

   - The power mesh consists of horizontal power lines that form a grid-like structure, ensuring a robust power distribution network.

5. Verify Design Rules and Constraints:

   - Validate that the power rings, stripes, and mesh comply with the design rules and constraints specified by the technology and foundry.

   - Ensure that the power distribution structures do not violate spacing, width, and other layout rules.

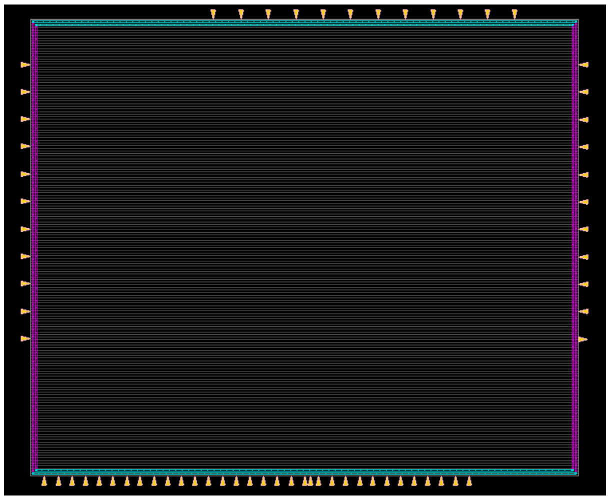


Figure 5.9 Power Planning

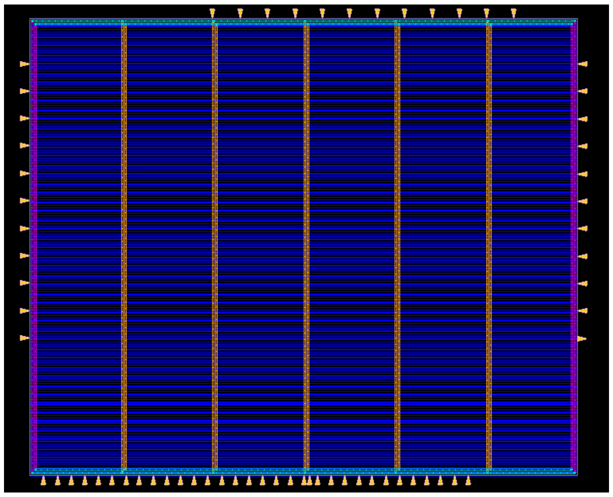


Figure 5.10 After Floor and Power Planning

5.4.3 Placement

Placement Process mainly focus on placing the standard cell inside the core boundary in the optimal location this occur in two main stages which is

- Global Placement: in this stage the tool will divide the cells into bins to try to minimize the number of connections between them that's why the tool won't check overlapping between in instances in this stage.
- Detailed Placement [Legalization]: in this stage the tool will place the cells in the optimal core site by providing legal placement for each instance.

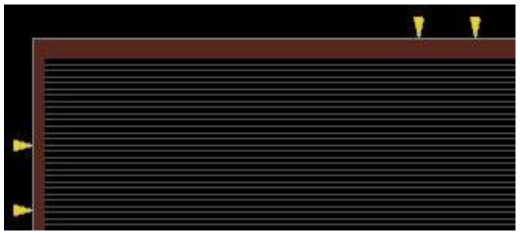


Figure 5.11 Placement Blockage

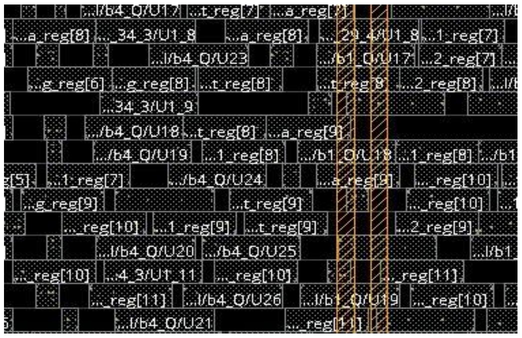


Figure 5.12 Std Cells Placement

Placement process should be provided with the required outcomes from previous process as Netlist of gates and wires, Floorplan and Technology Constraints to get the required output which is making sure that all Cells legally located in the optimal place.

Main of goals of the Placement Process is

- Provide legal and optimal location of netlist
- Reduce Routing Congestion and Placement Density
- Achieve best timing, area, power

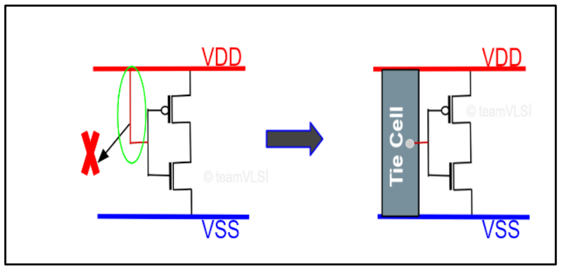


Figure 5.13 Tie Cells

Placement in this project is done by

1. Adding Placement Blockage around the core area to avoid routing congestion with the I/O Ports as shown in figure 5.9
2. Placing of the Standard Cells as shown in figure 5.10
3. Placing Tie-Low and Tie-High Cells to avoid any glitches that arises in the supply voltage that can lead to damage of gate oxide of the transistor

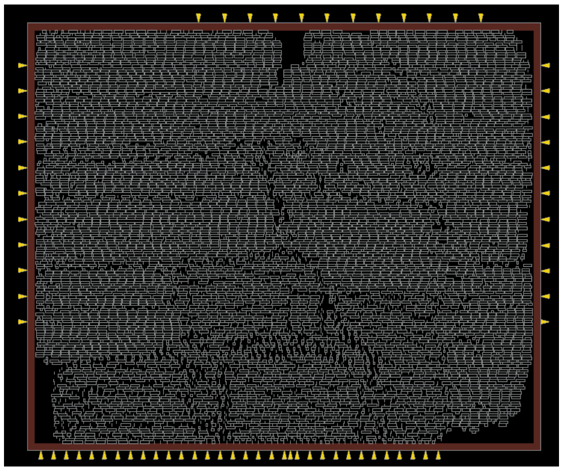


Figure 5.14 Chip after Placement

5.4.4 Clock Tree Synthesis

CTS is building a buffer/inverter network in order to balance the relative delays of flip flops belonging to a clock domain that are triggered by the same clock in order to minimize the clock skew and minimize the insertion delay too

Option Specified on CTS in this project

Options	Value
Clk	400 MHz
Max skew	0.5 nsec
Max Transition	0.25 nsec
Use Metal Layer	Metal 2:5

Table 5.8 CTS Constraints 1

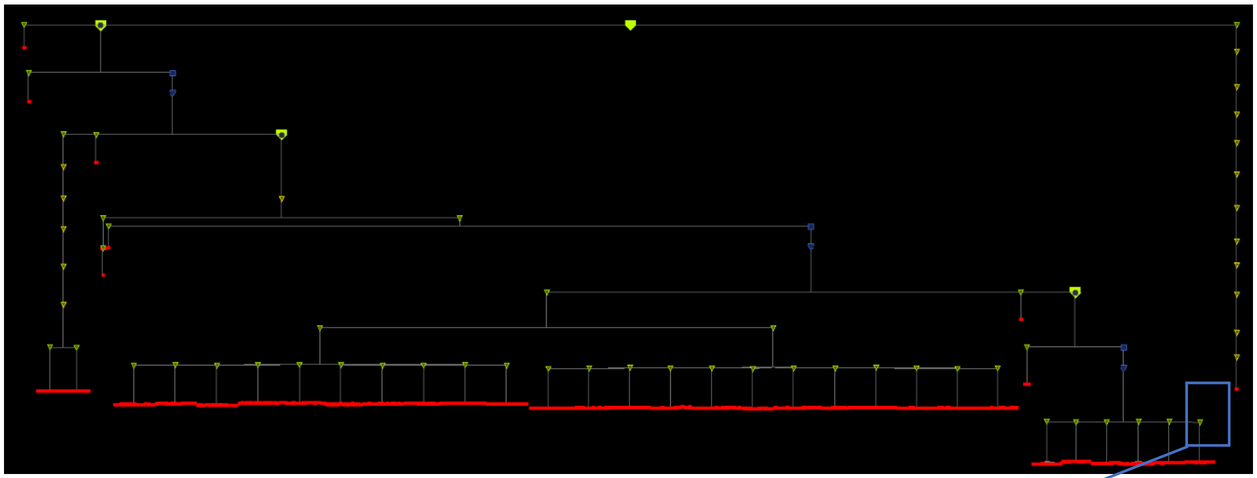


Figure 5.15 CTS

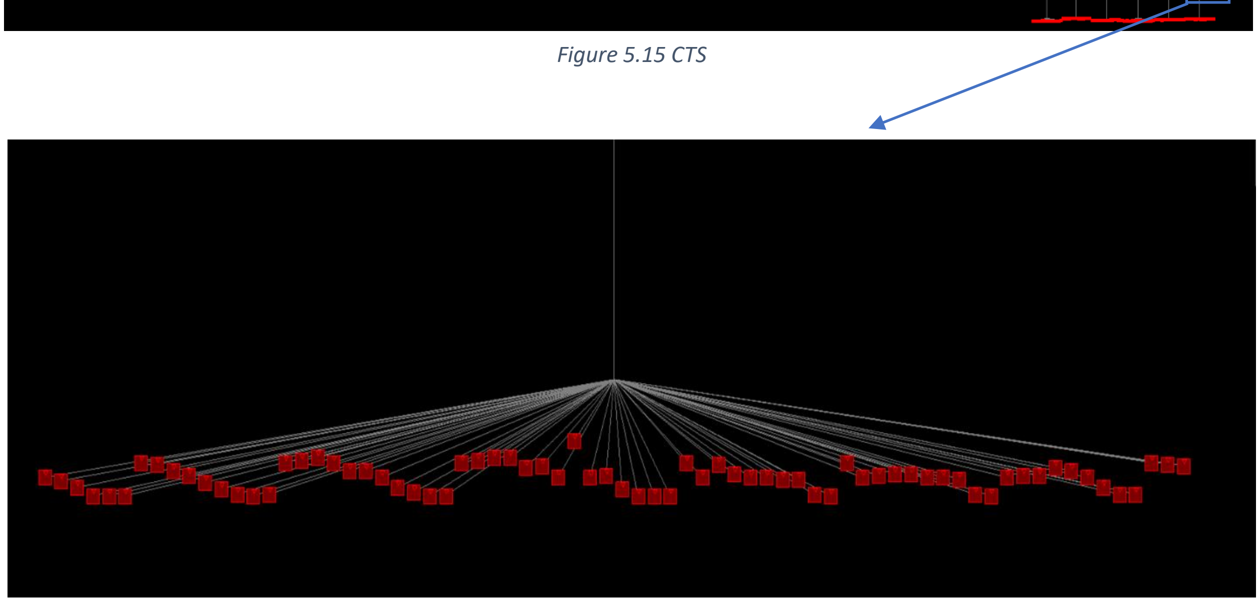


Figure 5.16 CTS Branch

5.4.5 Routing

Routing involves establishing physical connections between clock and signal pins using metal interconnects. This process is carried out while ensuring that the routed path satisfies various important constraints. These constraints include meeting setup and hold timing requirements, accommodating maximum capacitance and transients, and addressing clock skew requirements. Additionally, the metal traces must adhere to specific physical design rule check (DRC) requirements. By taking all these factors into consideration, the routing process ensures the successful creation of a functional and optimized layout for electronic circuits.

5.4.6 Chip Finish

Chip Finnish Stage mainly focus on:

- Addition of Filler cell
  - EndCap
  - Tie high/low
- Re-running checks (DRCs, Connectivity, Timing)
- Generate netlist files & Reports
- Generated GDSII file

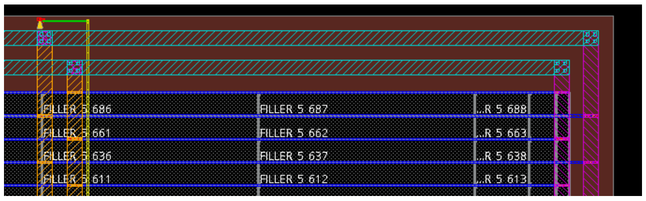


Figure 5.17 Filler Cell insertion

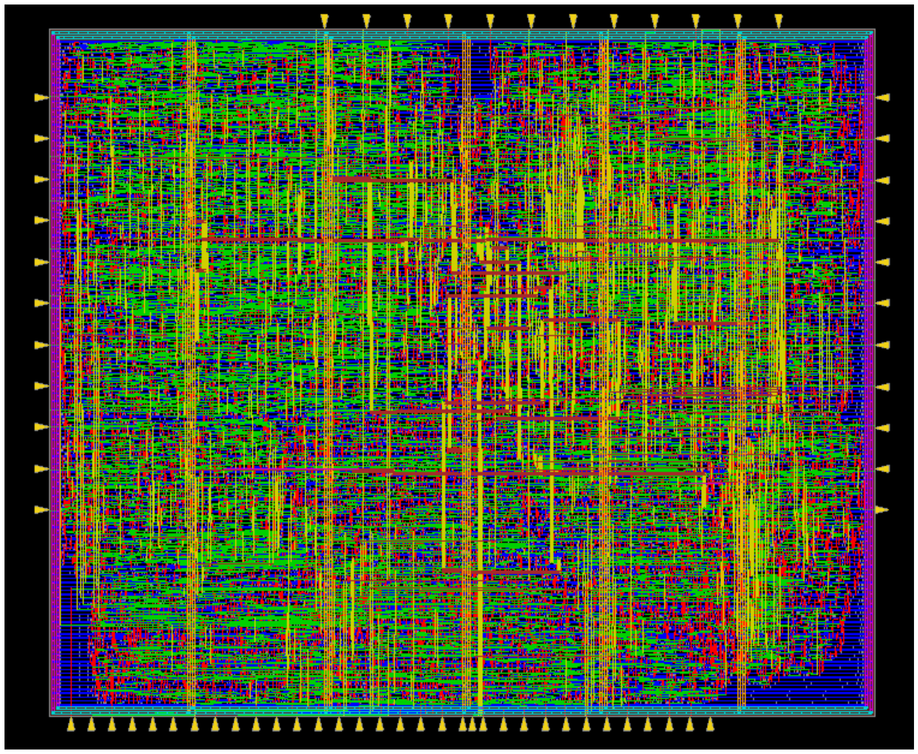


Figure 5.18 Full Chip

5.5 Results

Synthesis results reveal a total power consumption of 2.437 mW for the design. This metric provides an overall assessment of the power requirements of the design, considering all power sources within the circuit. Further analysis reveals that the internal power consumption is estimated to be 1.7594 mW. Internal power refers to the power dissipated within the circuit due to the switching activity of the internal nodes and the internal circuitry. Additionally, the switching power, which accounts for the power consumed during the transitions of the signals within the design, is measured to be 0.5439 mW. This power component is primarily influenced by the frequency of the clock signal and the capacitance of the load.

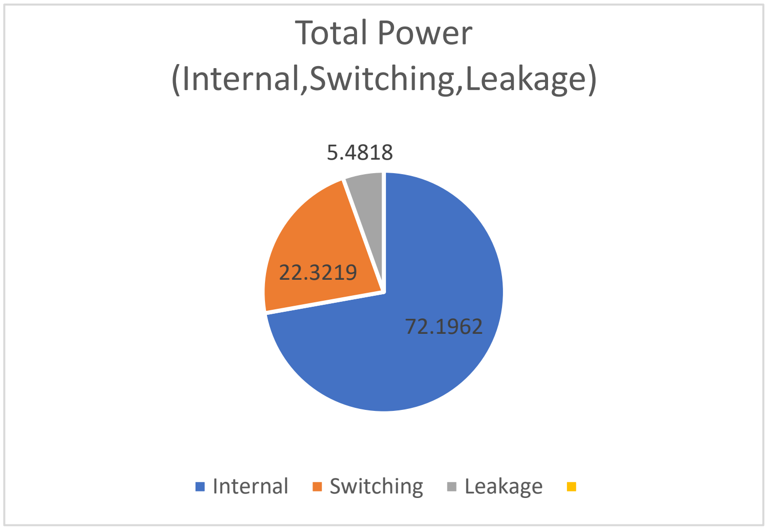


Figure 5.19 Power Chart

$$P_{switching} = \frac{1}{2}\alpha\, C_L\, V_{DD}^2\, f$$

Equation 5.1 Power Switching Equation

Furthermore, the cell leakage power, which represents the power consumed by the cells even when they are not actively switching, is reported to be 0.133 mW. Cell leakage power is associated with the inherent characteristics of the semiconductor devices and their leakage currents.

The PnR results indicate a total cell area of 175262 μm² for the design. This metric provides an estimation of the overall physical footprint occupied by the cells in the integrated circuit.

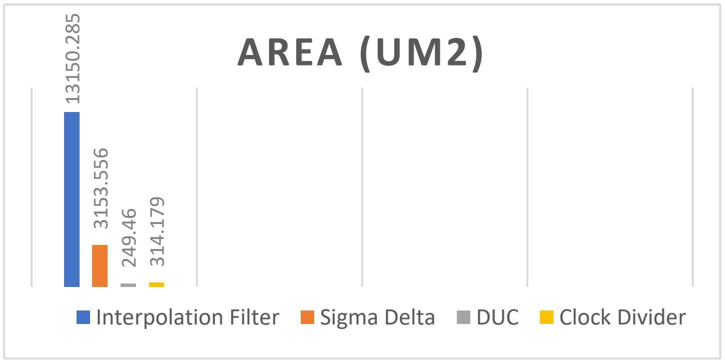


Figure 5.20 Area

In comparison to the Flynn paper, our work exhibits several differences, as summarized in the following table:

Parameter	This Work	Direct to RF Beamforming Tx
Technology	TSMC 130 nm CMOS	40 nm CMOS
Sampling Rate	400 MHz	1.2 GHz
Input Frequency	100 MHz	300 MHz
OSR	32	8
Bandwidth	6.25 MHz	75 MHz
Power (mW)	2.437	16
Area (mm^2)	0.1387	0.02

Table 5.8 Comparison with Flynn

Chapter 6: ASIC Flow Using OpenLane

6.1 Introduction to ASIC Flow Using OpenLane

This section provides an introduction to ASIC flow using OpenLANE, an open-source toolchain for digital ASIC design and implementation. It highlights the significance of utilizing open-source tools in the ASIC design process and explores the benefits and opportunities offered by OpenLANE. The section aims to familiarize readers with the fundamentals of OpenLANE, its role in streamlining the ASIC flow, and its impact on the accessibility and innovation within the field of ASIC design.

6.1.1 OpenLane Flow

In the realm of Application-Specific Integrated Circuit (ASIC) design, the use of open-source tools has gained significant traction, enabling chip designers to enhance productivity, flexibility, and customization. OpenLane, an open-source digital RTL to GDSII flow toolchain, has emerged as a powerful solution that automates the process of designing integrated circuits. This section provides an in-depth exploration of OpenLane, its key features, and the impact it has on the ASIC design landscape.

6.1.1.1 Overview of OpenLane

OpenLane is a complete RTL-to-GDSII implementation flow that encompasses all the necessary steps involved in designing a manufacturable chip. It integrates various open-source Electronic Design Automation (EDA) tools, allowing designers to efficiently create ASIC designs while providing the flexibility to customize the flow based on specific requirements. OpenLane has gained popularity for its ability to streamline the ASIC design process and its extensive use within the academic, research, and industrial communities.

6.1.1.2 Components and Capabilities

One of the notable advantages of OpenLane is its compatibility with open-source tools and technologies, providing designers with the flexibility to tailor the flow to their specific needs. OpenLane framework also allows for the integration of additional EDA tools, empowering users to further customize and extend the flow as required. This flexibility enables designers to optimize the ASIC design process for various application domains and explore innovative design approaches.

OpenLane is built around a set of open-source EDA tools that work together seamlessly.

- Yosys is utilized for RTL synthesis, enabling the conversion of Register Transfer Level (RTL) designs to a gate-level representation.
- OpenROAD is employed for physical synthesis and optimization, optimizing the design for power, performance, and area.

- Magic is used for layout design and performs Design Rule Checking (DRC) and Layout vs. Schematic (LVS) checks.

The integration of these tools within OpenLane ensures a comprehensive and efficient flow from RTL to GDSII shown in figure 6.2.

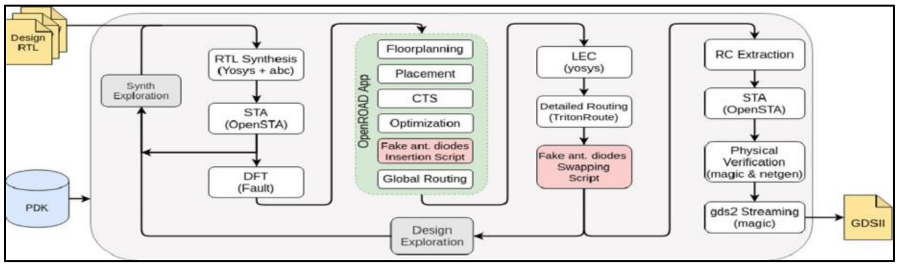


Figure 6.1 OpenLane Tools

6.1.1.3 OpenLane Development and Availability

OpenLane is developed and maintained by the Efabless Corporation, which ensures continuous updates, bug fixes, and improvements to the toolchain. It is available on GitHub under the Apache License 2.0, promoting collaboration and knowledge sharing within the ASIC design community.

Figure 6.2 Efabless Corporation

The Apache License 2.0 is an open-source software license widely used for the distribution and licensing of software projects. It is recognized by the Open-Source Initiative (OSI) and provides users with the freedom to use, modify, distribute, and sublicense the software under certain conditions.

Figure 6.3 The Apache Foundation

This open-source nature encourages designers to contribute enhancements, share their experiences, and collectively advance the field of ASIC design.

6.2 Logic Synthesis Using Yosys

6.2.1 Synthesis Overview

OpenLane utilizes Yosys, an open-source RTL synthesis tool, to convert the RTL description of the design into a gate-level representation. Yosys optimizes the design for area, power, and timing, preparing it for subsequent stages.

To facilitate the design process, it is essential to have access to a Process Design Kit (PDK) that provides the necessary information and guidelines for designing ICs with a specific semiconductor manufacturing process. In the context of open-source design, the Skywater-130nm PDK is an example of an open-source PDK that enables designers to create ICs using the 130nm technology node.

Figure 6.4 Skywater Technology

The Skywater-130nm PDK, being an open-source PDK, provides designers with access to the necessary process information, layout rules, device models, and other design-specific data required to implement their designs using the 130nm technology. This open-source approach empowers designers to leverage the capabilities of the Skywater-130nm process, customize their designs, and contribute to the development and improvement of the PDK itself.

Following the synthesis of the netlist using Yosys with the Skywater 130 nm PDK, a log file analysis was performed to detect critical paths within the design. The analysis facilitated the identification of potential timing violations, as well as opportunities to optimize slack, area, and power consumption.

Through the implementation of various techniques such as logic restructuring and pipelining, timing performance was improved and power consumption was reduced, while the overall design layout was optimized. This iterative optimization process continued until all design specifications were satisfied.

Standard cells in the technology library are characterized based on specific operating conditions, referred to as Process, Voltage, and Temperature (PVT) variations. During the synthesis and timing analysis stages, it is crucial to take into account these PVT variations, as they have a significant impact on the performance and functionality of the designed integrated circuits. Refer to the accompanying figure for a visual representation of these variations.

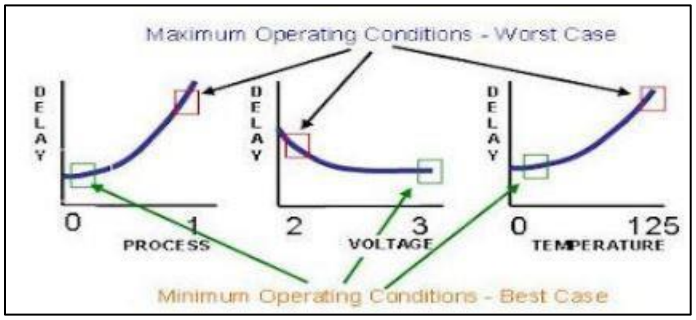

Maximum Operating Conditions - Worst Case
DELAY
DELAY
DELAY
0
1
PROCESS
2
3
VOLTAGE
0
125
TEMPERATURE
Minimum Operating Conditions - Best Case

Figure 6.5 Process, Voltage and Temperature Variations

Two essential libraries used in the design process are the worst-case library (SS) and the best-case library (FF). The worst-case library (SS) “sky130_fd_sc_hd__ss_100C_1v60.lib” contains cells characterized for maximum delay, representing scenarios with the slowest possible operation due to process, voltage, and temperature variations. Conversely, the best-case library (FF)” sky130_fd_sc_hd__ff_n40C_1v95.lib” includes cells characterized for minimum delay, capturing conditions that result in the fastest possible operation.

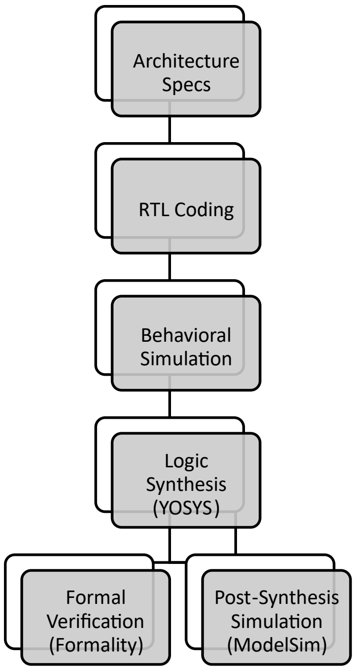


Figure 6.6 OpenLane ASIC Flow

6.2.2 Synthesis Flow

During the initial phase of the design process, YOSYS is employed to read and analyze the designs, ensuring that all syntax is properly synthesized and establishing the necessary connections between the different design components.
In the specific design under consideration, an asynchronous reset is implemented alongside a main clock signal. This main clock signal is divided into four additional generated clocks: clock/4, clock/16, clock/32, and clock/128. These generated clocks enable the synchronization and timing requirements of the design.

To ensure consistent and reliable input signal propagation, a driving cell named "sky130_fd_sc_hd__inv_2" is assigned to all input ports. This driving cell governs the input characteristics and influences the behavior of the input signals.

To account for timing considerations, an input delay of 20% of the main clock period is applied to all input ports. This delay helps synchronize the input signals with the clock domain and enables proper timing analysis.

The output load at all output ports is set to a specific value of 33.442 pF. This value represents the expected load conditions that the design must be able to drive effectively.

In terms of clock considerations, a Clock Transition value of 0.15 ns is defined. This value represents the desired percentage of clock transition time within the design.

Furthermore, a Clock Uncertainty value of 0.25 ns is specified to account for any uncertainty or variation in the arrival time of the clock signal. This uncertainty factor ensures that the design can accommodate potential variations in the clock timing.

To estimate the wire delays accurately, a wire-load model from the library is employed. This model provides an approximate value for calculating the delay introduced by the wires within the design.

Input Delay	20%
Output Load	33.442 pF
Clock Transition	0.15 ns
Clock Uncertainty	0.25 ns
Input Driving Cell	sky130_fd_sc_hd__inv_2

Table 6.1 Synthesis Constrains

6.2.3 Synthesis Results

This section presents an in-depth analysis of the synthesis results obtained for the design under consideration. It explores the impact of synthesis on various design metrics, including area utilization, power consumption, and critical path timing. By examining these results, valuable insights can be gained regarding the efficiency and effectiveness of the synthesis process in achieving the design goals.
After a successful compilation of the design, we have achieved a clock frequency of 400 MHz without any violations. This accomplishment signifies that the design meets the timing requirements and can operate reliably at this frequency

6.2.3.1 Area analysis

The synthesis results indicate a total cell area of 165965 µm² for the design. This metric provides an estimation of the overall physical footprint occupied by the cells in the integrated circuit.

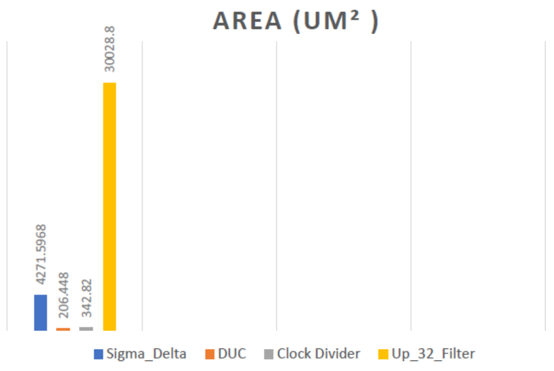


Figure 6.7 Area Analysis

6.3 Physical Synthesis

6.3.1 Overview

Physical synthesis is a crucial component of modern VLSI design methodologies for ASICs, as it plays a vital role in achieving design closure.

The following figure shows the physical synthesis flow in OpenRoad

The physical synthesis flow in OpenRoad begins with the definition of required Physical Design Kits (PDKs) Libraries, including:

- .Techlef (sky130_fd_sc_hd_nom.tlef)
- .Lef files (sky130_fd_sc_hd.lef)

These libraries are integrated with Logical Libraries (e.g., .Lib files).

The subsequent steps involve providing the design netlist and constraints, performing technology mapping and logic optimization, and finalizing the placement of circuit components within the chip area.

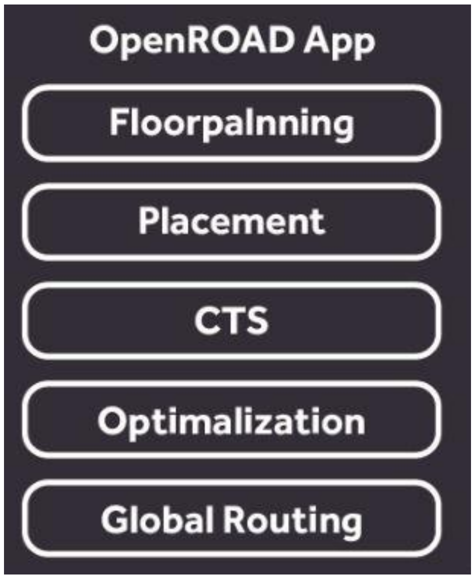


Figure 6.8 Physical Synthesis Flow

This flow enables efficient conversion of a logical design into its physical implementation, considering various design considerations.

6.3.2 Floorplan

The goal of Floorplan is to efficiently allocate and position the functional blocks of a chip on the silicon area while considering various design constraints.

In floorplan, four main procedures occur: initialization, I/O Placement, tap cell placement, and power planning.

1. Initialization: This procedure focuses on determining the die area and core area for the chip design.
   Die Area is 600 * 500 μm^2
   Core Area is 588.955 * 478.24 μm^2

2. I/O Placement
   Specify metal layer 2 for left and right column.
   Specify metal layer 3 for up and down row.

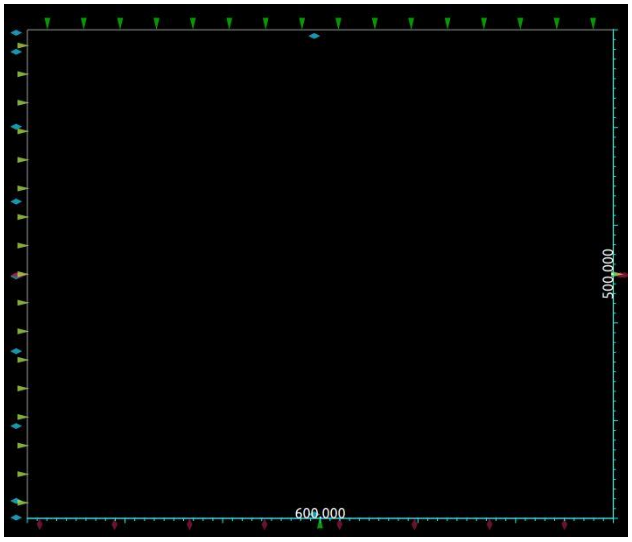


Figure 6.9 Chip after Initialization

3. Placement: In this step, decap cells and well tap cells are placed within the chip layout.
    - Decap Sells are basically a charge storing device made of capacitor works as charge reservoirs and support the power delivery network and make it robust
    - Well-Tap Cells are used to prevent latch-up issue in CMOS design by connecting the nwell to VDD and p-substrate to VSS.

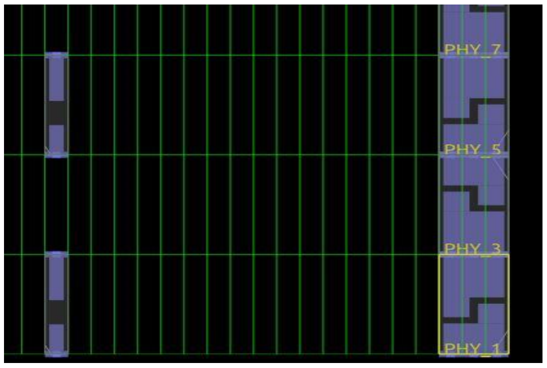


Figure 6.10 Decap and Tap Cells

4. Power Planning: The final procedure focuses on determining the required power ring, as well as the number of stripes and rails for effective power distribution.

    Power Ring on Layer 5 & Layer 4:

    Place power rings on metal layer 7 and metal layer 8 to provide power supply to the circuitry.

    Stripes on Metal Layer 4:

    Implement power stripes on metal layer 6 to enhance power distribution and reduce voltage drop across the chip.

    Power Rails:

    Establish a power Rails network throughout the chip, connecting the power rings, stripes, and other power distribution structures.

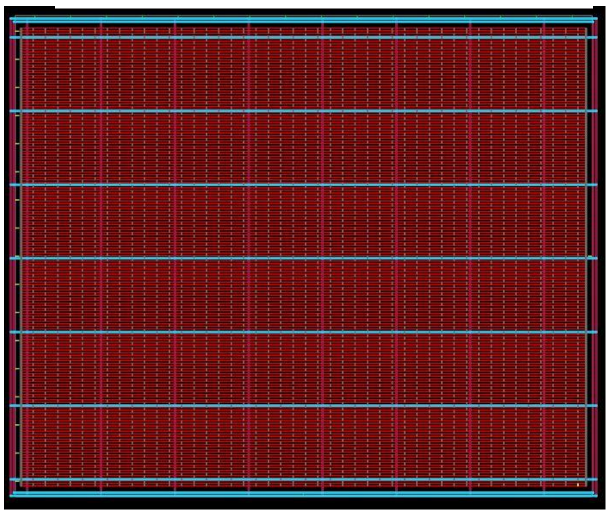


Figure 6.11 Power Ring

6.3.3 Placement

Placement is a pivotal stage in the design flow where each instance is assigned a precise location. The process involves the following steps:

Readiness Check: Ensuring the readiness of floorplan, netlist, and design constraints before commencing the placement.

Cell Placement and Reporting: Assigning specific positions to all cells and generating a report that includes details of the placed cells and available legal sites for further cell placement.

Tie Cell Insertion: Examining the entire design to identify inputs connected to VDD or VSS. Tie cells are inserted at these points to mitigate potential errors caused by IR drop.

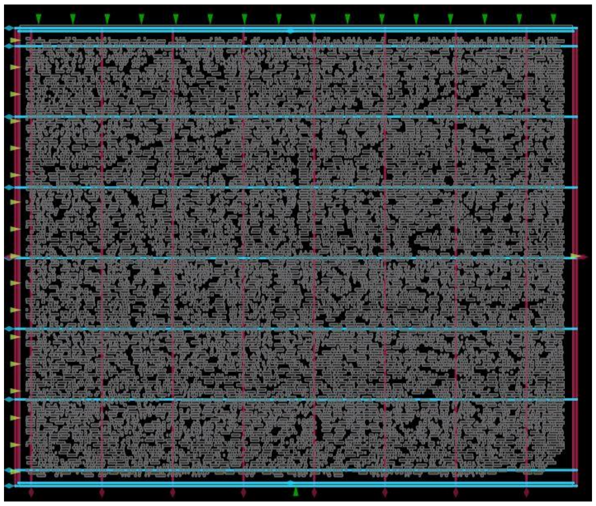


Figure 6.12 Cells After Placement

After placing cells, the timing constraints are checked to ensure that there are no setup violations nor hold Violation.

Worst Slack -max (Setup)	0.60
Worst Slack -min (Hold)	0.19

Table 6.2 Timing after Placement

6.3.4 Clock Tree Synthesis

Two issues have been identified in the design:

Timing Skew: The timing skew between adjacent cells is caused by factors such as propagation delay and other parameters. This leads to imbalances in signal arrival times, affecting overall circuit performance.

High Fanout Cells: Certain cells with high fanout are connected to the clock pin, requiring a buffer driver chain to adequately drive all these cells. Without proper buffering, the clock signal may experience degradation or fail to reach the high fanout cells effectively.

To address these problems, the introduction of a Clock Tree is necessary. The Clock Tree synthesis process involves designing a well-structured network of buffers and wires to distribute the clock signal evenly and efficiently across the chip.

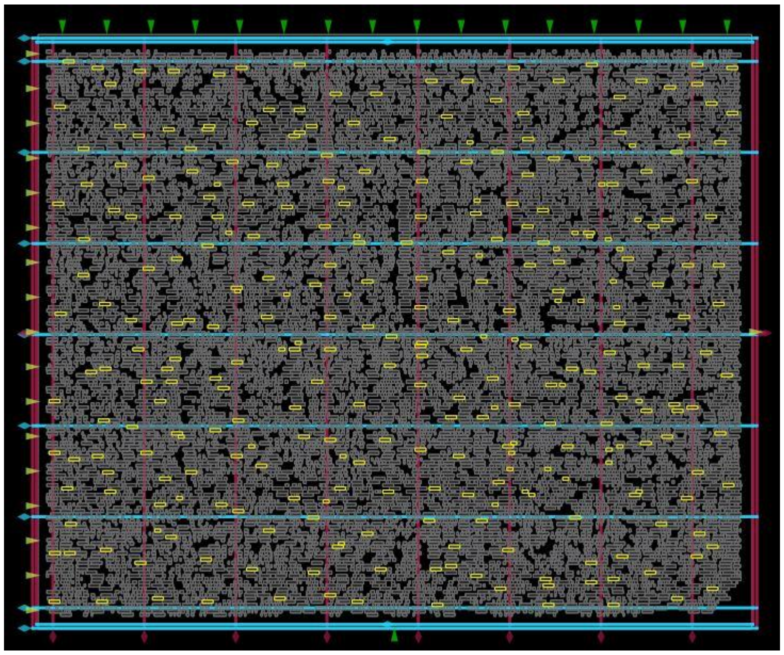


Figure 6.13 Chip after CTS

By implementing a Clock Tree, both timing skew and the driving capability of high fanout cells can be effectively resolved, improving the overall performance and reliability of the design.

Driving cell used is “sky130_fd_sc_hd_clkbuf_16”, we use that instead of normal buffer or inverter because it has almost 50% duty cycle and which will solve the problem of minimum pulse width this shown in the following figure where the highlighted cells are the Driving cell buffer which uniformly distributed across the core

Worst Slack -max (Setup)	0.63
Worst Slack -min (Hold)	0.07

Table 6.3 Timing After CTS

6.3.5 Routing

The routing stage is responsible for establishing physical connections between clock and signal pins using metal interconnections. The routed paths must satisfy various requirements, including:

- Timing Constraints: The routed paths must meet setup and hold timing requirements to ensure proper signal capture and retention.
- Maximum Capacitance/Transitions: The routing must consider the maximum capacitance and transitions on the paths to ensure signal integrity and minimize delays.
- Clock Skew: Clock signals need to be distributed with minimal skew, ensuring synchronized clocking across the design.

Worst Slack -max (Setup)	0.63
Worst Slack -min (Hold)	0.07

Table 6.4 Timing After Routing

6.3.6 Chip Finish

Chip Finnish Stage mainly focus on:

- Addition of Filler cell
    - EndCap
    - Tie high/low
- Re-running checks (DRCs, Connectivity, Timing)
- Generate netlist files & Reports
- Generated GDSII file

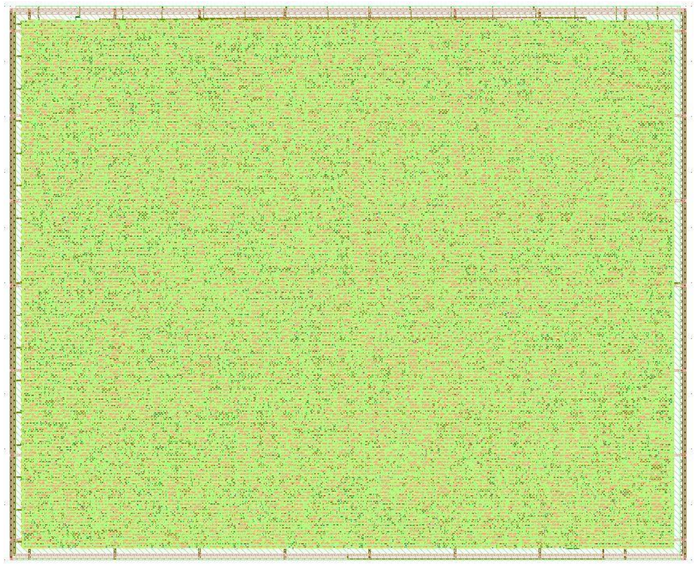


Figure 6.14 GDSII

6.4 Results

After each stage of the design process, reports are generated to provide information about the area utilization and power consumption, enabling designers to assess the impact of the changes made in each stage on these key metrics.

Stage	Area (µm^2)	Power (mW)
Placement	141831	5.01
CTS	148923	5.24
Routing	148963	5.40
Chip Finishing	148863	5.40

Figure 6.15 Area and Power Reports

Placement density after routing is reported as a metric that quantifies the efficient utilization of the chip area. It provides insights into the level of congestion and packing density achieved during the routing process. By analyzing the placement density, designers can evaluate the effectiveness of the routing phase in optimizing the placement of components and interconnections within the available chip area.

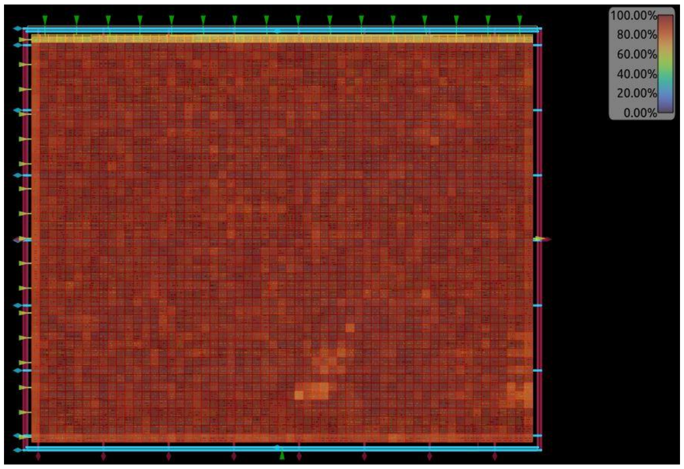


Figure 6.16 Placement Density after Routing

Power density after routing refers to the measure of power dissipation per unit area in the chip layout after the routing process. It provides information about the power distribution and concentration in different regions of the design. By reporting power density, designers can assess areas of potential high power consumption and identify any localized power hotspots that may require mitigation strategies to ensure proper thermal management and reliable circuit operation.

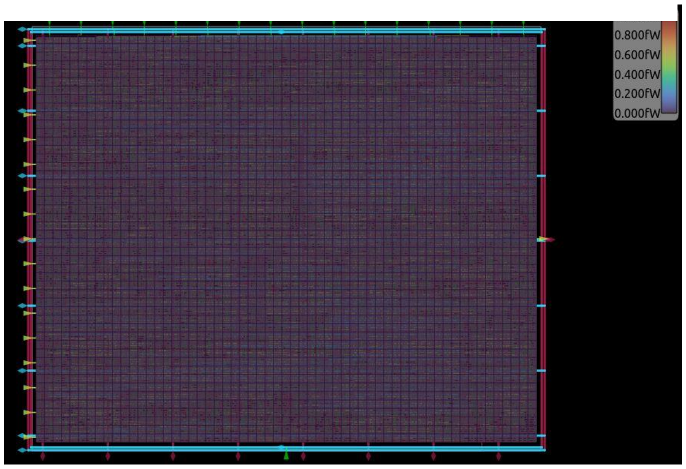


Figure 6.17 Power Density after Routing

In comparison to the Flynn paper, our work exhibits several differences, as summarized in the following table:

Parameter	**This Work**	**Direct to RF Beamforming Tx**
Technology	Skywater 130 nm CMOS	40 nm CMOS
Sampling Rate	400 MHz	1.2 GHz
Input Frequency	100 MHz	300 MHz
OSR	32	8
Bandwidth	6.25 MHz	75 MHz
Power (mW)	5.40	16
Area (mm^2)	0.1488	0.02

Table 6.5 Comparison with FLynn

Conclusion

The transmitter architecture discussed throughout the thesis showed its ability in steering and controlling the directivity accurately to achieve the 5G transmitting characteristics depending on the digital beamforming MIMO technique. The output signal is one-bit signal which is suitable to interface with the one-bit DAC. Table 6.5 shows the performance of this work comparing to Flynn's work.

Parameter	**This Work**		**Direct to RF Beamforming Tx**
Technology	TSMC 130 nm CMOS	Skywater 130 nm CMOS	40 nm CMOS
Sampling Rate	400 MHz	400 MHz	1.2 GHz
Input Frequency	100 MHz	100 MHz	300 MHz
OSR	32	32	8
Bandwidth	6.25 MHz	6.25 MHz	75 MHz
Power (mW)	2.245	5.40	16
Area (mm^2)	0.1387	0.1488	0.02

Table 7.1 Comparison with Flynn's Results

Future Work

After discussing this work in details throughout the thesis, this section would discuss the future work for the transmitter showing what can be done next. Some of improvements that can be done:

- Complete ASIC Implementation of 8 chains beamforming transmitter.
- Implement the sigma delta block with pipelining technique which would increase system speed and bandwidth.
- Perform verification phase for the system.
- Get The system error vector magnitude (EVM) for the output signal to provide more tests for the system accuracy.

References

[1] Gutierrez, Carlos & Caicedo, Mauricio & Campos Delgado, Daniel Ulises, “5G and Beyond: Past, Present and Future of the Mobile Communications,” IEEE Latin America Transactions, 2021.

[2] Boyi Zheng, Lu Jie, John Bell, Yan He and Michael P. Flynn, “A Two-Beam Eight-Element Direct Digital Beamforming RF Modulator in 40-nm CMOS,” IEEE, 2019.

[3] Christian Wolff, "Phased array antenna," Radar Tutorial, web. archive. org, radartutorial. eu, 2014.

[4] Hassan Aboushady, Y. Dumonteix and M. Loueart, "Efficient polyphase decomposition of comb decimation filters in ΣΔ analog-to-digital converters," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 48.10, 2001.

[5] Arar, Steve, "The Polyphase Implementation of Interpolation Filters in Digital Signal Processing," Circuit Cellar, no. 320, 2017.

[6] F. J. Taylor, “Digital Filters Principles and Applications with MATLAB,” IEEE Press, 2012.

[7] Richard Schreier and Gabor C. Temes, “Understanding Delta-Sigma Data Converters,” Oregon State University, 2005.

[8] Hugo Cheung and Sreeja Raj, “Implementation of 12-bit delta-sigma DAC with MSC12xx controller,” MSC Group, 2005.

[9] Sohail Imran Saeed, Khalid Mahmood and Mehr e Munir, “Design of an Error Output Feedback Digital Delta Sigma Modulator with In–Stage Dithering for Spur–Free Output Spectrum,” IJACSA, 2018.

[10] Lyons, R. G., “Understanding digital signal processing (3rd ed.),” Prentice Hall, 2010.

[11] Synopsis Formality User Guide Version P-2019.03, March 2019.

[12] Ghazy, Ahmed Alaa and Mohamed Shalan, “OpenLANE: The Open-Source Digital ASIC Implementation Flow,” (2020).