A Massive MIMO mm-Wave Receiver For 5G

A Thesis

Submitted to

THE DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS

FACULTY OF ENGINEERING, CAIRO UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF BACHELOR OF SCIENCE IN ELECTRONICS AND

ELECTRICAL COMMUNICATIONS ENGINEERING

ASMAA SAMEH ABUSRIEA

BAHY YEHIA YOUSSEF

MANAR MOUSTAFA ASHOUB

WAFAA ALGOHARY MOUSSA

ZIAD TAMER SAYED

ZIAD SAEED LOTFY

UNDER SUPERVISION OF

DR. HASSAN MOSTAFA

DR. HASSAN ABU SHADY

TECHNICALLY SPONSORED BY

SEAMLESS WAVES

ONE LAB EGYPT

JULY 2022

A Massive MIMO mm-Wave Receiver For 5G

A Thesis

Submitted to

THE DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS

FACULTY OF ENGINEERING, CAIRO UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF BACHELOR OF SCIENCE IN ELECTRONICS AND

ELECTRICAL COMMUNICATIONS ENGINEERING

ASMAA SAMEH ABUSRIEA

BAHY YEHIA YOUSSEF

MANAR MOUSTAFA ASHOUB

WAFAA ALGOHARY MOUSSA

ZIAD TAMER SAYED

ZIAD SAEED LOTFY

UNDER SUPERVISION OF

DR. HASSAN MOSTAFA

DR. HASSAN ABU SHADY

TECHNICALLY SPONSORED BY

SEAMLESS WAVES

ONE LAB EGYPT

JULY 2022

Table of Contents

List of Tables:

List of Figures:

List of Symbols and Abbreviations:

Abbreviation	Definition
ABF	Analog Beamforming
ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
BSP	Bit-Stream Processing
BW	Band width
CLK	Clock
CMOS	Complementary Metal Oxide Semiconductor
CTBPDSMs	Continuous Time Bandpass $\Delta\Sigma$ Modulators
CWM	Complex Weighted Multiplier
DBF	Digital Beamforming
DDC	Digital Down Conversion
DR	Dynamic Range
DSP	Digital Signal Processing
Fig	Figure
FIR	Finite Impulse Response
HB	Half-band
IC	Integrated Circuit
IF	Intermediate Frequency
LO	Local Oscillator
MIMO	Multiple Input Multiple Output
MUX	Multiplexer
OSR	Over Sampling Ratio
PNR	Place And Route
RF	Radio Frequency
RTL	Register Transfer Level
RX	Receiver
SNR	Signal to Noise Ratio
UVM	Universal Verification Methodology
VM	Virtual Machine

Acknowledgments:

Without the participation and assistance of so many people the completion of this undertaking could not have been possible. Their contributions are sincerely appreciated and gratefully acknowledged. However, the group would like to express their deep indebtedness and appreciation to Dr. Hassan Mostafa, Dr. Hassan Abu Shady for their endless support, kind and understanding spirit. We also would like to thank Eng. Abdelrahman Emad, Eng. Ahmed Ghoniem and Eng. Marco Seif for their help and contributions and for helping us understanding everything.

Abstract:

In this work, we introduce an architecture of a digital receiver that is based on combination of continuous time bandpass ΔΣ modulators (CTBPDSMs) and bit-stream processing (BSP). This architecture gives us area and power efficient digital beamforming (DBF). In this work we deal with 4 elements of 325 MHz IF and sampling rate of 1.3 GHz. The used technology is 45 nm CMOS. The architecture performs the digital down conversion and the phase shifting by using only multiplexers. By combining the inputs of 4 elements we ger SNR around 77db over 20 MHz BW. Our design occupies 29343.79 μm^2 and consumes power by 9744 μW.

Chapter 1: Introduction

1.1 5G and MIMO

Emerging wireless systems such as 5G will use mm-wave bands to support more users and deliver higher data rates. However, mm-wave systems face severe link-budget challenges due to high path losses and the lower diffraction of mm-wave signals. Large-array beamforming is an essential technology to make up for the reduced link budgets with array gain and to establish reliable wireless connections through the use of MIMO algorithms. [1]

MIMO has been used in wireless communication to control how the data maps into antennas and where to focus energy on space. As the technology evolved and now, we have designs based on 5G, the Massive MIMO become established. Massive MIMO (which is an extension of MIMO) is based on adding higher number of antennas on the base station. It is important for 5G wireless technology, as it provides increasing in the throughput and the signal to noise ratio. The reason behind this is that by increasing the number of the elements, the main lobe become narrower, and the side lobes have less power.

1.2 Beamforming Motivation

Beamforming is a type of radio frequency (RF) management in which a wireless signal is directed toward a specific receiving device. It is an array processing technology used in the MIMO to focus energy along a specific direction. It filters the incoming signals spatially (it avoids interference from different locations). Beamforming removal of inference is very useful in the case of the interferer frequency is close to the desired signal frequency (in this case frequency filtering is not very useful). It improves SNR (signal to noise) ratio of RX signal by 3 DB for each doubling the number of antenna elements.

Applications of Beamforming:

- It is used in military systems to suppress jamming signals.
- Radar, Sonar, Astronomy, Acoustics & Wireless Communications.

1.3 Beamforming Receiver Architectures

In the beamforming receiver, the phase and amplitude of each antenna element are adjusted to create beams and to steer nulls. The phase (θ) and amplitude (A) in antenna path can be represented as a complex weight (A$e^{j\theta}$). In the far field, the received amplitude in each antenna element is approximately the same and therefore only phase adjustment is sufficient.

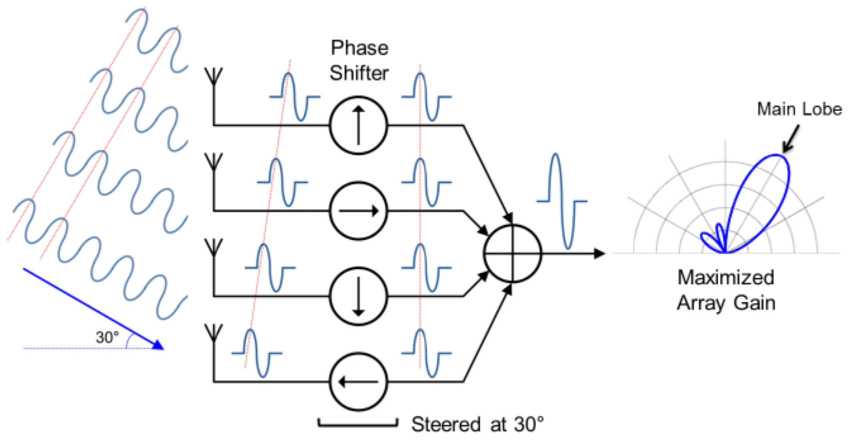


Figure 1 Constructive Interference at angle 30

Fig (1) shows a plane wave with an incidence angle of 30° is received by a 4-element linear antenna array with $\frac{\lambda_{RF}}{2}$ spacing (phase difference =90°) between each two adjacent signals.

The phase difference is compensated by phase shifters, resulting in coherent signals at the outputs of phase shifters (maximizing the array gain for incident angle of 30°). The array gain is 12 dB.

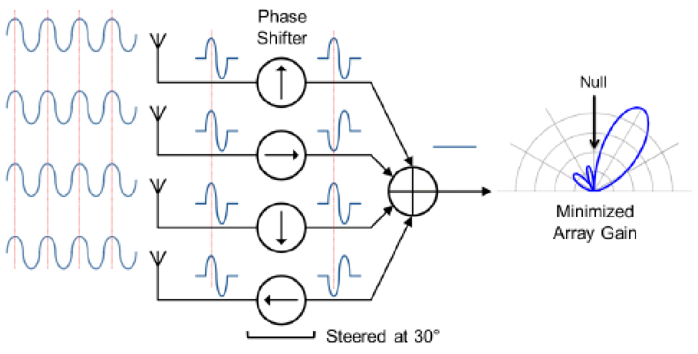


Figure 2 Destructive Interference at angle 0

Fig (2) shows when the incidence angle is equal to 0. In that case, signals at the outputs of phase shifters are out of phase. Therefore, they are destructively combined (resulting a null).

1.3.1 Antenna parameters:

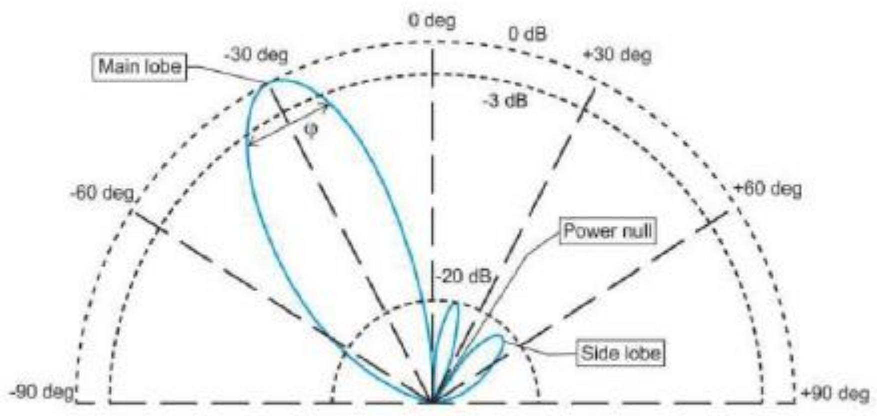


Figure 3 Antenna Parameters

Beam pattern: The plot in which the array gains for different incidence angles (usually plotted in polar diagram).

Main lobe: The lobe which contains the max power, and the other lobes are called as side lobes.

Beam width (∅): the angle between half-power (-3dB) points in the main lobe.

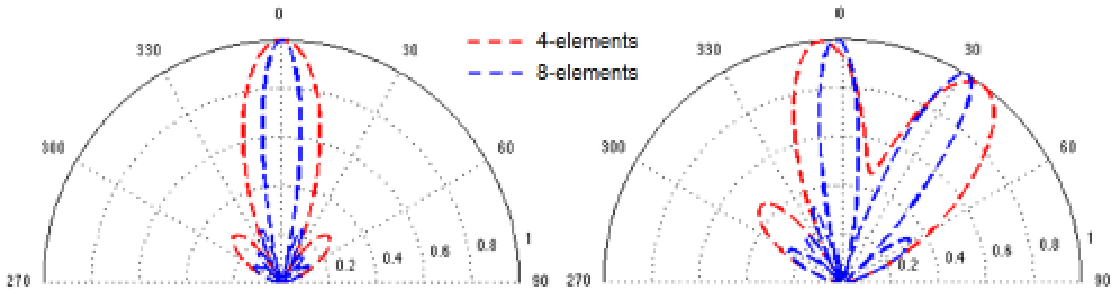


Figure 4 Beam patterns of four and eight-element antenna arrays

As shown in fig (4), as the number of antenna elements increases, the beam width decreases, and the side lobes become smaller.

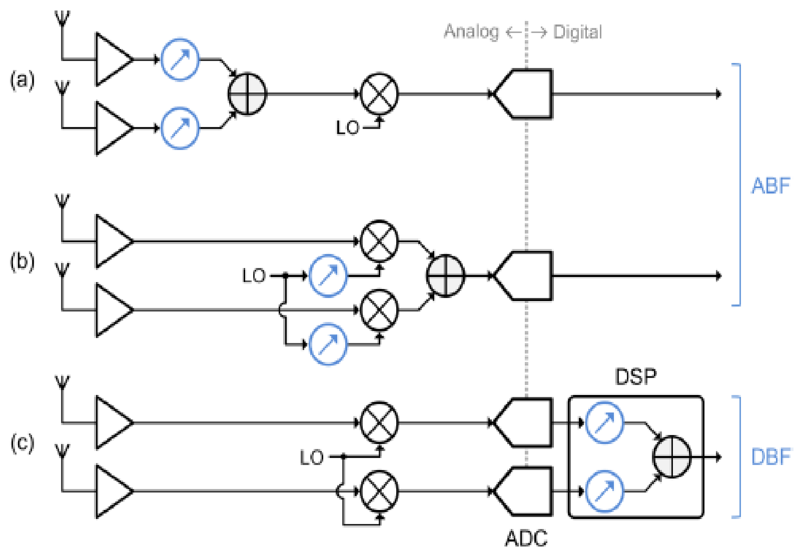


Figure 5 (a) ABF in the RF signal path. (b) ABF in the LO path. (c) DBF [2]

As shown in fig (5) we have 3 different receivers that we will explain each of them.

1.3.2 Analog Beamforming

In fig (5) graph (a), we have an implementation of an ABF in the RF signal path. In this receiver the phase shifters can be implemented in the RF signal path. In this way, multiple signal paths are combined at the very early stage of RX and by this the number of subsequent Hardware including down converters and ADCs (analog to digital converter) can be minimized.

It has many advantages: It relaxes the linearity & DR (dynamic range) requirements of down converters and ADCs. But it suffers from high insertion loss, limited phase-shift resolution, and component mismatch, which result in the degradation of system performance as well as losing of the information due to early combination. All of this causes limit for flexibility and the ability to form multiple simultaneous beams. [2]

In fig (5) graph (b), we have another implementation of an ABF receiver in the LO (local oscillator) path. In the LO-path beamforming, phase shifting is implemented in LO-distribution network, LO-path beamforming has less impact on SNR (signal to noise ratio).

From the disadvantages of LO-path beamforming has less impact on SNR:

- It requires multiple analog mixers and a large LO distribution network.
- It increases system complexity area. [2]

1.3.3 Digital Beamforming

In fig (5) graph (c), we have implementation of a DBF. This technique is known as complex weight multiplication (CWM) because here we express the signals in a complex form similar to be multiplied by $e^{j\theta}$. In the DBF the received signals are down converted to base band I/Q signals then digitalized by ADC's.

DBF achieves the highest accuracy and flexibility. In addition, multiple simultaneous beams can be formed. Moreover, DSP algorithms can be easily applied in DBF for advanced functions including adaptive beamforming and array calibration. However, DBF requires multiple down converters, high-performance ADCs, and an intensive DSP unit, resulting in high power consumption and large die area.[2]

That's why for efficient implementation of DBF, it was proposed a new DBF architecture based on CTBPDSMs and BSP. This is the used implementation in our work. In this receiver the digital down conversion (DDC) and phase shifting are implemented with only multiplexers (MUXs). Moreover, directly processing the CTBPDSM outputs avoids the need for multiple decimators for DBF. As a result, the architecture achieves power- and area-efficient compared to normal DBF.

1.4 Thesis Overview

In this thesis we represent a 4-element, 325 MHz IF, 1.3 GHz sampling rate, 20 MHz BW digital beamforming receiver. This receiver is based on the combination of CTBPDSM with BSP. In this work we address the problem of minimum pulse width and how to meet the timing constraints. Our target is to increase the used IF frequency without violating the timing constraints.

This paper is organized as follows. Chapter 2 provides the system modeling of the work and explanation to the ADC sigma modulator. It also provides the mathematical analysis. Chapter 3 provides the system implementation and results of the SNR. Chapter 4 provides the RTL description of the model. Chapter 5 and 6 provides the use of the VM tool to meet timing constraints and possible ways to increase the frequency. It also provides the total area, power, and layout of the system. Chapter 7 verifies that during the development of the design the function is still done correct, and no differences occur. Finally, there is conclusion for the thesis and suggestions for our future work.

Chapter 2 System Analysis

2.1. Sigma-Delta ADC

In ΔΣ modulation, the combination of oversampling and noise shaping enables a high SNR modulator output with a low-resolution quantizer. The ADC area and power consumption have a huge bearing on the die area and power consumption of the entire beamformer, small-area, and low-power consumption are critical as DBF uses a large number of ADCs. We choose a CTBPSDM for its energy efficiency and small size. Furthermore, a CTBPSDM is easy to drive, resilient to aliasing, and very attractive for digitizing IF signals. Band-pass digitization also halves the number of required ADCs because the quadrature signals share an ADC. On the other hand, two low-pass modulators would be required to digitize I and Q analog signals [7].

The concept of direct IF (or RF) sampling has arisen to enable digital-intensive receivers. By digitizing higher frequencies (i.e., IF or RF), most of the signal processing chain including down-conversion and filtering is carried out in the digital domain. This enables perfectly matched digital I/Q down conversion as well as high-performance channel selection filtering [1].

2.1.1 Supplied Sigma Delta

CTBPSDM MATLAB model with output as shown in the figure 6 was supplied by our main sponsor Seamless Waves with the following features:

1- Sixth order bandpass ADC with IF frequency equals to quarter of the Sampling Rate (F_s).
2- One Bit output bit stream.
3- Output Average SNR equals 68.65 dB.
4- Over Sampling Ratio (OSR) equals 32.

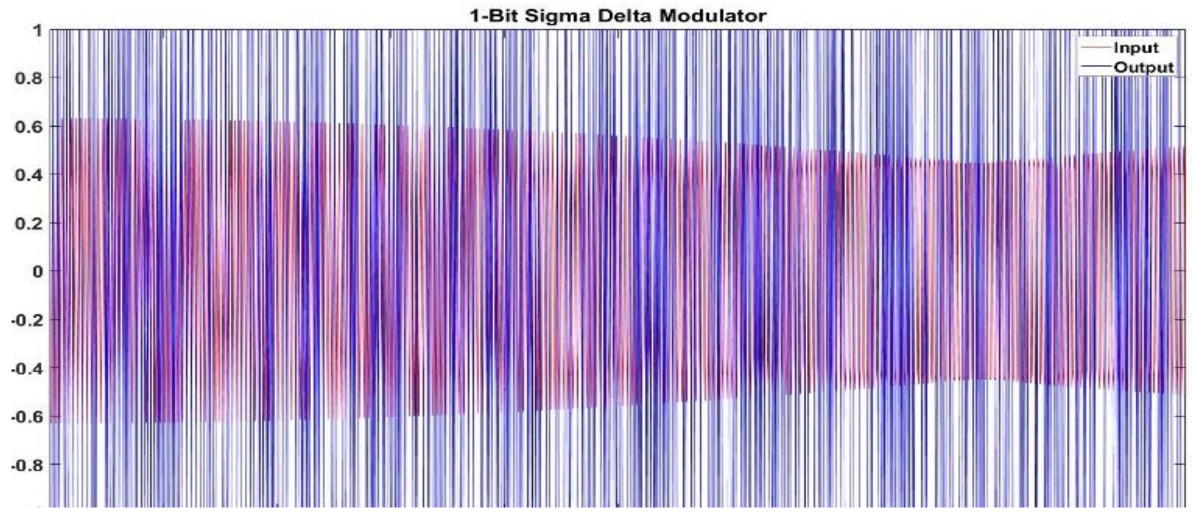


Figure 6 Sigma-Delta Modulator Input-Output

2.2. Decimation

Sampling rate conversion systems are used to change the sampling rate of a signal. The process of decreasing the sampling rate is called decimation, the down-sampler is the element that changes the sampling rate of the signal.

The drawback of the down-sampling is the aliasing effect, decimation must be performed in such a way as to avoid the effects of aliasing.

2.2.1 Theory of Decimation

Decimation filter is a must after the oversampled sigma-delta ADC output to be ready for further processing, it is done on two steps which are anti-aliasing then down sampling as shown in figure 7.

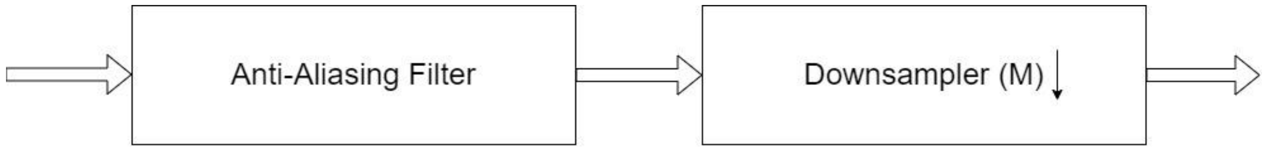


Figure 7 Decimation Filter Stages

Consider down sampling a signal with sampling rate T_1 by the ratio (M) to the output sampling rate T_2 as shown in figure 8, then the relation of the down sampler as shown equation (1):

$$y(kT_2) = x(kMT_1) \tag{1}$$

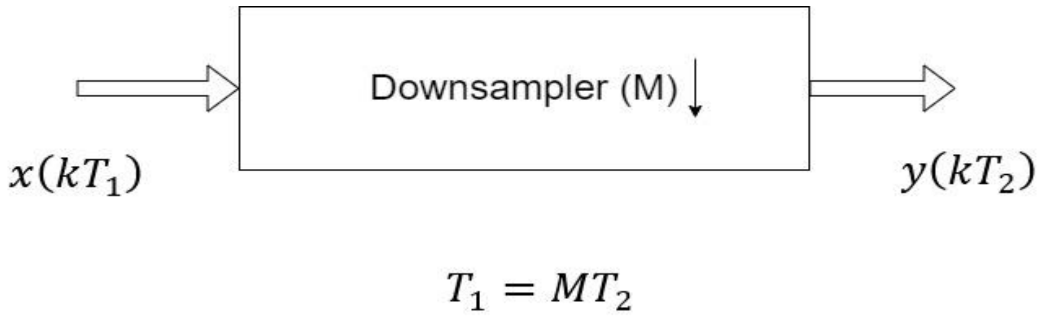


Figure 8 Down sampler

Signal in time domain is down sampled by ratio M through taking the M^{th} sample and leaving samples in between as shown in figure 9 while the signal in frequency domain after down sampling will be repeated every $\frac{F_S}{M}$ as shown in the figure 10.

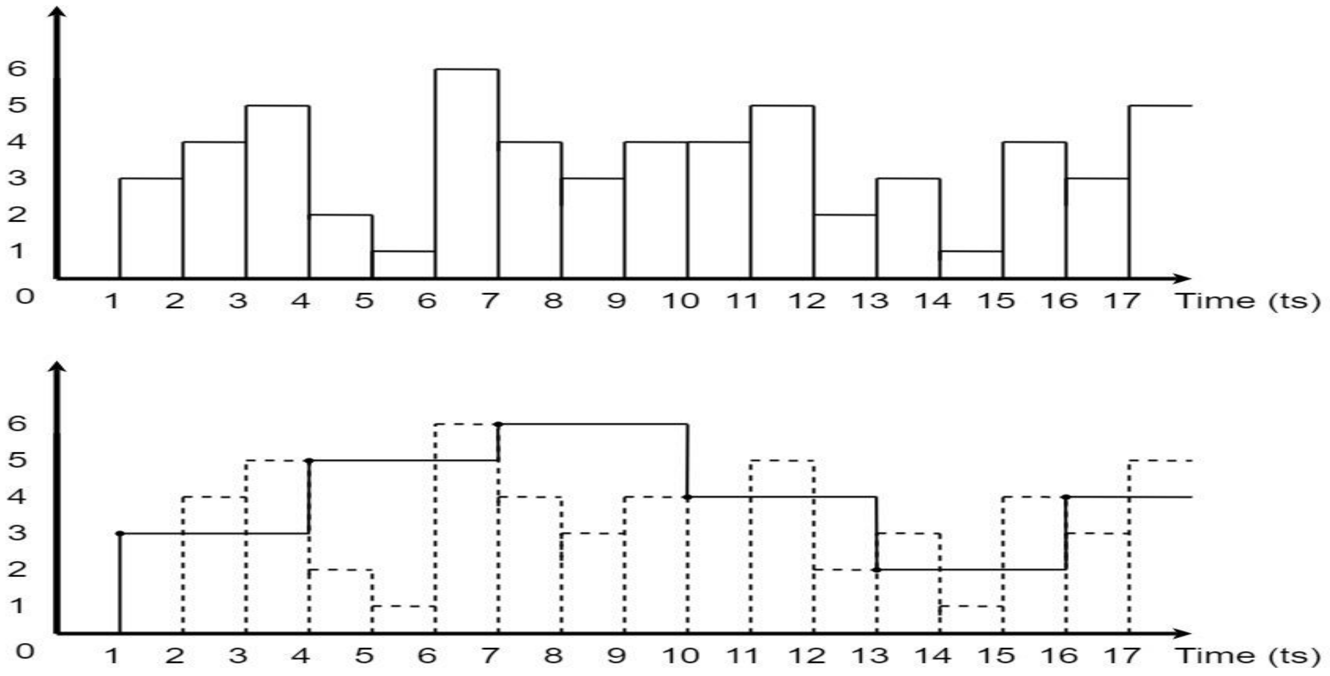


Figure 9 Down sampling in Time Domain

Before down sampler, we need anti-aliasing filter to avoid aliasing during the repetition of the signal as shown in figure 10.

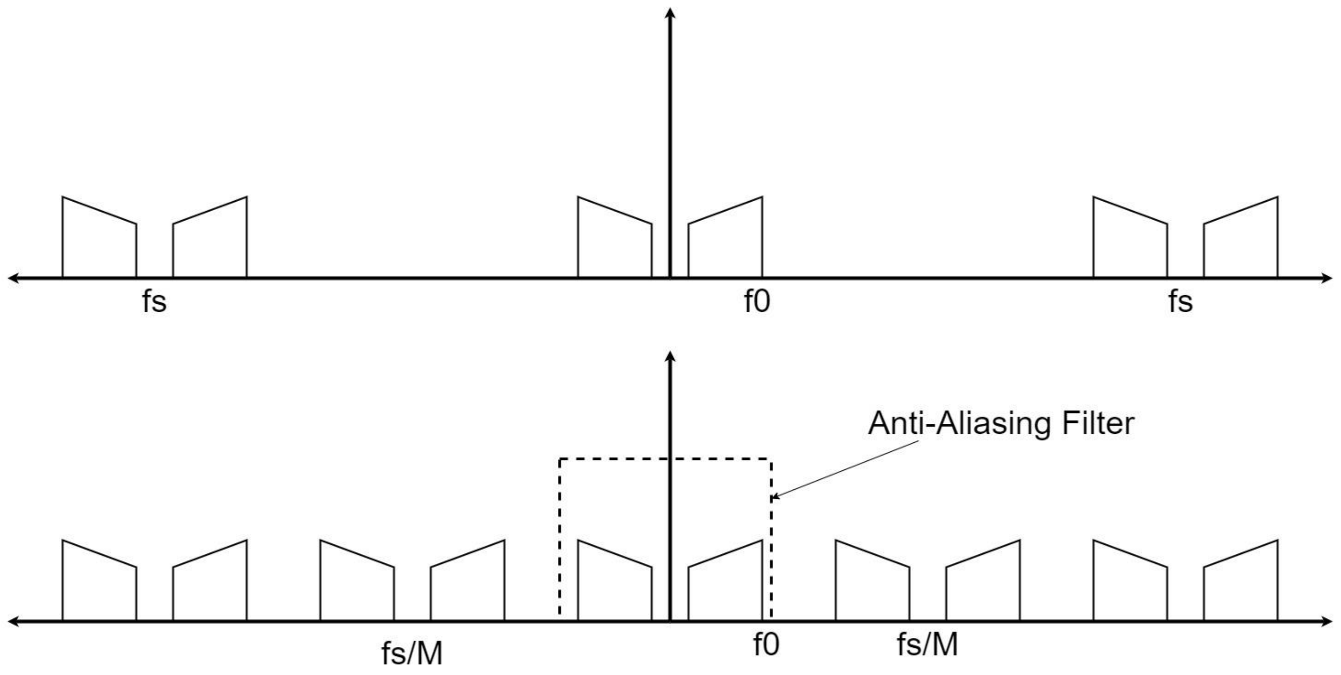


Figure 10 Down sampling in Frequency Domain

A very important aspect of the filter is the word length which affects power and area greatly, so instead of using one stage of filtering we use multiple stages with the first stage with the highest down sampling rate then the following stages are kept at the lowest ratio. The reason behind choosing this representation is that we would like to decimate as much as possible in the first stage. The following stages are kept with the minimum decimation ratio M=2 because, word length of the input signal is high, reducing the sampling frequency does not compensate for the added complexity [12].

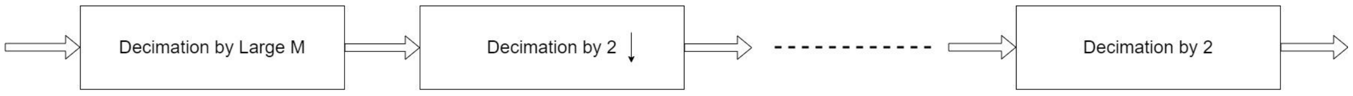


Figure 11 Decimation Stages

2.2.2 Noble Identity

In the previous stated design, the anti-aliasing filter works at the higher sampling rate which means higher clock frequency and higher power, but we can use the third noble identity to change that. Third noble identity states that filtering with $H(z^M)$ and down-sampling by M is equal to the down-sampling by M and then filtering with $H(z)$ as shown in figure 12, so now the bulk of multiplications and additions is done in the lower sampling rate domain which directly decreases power of the implementation [11].

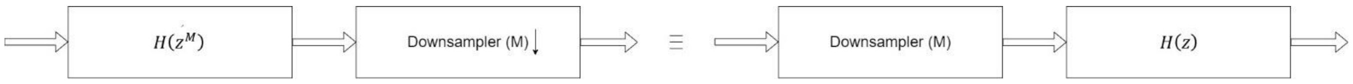


Figure 12 Noble Identity

2.2.3 Comb Filters

In signal processing, a comb filter is a filter implemented by adding a delayed version of a signal to itself, causing constructive and destructive interference. The frequency response of a comb filter consists of a series of regularly spaced notches in between regularly spaced peaks (sometimes called teeth) giving the appearance of a comb in figure 15. Comb filter used in decimation as the first stage with the high down sampling ratio. Comb filter of order k and down sampling ratio M we can write its pulse transfer function as shown in equation (2):

$$H(z)\left(\frac{1-z^{-M}}{1-z^{-1}}\right)^{k} \tag{2}$$

As shown in the figure 13 is the signal in the baseband, with normalized frequency on the old sampling rate which is much higher than Nyquist, and figure 15 is the signal after decimation with same normalized frequency (before down sampling), and lastly in figure 14 is the signal after decimation with normalized frequency on the new sampling rate.

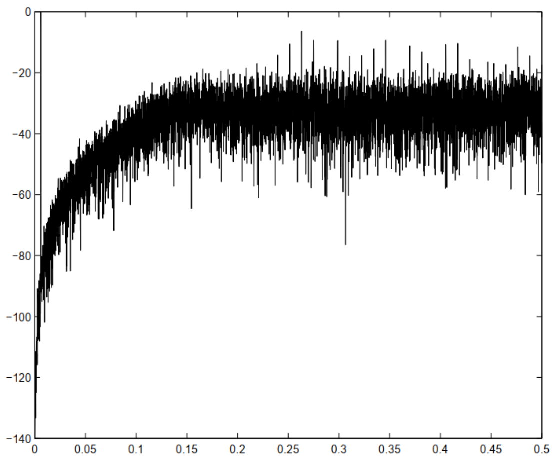


Figure 13 Comb Filter Input Signal [13]

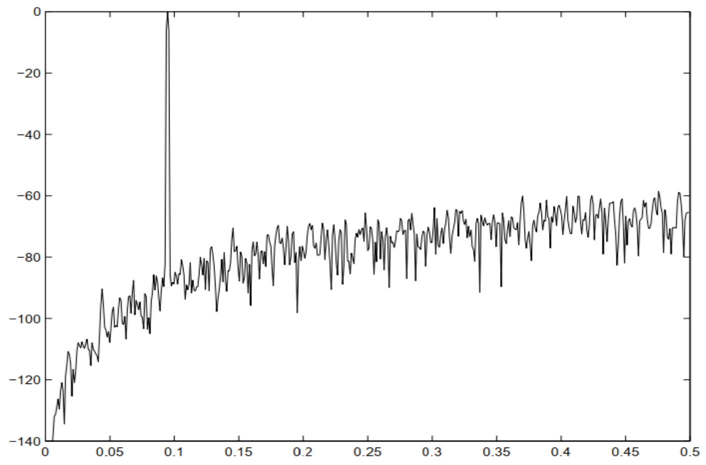


Figure 14 Comb Filter Output Normalized on new Sampling Rate [13]

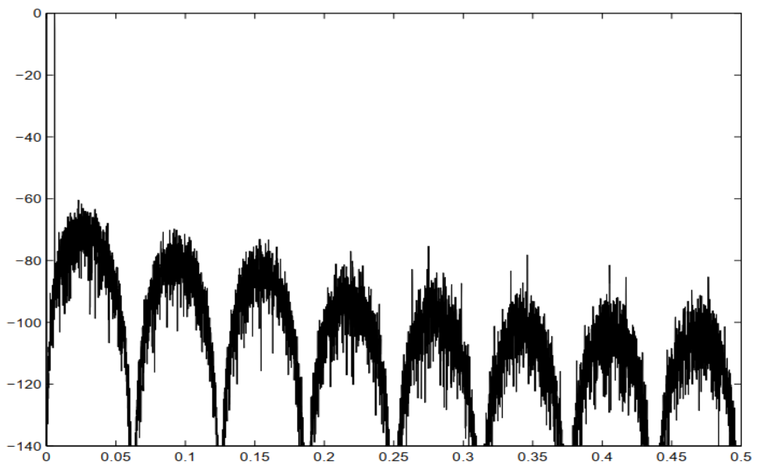


Figure 15 Comb Filter Output Normalized on old Sampling Rate [13]

2.2.4 Half band Filters

A half band filter is a decimation filter with down sampling rate M=2, and consequently, the half band filter divides the bandwidth of the signal into two equal sub bands as shown in figure 13. In the linear-phase half band filter, half of the constants are zero-valued making the implementation very attractive as shown in equation (3). Multiple Half band filters are used in cascaded form after the comb filter.

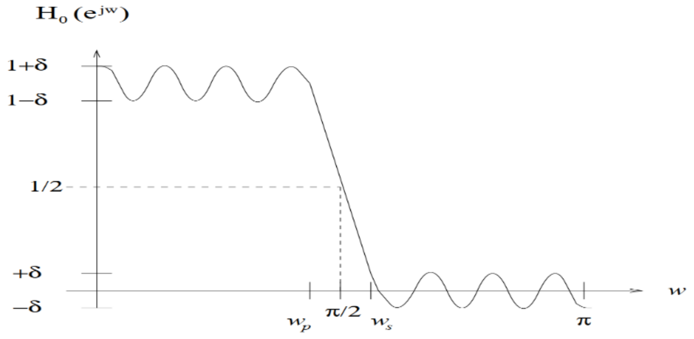


Figure 16 Halfband Frequency Response [13]

$$h[k \pm 2r] = 0 \;\; for\; r = 1,2, \dots . [\frac{K}{2}] \qquad (3)$$

The Pulse Transfer Function of the Half band with order K which is an odd number as shown in equation (4). Filter coefficients are symmetric about the middle coefficient at h[K] which is the highest coefficient in value and equals to 0.5 exactly, resulting in relation between coefficients as shown in equation (5) [11].

$$H(z) = \sum_{n=0}^{2K} h[n] z^{-n} \qquad (4)$$

$$h[2K - n] = h[n] \qquad (5)$$

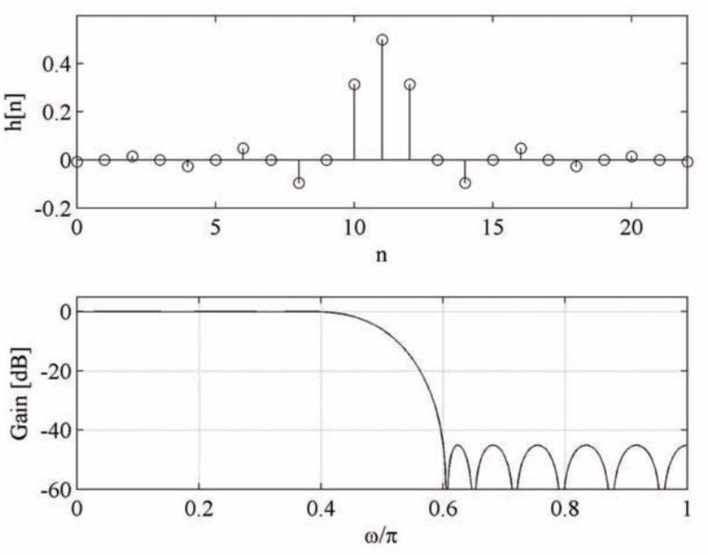


Figure 17 Half band in Time-Frequency Domains [11]

2.3. Down Conversion

Digitizing higher frequencies (i.e., IF or RF) requires the last stage of demodulation to be done in the digital domain and it is called down conversion.

2.3.1 In-phase and Quadrature Components

For passband signal carrying both I and Q components we need to generate both the components by multiplying the signal by the IF frequency ($\cos(\omega_{IF}t)$) to generate the In-phase component and by the 90° shifted version of the IF frequency ($\sin(\omega_{IF}t)$) to generate the Quadrature component as shown in figure 18.

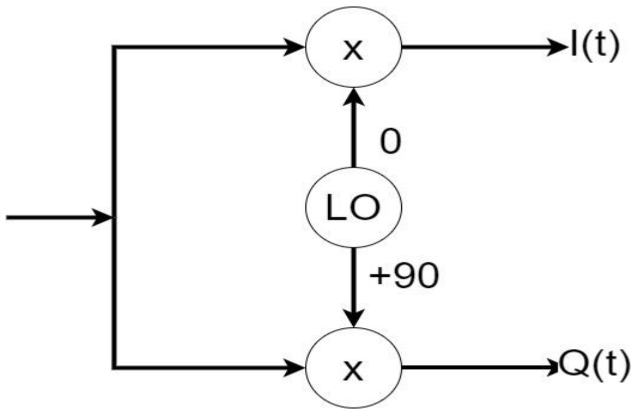


Figure 18 Demodulation Theory

2.3.2 Single Tone Digital Multiplication

Consider the received signal in the IF stage of frequency to be $x(t)$ carrying the baseband $m(t)$ signal in the amplitude and sampled after the sigma-delta ADC by sampling rate (f_s) and sample time T_s:

$$x(t)_{Before\ Sampling} = m(t)\cos(2\pi f_{IF}t)$$

$$x(t = nT_s) = m[n]\cos\,[2\pi f_{IF} * nT_s]$$

The sigma delta modulator we use is featured by having sampling rate four times the intermediate frequency, so the received signal becomes:

$$x[nT_s] = m[n]\cos\,[\frac{n\pi}{2}]$$

The sampled signals are fed to a digital I/Q down converter, and the I/Q outputs of the down converter ($[I]$ and $[Q]$) are given by:

$$I[n] = \cos[2\pi f_{IF} * nT_s] * x[n] = \cos\left[\frac{n\pi}{2}\right] * x[n]$$

$$I[n] = \frac{m[n]}{2} * \left(\cos\left[\frac{n\pi}{2} + \frac{n\pi}{2}\right] + \cos\left[\frac{n\pi}{2} - \frac{n\pi}{2}\right]\right) = \frac{m[n]}{2}(\cos[n\pi])$$

$$Q[n] = \sin[2\pi f_{IF} * nT_s] * x[n] = \sin\left[\frac{n\pi}{2}\right] * x[n]$$

$$Q[n] = \frac{m[n]}{2} * \left(\sin\left[\frac{n\pi}{2} + \frac{n\pi}{2}\right] + \sin\left[\frac{n\pi}{2} - \frac{n\pi}{2}\right]\right) = \frac{m[n]}{2}(\sin[n\pi])$$

Where (n) is integer (multiplies of T_s) so we can look at the demodulation as multiplication by three levels (+1, 0, -1) I/Q Local oscillator sequences as shown in the figure 19.

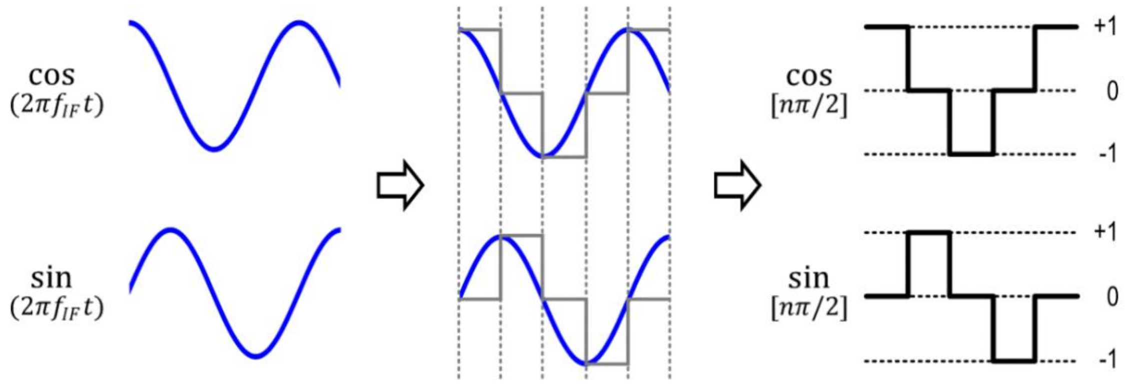


Figure 19 Single Tone Multiplication for Fs=4 IF [1]

2.4. Antenna Array

Antenna array is a set of multiple connected identical antennas which can be arranged in a linear or planar array. Each antenna in this array is called an element. Each element will receive a different delayed version of the signal due to the spacing between them and by adding them together it can be destructively or constructively interfered.

2.4.1 Array Parameters

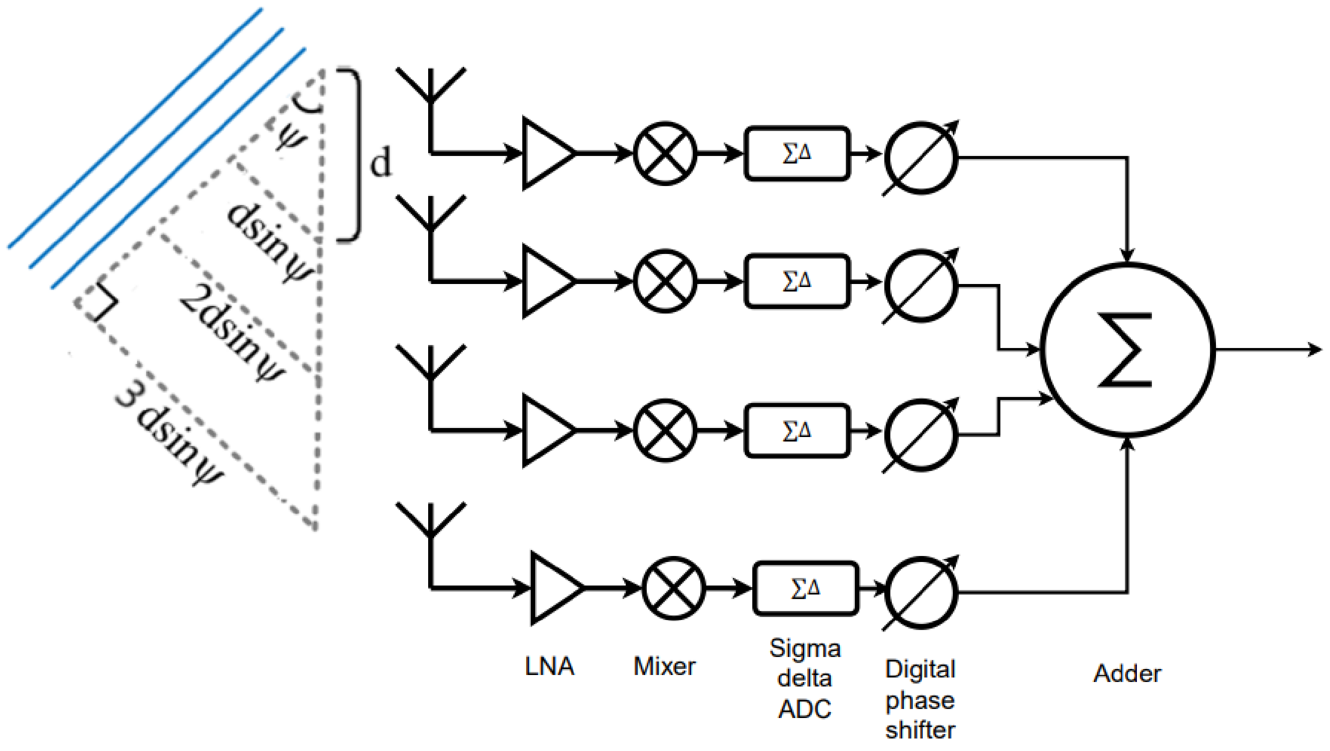


Figure 20 Phased Antenna Array [5]

The antenna elements receive different delayed versions of the signal due to the time delay difference τ_K between them where:

$$\tau_K = \frac{k\,\mathrm{d}\sin\psi}{c} \tag{6}$$

Where k is the element number, d is the distance between two adjacent antennas, c is the speed of light and ψ is the incident angle.

The distance between two adjacent antennas is usually chosen to be half the carrier wavelength in order to have a good tradeoff between 3-dB beamwidth and number of grating lobes so the equation can be written as:

$$\tau_K = \frac{k\ \sin\psi}{2\,f_C}$$

By defining $\tau = \frac{sin\psi}{2\, f_c}$, it can be simplified to be: $\tau_K = k\ \tau$

When a phased array is receiving a narrowband sine wave signal centered at f_c, the k_{th} element receives a signal $R_k(t)$ which is:

$$R_k(t) = \cos((f_c + \Delta f)(t - \tau_k))$$

$$R_k(t) = \cos((f_c + \Delta f)t + (f_c + \Delta f)\tau_k)$$

For narrowband signals we can approximate $(f_c + \Delta f)\ to\ f_c$ so we can simplify to reach equation (7)

$$R_k(t) = \cos((f_c + \Delta f)t + f_c\tau_k) \tag{7}$$

So now a phase shift of $f_c\tau_k$ can replace that time delay of τ_k, this phase shift is independent of the signal frequency over the bandwidth, actually it's independent of the carrier frequency too as shown in equation (8)

$$f_c\tau_k = \frac{f_c k \sin\psi}{2\, f_c} = k\pi \sin\psi \tag{8}$$

So, this phase shift is only dependent on the steering angle.

2.4.2 Space Factor

The space factor equation expresses the normalized beam gain over the space as in equation (9).

$$|\mathrm{SF}(\psi)| = \left| \frac{\sin\left(\frac{N\,kd}{2}(\sin\psi - \sin\psi_0)\right)}{N\ \sin\left(\frac{kd}{2}(\sin\psi - \sin\psi_0)\right)} \right| \tag{9}$$

Where N is the number of elements, $k = \frac{2\pi}{\lambda}$, (d) is the distance between two adjacent elements, ψ is the incident angle and ψ_0 is the steering angle.

As the number of elements increases the gain increase and the beamwidth becomes narrower and more grating lobes will appear.

Figure 21 shows 4 elements

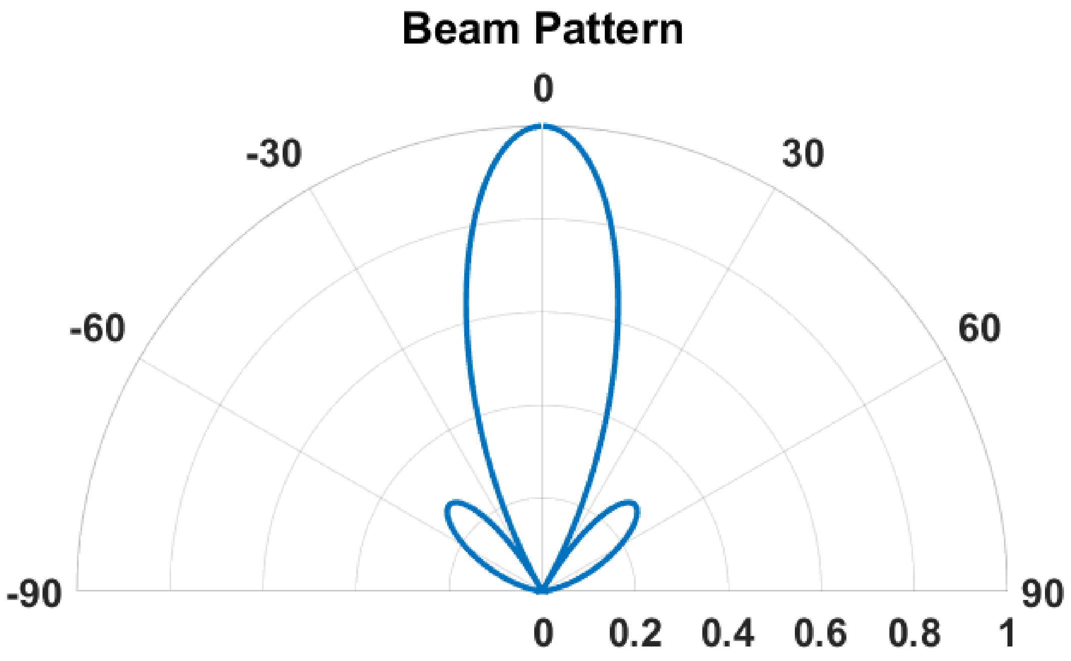


Figure 21 Space Factor Beam for 4-Elements Array

Figure 22 shows 8 elements

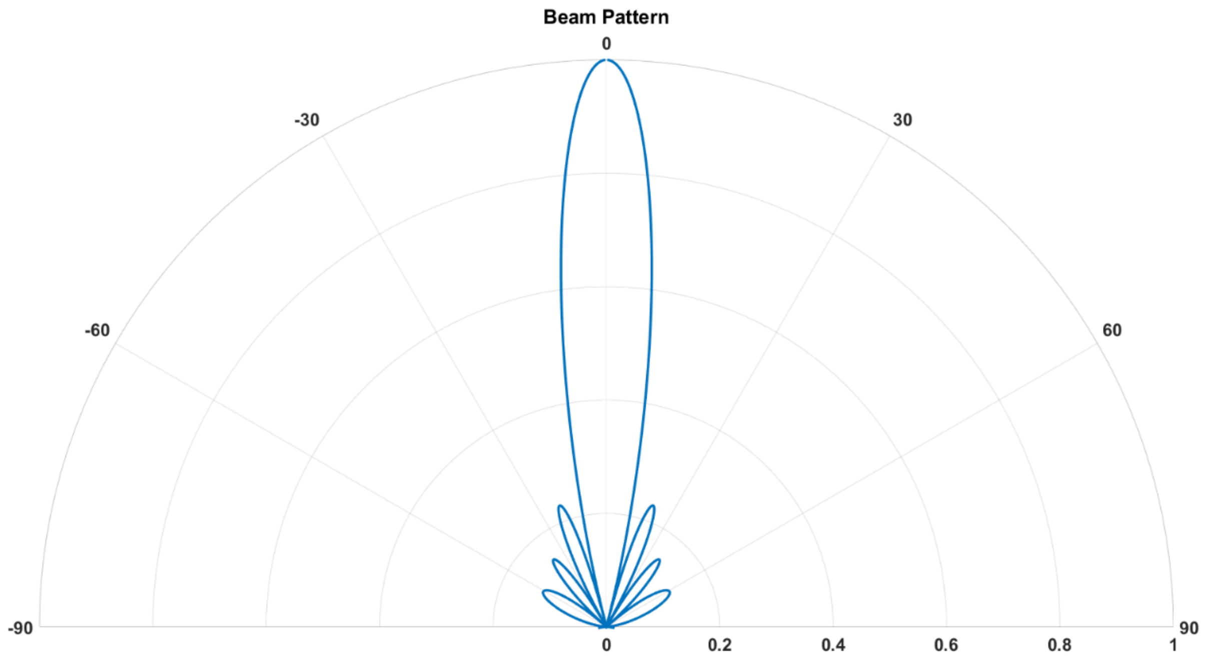


Figure 22 Space Factor for 8-Elments Antenna Array

.

2.5. Digital Phase Shifters

For narrowband signals, beamforming is often implemented with phase shifters since a time delay can be approximated by a constant phase-shift over the bandwidth of interest.

In digital beamforming (DBF), incoming signals received at an antenna array are down-converted to baseband I/Q signals, and digitized by ADCs. By controlling the phase of each down-converted signal (x_k) at the kth element with DSP, signal paths are constructively or destructively combined according to the added phase on each element.

2.5.1 Complex Weighted Multiplier (CWM)

We seek to add phase shift to the signal with phase shift value θ regardless of the frequency of the signal and the stage of its carrier

$$I = \sin(\omega t)\ ,\ \ Q\ = \cos(\omega t)$$

Phase Shift Value $= \theta$

$$I' = \cos(\theta) * \sin(\omega t) + \sin(\theta)\cos(\omega t)$$

$$Q' = -\sin(\theta) * \sin(\omega t) + \cos(\theta)\cos(\omega t)$$

From the reverse of trigonometric functions of two added terms:

$$I' = \sin(\omega t + \theta),\ \ Q' = \cos\,(\omega t + \theta)$$

To achieve a phase-shift of θ, the baseband I/Q signals are scaled, and combined to generate I'/Q' phase-shifted outputs as in equation (10).

$$\begin{aligned} I' &= I\cos\theta + Q\sin\theta \\ Q' &= -I\sin\theta + Q\cos\theta \end{aligned} \tag{10}$$

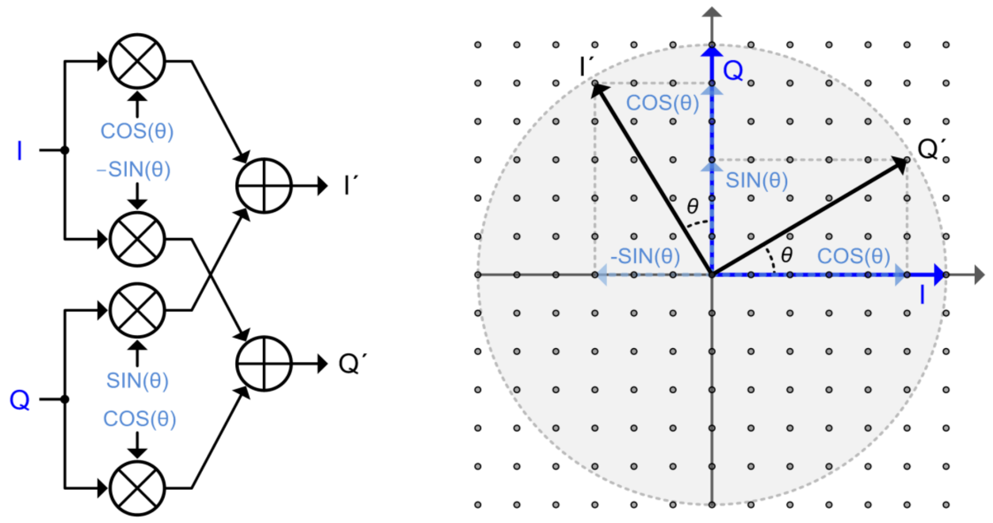


Figure 23 CWM Vector of Shifting [1]

When the I/Q signals are represented as a complex signal, the above operations are equivalent to multiplication by $e^{j\theta}$. For this reason, this technique is often called complex weight multiplication (CWM). For a uniformly spaced N-element linear antenna array, a complex weight of $ej^{(k\theta)}$ adjusts the delay at the kth element, and then all signal paths are combined to create a beam $\sum_{k=0}^{N} x_k e^{jk\theta}$, as shown in equation (11) the matrix of phasing of CWM is constructed.

$$R_k(t) = [\cos(\omega_c t) \quad \sin(\omega_c t)] \begin{bmatrix} \cos(\omega_c \tau_k) & -\sin(\omega_c \tau_k) \\ \sin(\omega_c \tau_k) & \cos(\omega_c \tau_k) \end{bmatrix} \begin{bmatrix} I_k \\ Q_k \end{bmatrix} \tag{11}$$

2.5.2 Finite Complex Weight Resolution

Digital phase shifter has a finite resolution due to the finite number of bits where the weights of the sine and cosine can be represented, so as be increasing the number of weights' bits the resolution increases.

There is a trade-off between the resolution and the power-area-product of the digital system, number of bits is chosen according to the required steering accuracy and phase shift resolution. By choosing specific number of bits for the weights of sine and cosine we specify the number of levels.

We represent all of the data in integer representation of value which means our representation of values is fixed floating point with point at the end after the least significant bit (all fractions are truncated). Weights which are only fractions (excluding sin (90) and cos (0) and their repetitions) have to be multiplied by the highest number in the range before it inputs our system. In other words, if we represent the weights in 3 bits only, weights' range will be [-(2^{3-1}-1), (2^{3-1}-1)] which is the range of signed representation in 3-bits excluding – (2^{3-1}) as we can only use values [-scale, +scale] to represents the weights that ranges [-1, +1].

Scale for weights represented in (n) bits can be calculated from equation (12)

$$Scale = 2^{n-1} - 1 \tag{12}$$

For specific scale the resulting average phase shift step is the quotient of division the 360 degrees by the number of used vectors for the weights squared as they are 2 weights. For 3-bits representation of weights we achieve a total of 49 levels $(2^3-1)^2$ but only 24 levels are used which are the levels nearest to the circle as shown in figure 24, which results in 15 degrees phase shifter average step.

There will be also phase shift error (phase shift variations) and amplitude error (amplitude variations) due quantization error in finite number of bits used for weights which can be seen in figure 24, there is difference in value between points and the circle causing these variations.

The three main features of the CWM are determined by the number of bits to represent the weights of the sine and cosine.

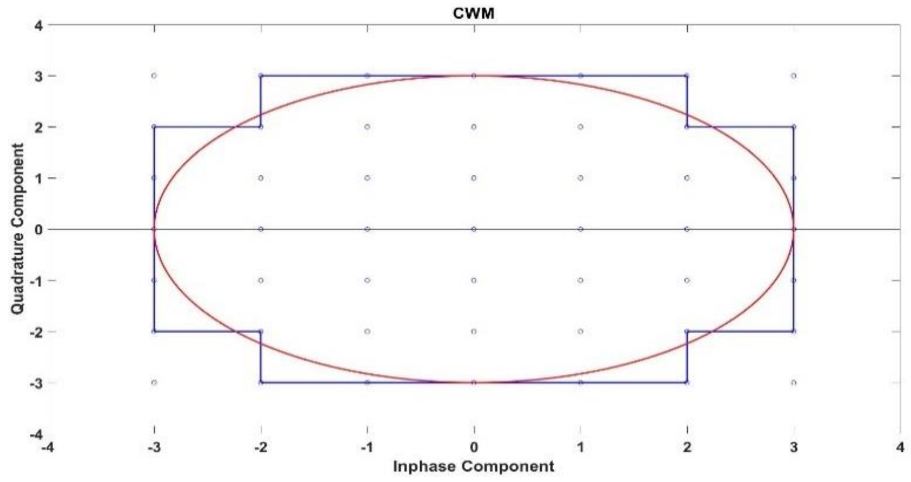


Figure 24 CWM Vector 3 bits

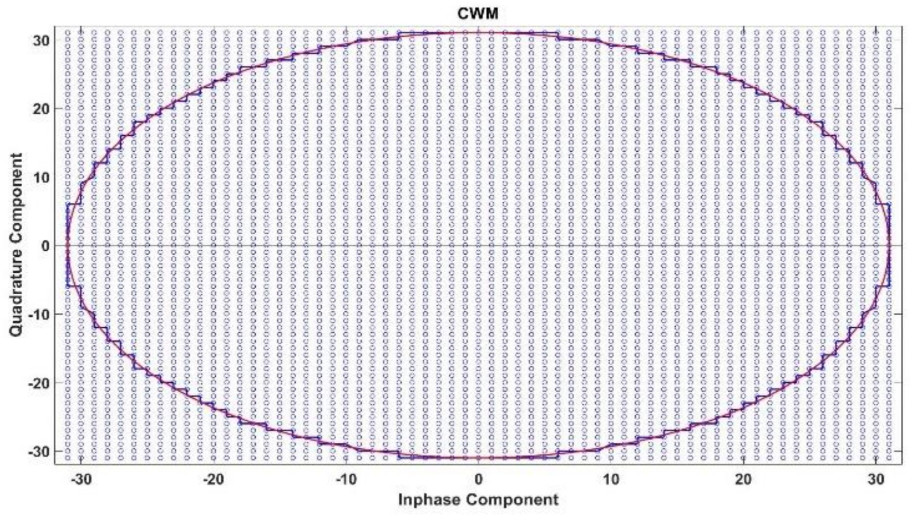


Figure 25 CWM Vectors 6 bits

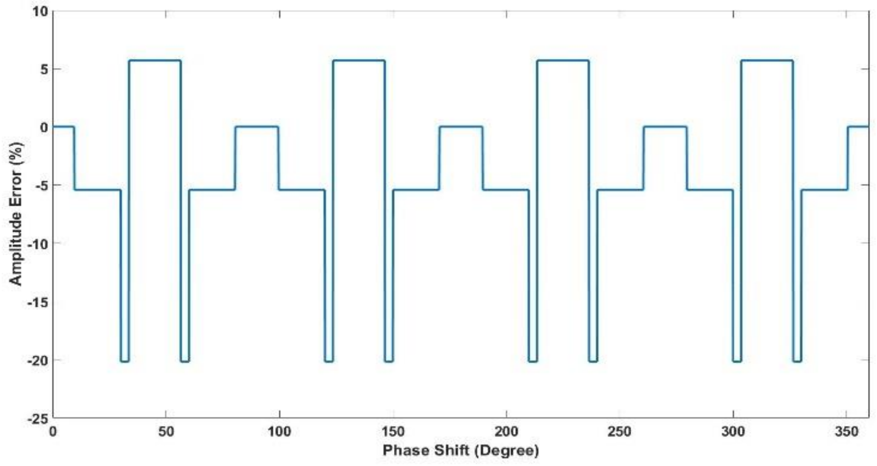


Figure 26 Amplitude Error 3 bits

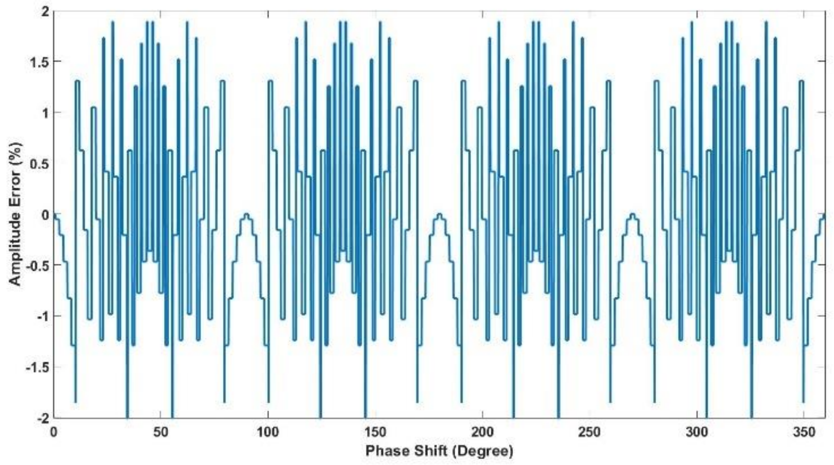


Figure 27 Amplitude Error 6 bits

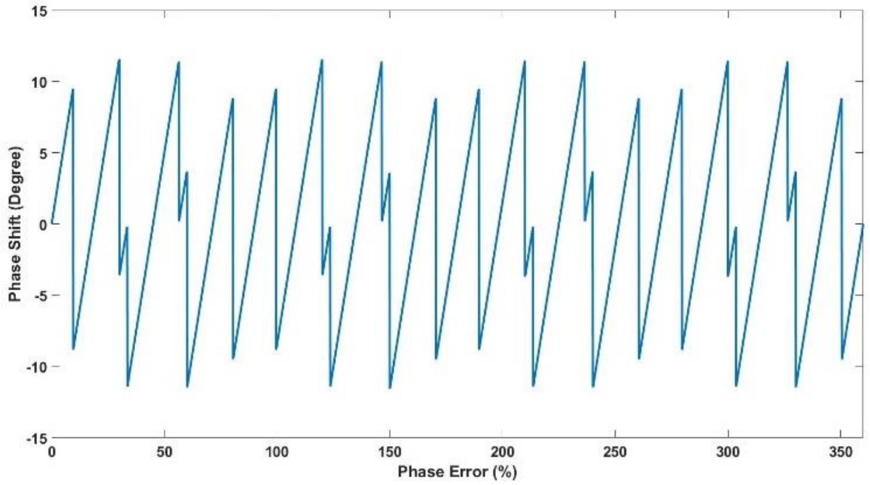


Figure 28 Phase Error 3 bits

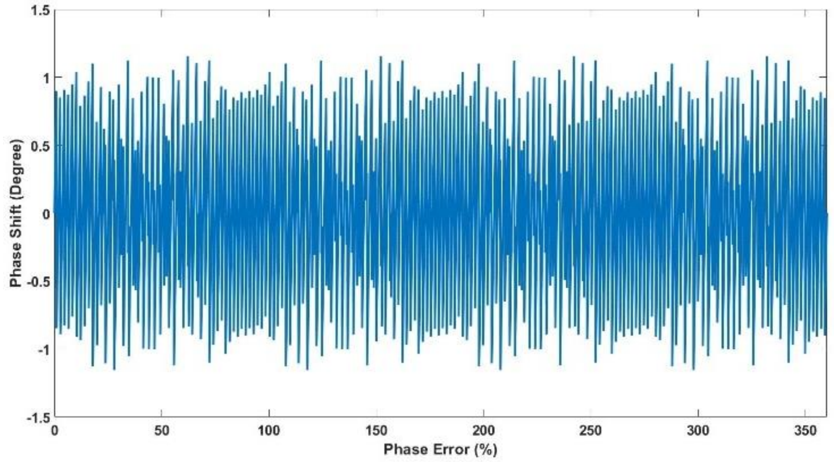


Figure 29 Phase Error 6 bits

By the same manner we can calculate the CWM features for number of bits starting from 3-bits till 8-bits where the max variation in amplitude <1% and max variation in phase <0.6 degrees as shown in table (1) and in figures (27), (29).

Phase shift angle step is used for table (1) is 0.1 degrees as the angle step used in simulations will change these numbers by very small value.

Weighting factor resolution [bit]	3	4	5	6	7	8
Total phase-shift steps	24	56	120	240	488	960
Amplitude variation [%]	25.904	17.323	8.201	3.899	2.04	0.998
Phase variation [º]	23.13	9.90	4.80	2.30	1.23	0.580
Average phase-shift step size [º]	15	6.428	3.0	1.5	0.737	0.375

Table 1: CWM Features

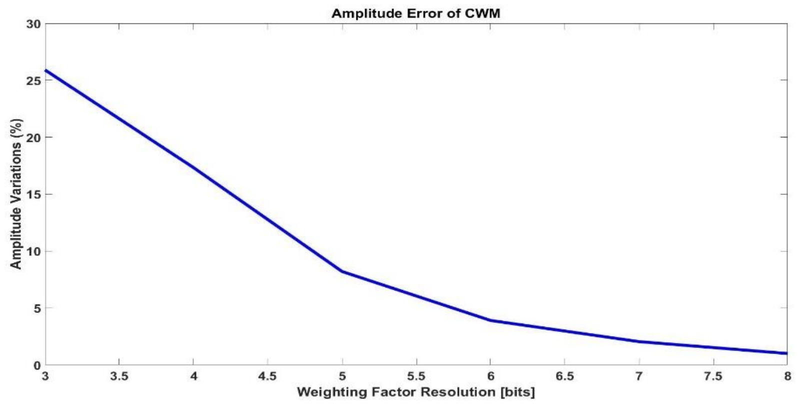


Figure 30 CWM Amplitude Variations Different Number of Bits

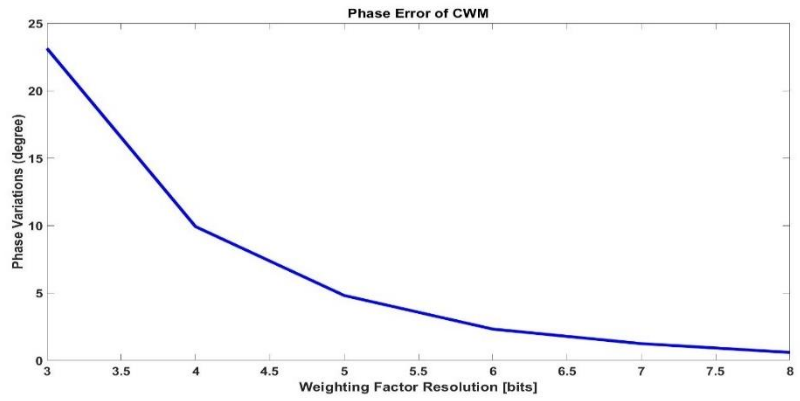


Figure 31 CWM Phase Variations Different Number of Bits

Chapter 3: Implementation

3.1. Bit-Stream Processing

The 1-bit continuous bandpass sigma delta is oversampled with oversampling ratio 32 so it needs a decimation filter after it. This decimation filter down samples the signal to meet Nyquist rate and act as a low pass filter so it increases the number of bits.

After decimation, DSP can be performed at lower clock rates but with much larger word length. Instead of doing the processing after the decimation we can directly processed the sigma delta output and take advantage of that low word length which is only 1-bit as in figure 32.

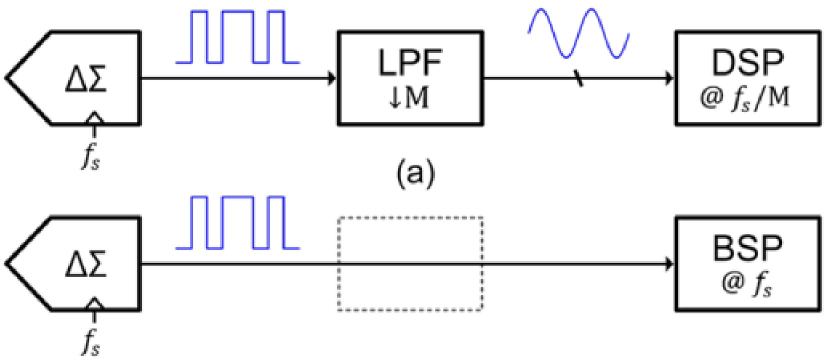


Figure 32 BSP vs DSP Operating Frequency [2]

Another advantage of BSP over the conventional DSP that it reduces the number of decimators to just two, one for the in-phase signal and another for the quadrature-phase one. In the conventional DSP each sigma-delta ADC requires a decimation filter after it. In Bit Stream Processing (BSP) only the final output needs to be decimated as shown in figure 33.

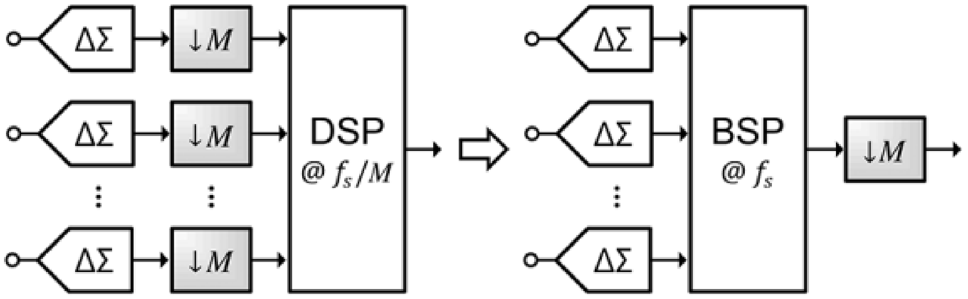


Figure 33 BSP vs DSP Number of Decimators [2]

Since decimation consumes a lot of power and requires a large area, the BSP helps to significantly reduce the power consumption and area of the entire system.

3.1.1 Multiplication As MUX

BSP generally replaces multipliers with simple MUXs for multiplication with a bit-stream as shown in Fig. 1. The five-level stream which consists of -2, -1, 0, +1, and +2. The bit-stream controls a MUX to multiply the input bit-stream by a multi-bit coefficient, W. When the value of the five-level stream is negative, the sign of W is inverted to implement multiplication by -1. When the value of the five-level stream is +2, W is left shifted to multiply by +2. When the value of the five-level stream is -2, both sign inversion and 1-bit left shift are performed to implement multiplication by -2 [4].

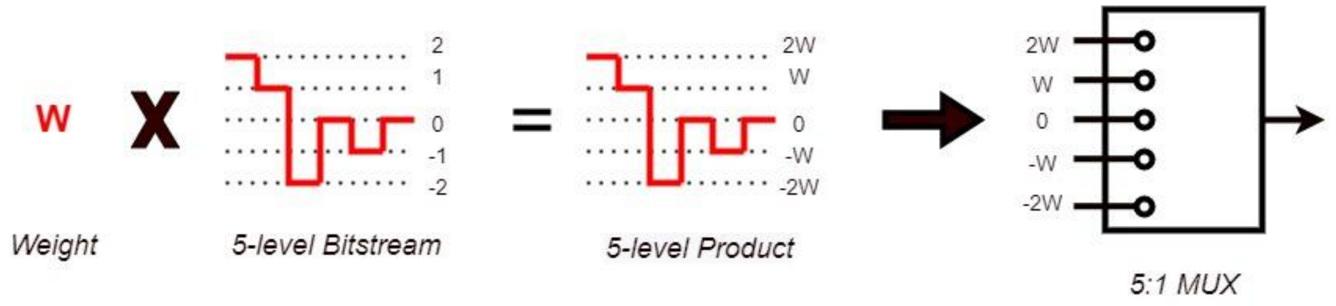


Figure 34 5-Level Bitstream Multiplication

The previous discussion was about five-level bit stream as it is the least number of bits for bit stream to be considered BSP, higher number of bits requires multipliers and can’t work with MUX implementation.

In our system it is better case as our sigma-delta ADC outputs 1-bit stream, resulting in 2:1 MUX instead of 5:1 MUX with MUX output of number of bits same as the weight W instead of increasing one bit due to the removal of the 2W case that occurred in the previous implementation. As shown in figure 35 the 5:1 MUX is replaced by 2:1 MUX with 3 levels output as the weight has zero level introduced on the stream, and in figure 5 5:1 MUX is replaced by 2:1 MUX as the weight doesn’t introduce zero level.

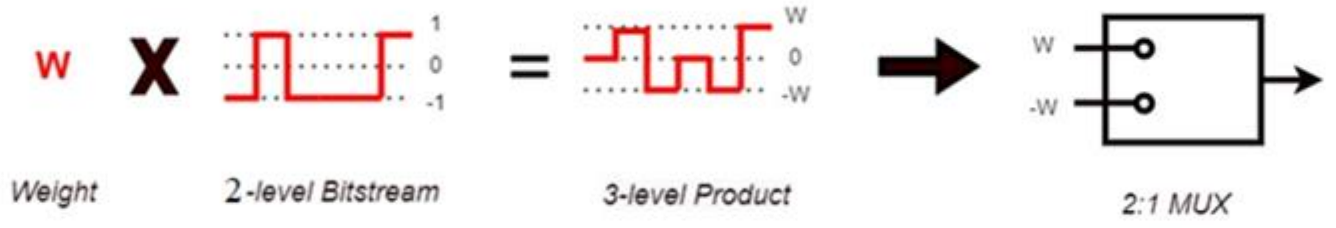


Figure 35 2-level Bitstream Multiplication by 3-level weight

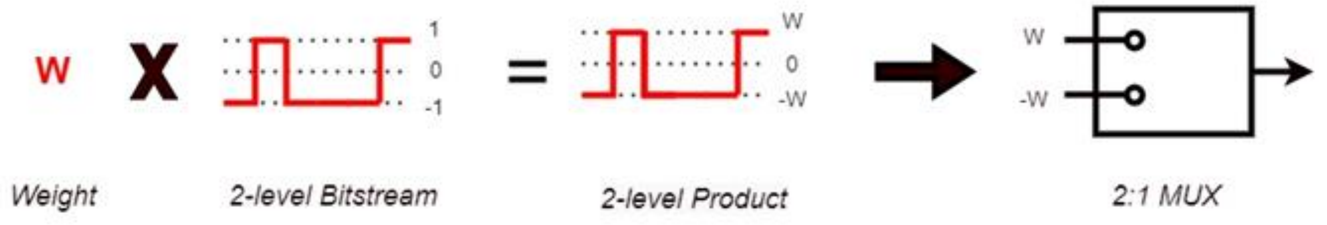


Figure 36 2-level Bitstream Multiplication by 2-level weight

3.1.2 Interleaved BSP

The system input is from a band pass sigma delta modulator, the signal is carried on an intermediate frequency (IF) then it will be down converted to baseband in the digital domain using digital down conversion (DDC). The IF is quarter the sampling frequency which make it easy to implement the DDC. Multiplication by a single tone with sampling rate four times the IF frequency as stated in figure 19 is 3-level multiplication by the input which is 2-level bitstream resulting in multiplication same aa shown in figure 35 resulting in output two bits instead of the one-bit stream coming from the sigma-delta ADC.

Modification I:

Instead of using 2:1 MUX with 3-level output for single tone multiplication we can make use of the zero repeats every 1 sample and implements as down sampler by 2 followed by 2-levels weight multiplication resulting in 2:1 MUX with 2-level output instead of 3-level output as shown in figure 37.

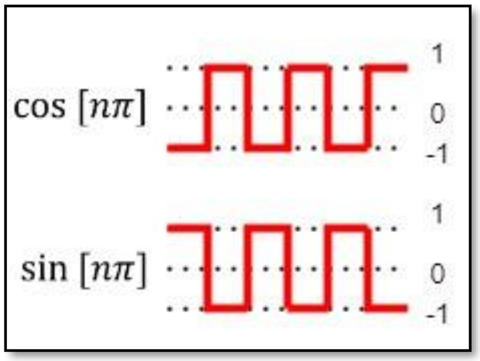


Figure 37 Demodulator Two Components

Modification II:

Generating I/Q components requires multiplication by the frequency and its normal (90 degrees phase shifted version as shown in figure 19), so the digital system will be obligated to generate clock (for cosine) and its inversion (for sine) which very high frequency. Instead, we make use of the ratio between IF frequency and sampling rate that shifts the signal by 90 degrees for every single delay in the IF stage, logically we generate delayed version(Q) of the signal and multiply both by the frequency without generating inverted clock.

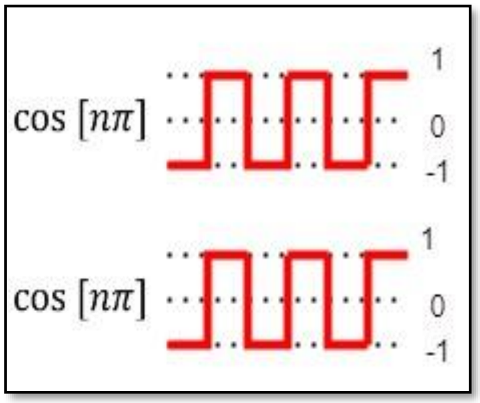


Figure 38 Demodulation by 1 Component Only

We can combine both modifications in one block called Interleaver with the function of generating down sampled Inphase and Quadrature components of the signal by down sampling ratio 2 which is down converted to baseband in the next block by multiplying both by the frequency as show in figure 39.

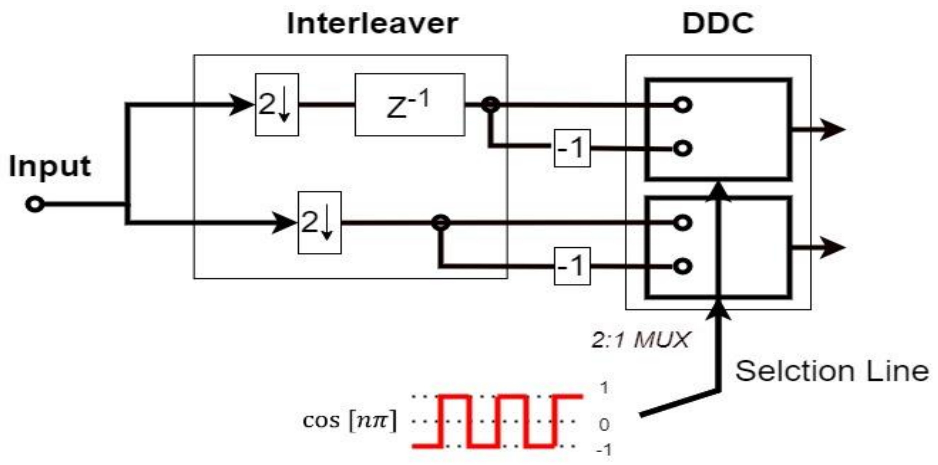


Figure 39 Interleaved BSP

3.2. Beamforming Implementation

Beamforming Implementation is considered the implementation of phase shifters and adders till the filter input, in which the constructive interference happens, and spatial multiplexing takes place.

3.2.1 Phase Shift Implementation

The phase shift is implemented using CWM technique which scale the I/Q baseband signals and combine them to generate I'/Q' shifted outputs as in equation (10)

To implement these equations, we need two multipliers and an adder for each component. Since the input I/Q signals are 1-bit this multiplier can be implemented using 2:1 MUX as shown in figure 40.

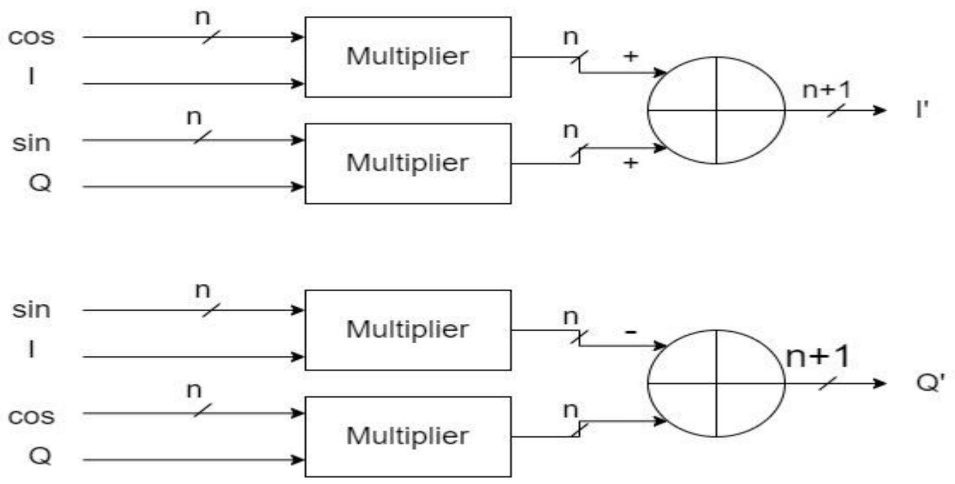


Figure 40 CWM Implementation

As shown in fig. if the weight of the sine and cosine is represented in n-bits, then the output of each element will be two phase-shifted I'/Q' in (n+1)-bits, for four elements outputs will be summed to generate final I'/Q' which is represented in (n+3)-bits as shown in figure 41. Where the output of this addition is the output of beamforming stage and decimations filter input, the number of bits for the filter input (N) is really important as filter has the bigger part in area and power of the whole system, decreasing the number of bits of filter input (N) deceases area and power of system remarkably.

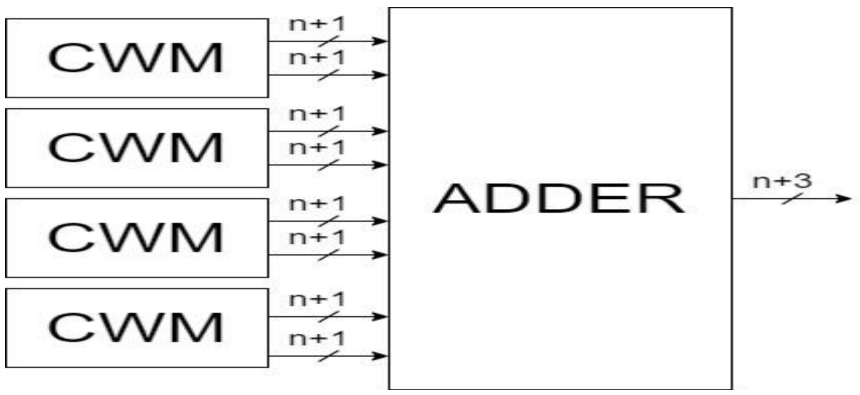


Figure 41 Four-Elements Beamforming Stage

As shown in figure 42 for eight elements we can consider it two of the four elements beamforming which results in adder with beamforming output number of bits is (n+4). Using same logic sixteen elements output number of bits is (n+5), and so on.

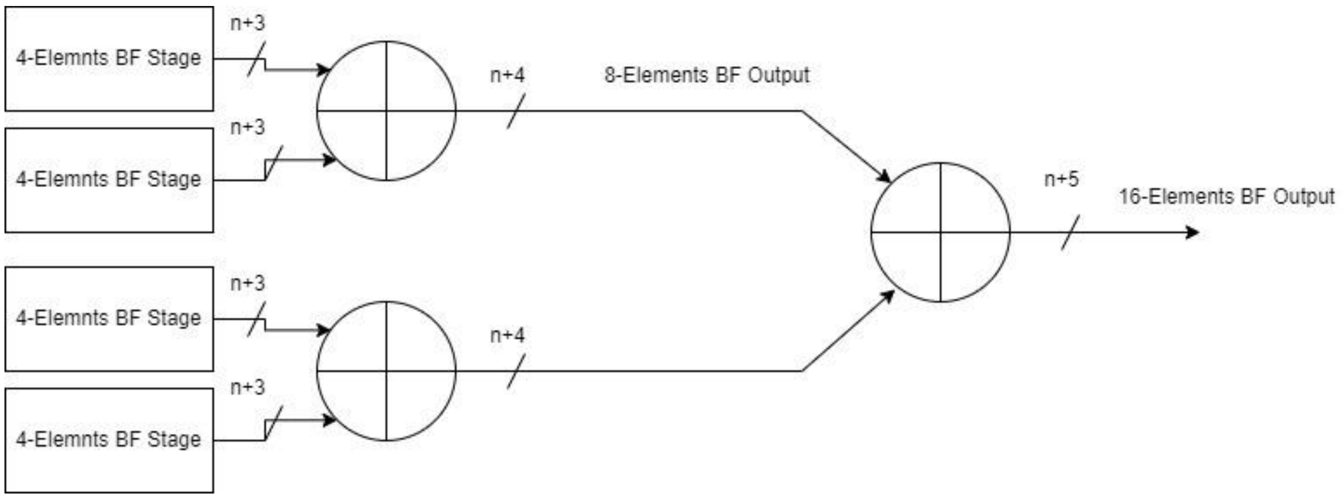


Figure 42 Using 4-Elemnts block to from higher number of elements

3.2.2 CWM and Number of bits

Up till now all calculations of number of bits used the scale as in equation (12), which focuses on the number of bits of weights (n) in calculation the scale but in reality, the number of bits we are interested in minimizing is the beamforming output or filter input (N). How can we get better scale than the conventional scale? Before we answer this question, we need to observe the drawbacks of the conventional scale and its effect on the accuracy of CWM and decimation filter.

For number of bits used for weights (n) equals to 5 resulting in scale equals to 15 and number of bits for filter inputs equals to 8 (considering 4-elemnts array). Here the problem arises and can be observed as the filter input (N) doesn't use the full range supported by the number of bits (N) which results in waste of levels for the filtering process, in other words we can say that same filter area and power can support higher CWM accuracy but the conventional scale limits it.

Focusing on the number if bits of filter input (N) we need to start by finding the maximum value that can inputs the filter:

Filter input (I- component in 1st quadrant) $I_0 sin(0) + Q_0 cos(0) + I_1 sin(\theta) + Q_1 cos(\theta) + I_2 sin(2\theta) + Q_2 cos(2\theta) + I_3 sin(3\theta) + Q_3 cos(3\theta)$

From figure 43

Max Value of Filter Input = 5.058

Min Value of Filter Input = - 5.058

Which is less than summation of max of each = 8, and this is because each elements adds sine and cosine and same angle so they can never be maximum at same time.

New Scale can be calculated by the formula shown in equation (10)

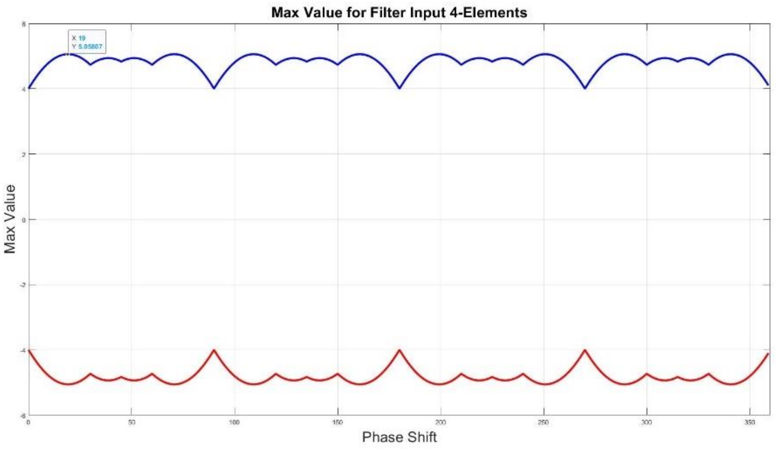


Figure 43 Max Value of Filter Input

$$Scale = floor\left(\frac{2^{N-1}-1}{Max\ of\ filter\ input}\right) \qquad (13)$$

Improvement can be seen in table (2) for four elements array.

Filter Input(N)	Old Number of Bits of Sine (n)	Conventional Scale	Old Avg CWM Step	Modified (n)	Modified Scale	Modified Avg CWM Step
6	3	3	15°	4	5	9°
7	4	7	6.428°	5	12	3.75°
8	5	15	3.0°	6	25	1.8°
9	6	31	1.5°	7	50	0.9°
10	7	63	0.737°	8	101	0.457°

Table 2: Effect of Modifying Number of Bits

We can calculate the scale for eight and sixteen elements’ arrays as shown in table (3).

N for 8-Elements	New Scale	N for 16-Elements	New Scale
7	6	8	6
8	12	9	12
9	24	10	24
10	49	11	49
11	99	12	98

Table 3: New Scale for 8 and 16 Elements Arrays

In case of using higher scale than the listed in the previous scale results overflowing in CWM and some angles called Forbidden Angles, where the CWM can’t shift by this value and will fail if asked to do so. In some cases, the number of forbidden angles is very small where it will not affect steering giving the chance for scale be increased by one at maximum. For four elements array with filter input (N) of 6 bits we can increase the new scale from 5 to 6 resulting in 8 forbidden angles happens at 8 steering values, all of the steering values are fractions. Forbidden angles for the stated case only appear for steering resolution lower than half.

Increasing scale any more than the listed in table resulting in limiting the resolution or worse so we need to be careful and study well the effect of that increase. For the case of four elements with scale 5 we can increase the scale to 6 with average CWM step of 7.5°, this limits the steering resolution of the system to 0.5°.

We can observe that value of scale changes for different number of array elements for the number of bits for the weights and this is because the quantization error as it might build up or eliminate quantization of each element.

3.3. Filter Implementation

The implementation of decimation filter we use are FIR realization as it more suitable than the IIR filters, The major drawback of this architecture is that the IIR filter is operating at maximum sampling frequency and with a very large word length.

The average power consumption of a digital signal processing system is proportional to the number of operations performed per sample, the word length and the sampling frequency.

3.3.1 Polyphase Implementation

A higher-order FIR filter can be realized in a parallel structure based on the polyphase decomposition of the transfer function. The FIR transfer function is decomposed into *M* lower-order transfer functions, called the polyphase components, which are afterwards added together to compose the original overall transfer function as shown in figure 44 (a). The polyphase components are sometimes called polyphase sub filters or polyphase branches. A polyphase component is usually implemented in the direct transversal form. As shown in figure 44 (b) decimator composed of the cascade of an FIR filter implemented as a parallel connection of *M* polyphase branches, and factor-of-M down-sampler. Here the arithmetic operations in the polyphase branches are to be performed at the input sampling rate, i.e. at the higher sampling rate of the system. Instead of down-sampling at the filter output, one can shift the down-sampling operation into the polyphase branches before the output adders. This modification opens the opportunity of applying the Third Nobel Identity and to arrive to the efficient implementation [11].

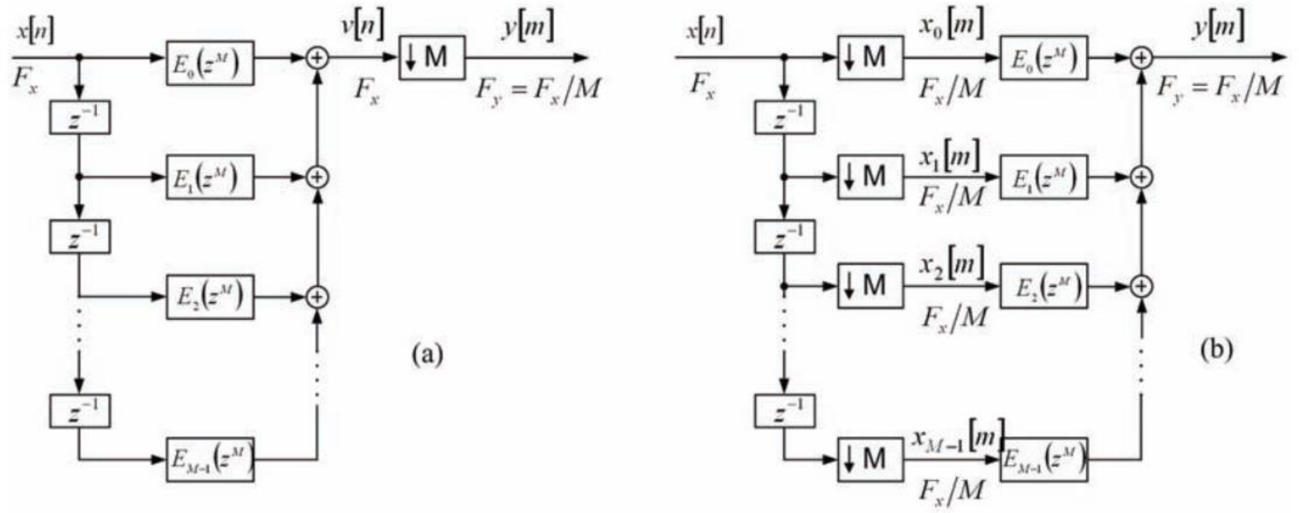


Figure 44 Polyphase Implementation of Filters [11]

In this structure, the down-sampling-by-M occurs at the inputs of the polyphase components and filtering is performed at the lower sampling rate. The overall computational complexity of the decimator is reduced by M.

For Comb filter of order k and down sampling ratio M we can generate the coefficients from the relation as shown in equation (11)

$$H(z) = \left(\sum_{i=0}^{M-1} z^{-i} \right)^{k} \tag{11}$$

Where the generation of Half band filter can be done by MATLAB for specific order. Our Half band coefficients are supplied to use by Seamless waves.

Another Improvement can be done due to symmetric coefficients of the filter by combining each two branches of the same coefficients and preform the multiplication once instead of twice for each branch.

3.3.2 Results of The Polyphase Implementation

Here we show and test the validity of our implementation for comb and the two half bands which down samples by ratio 32, comb filter of 31 order decimating by 8 and each half band of order 24 decimating by 2.

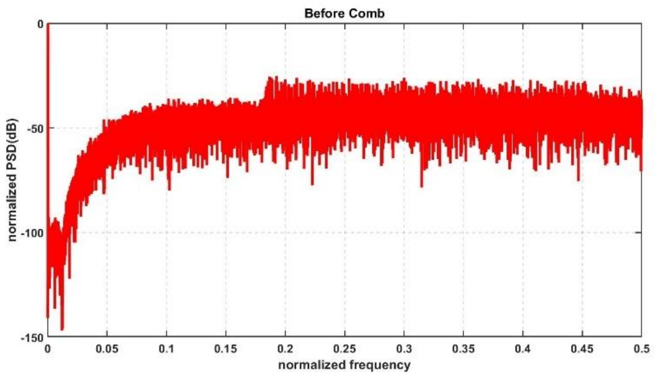


Figure 46 Signal in Baseband Before Comb

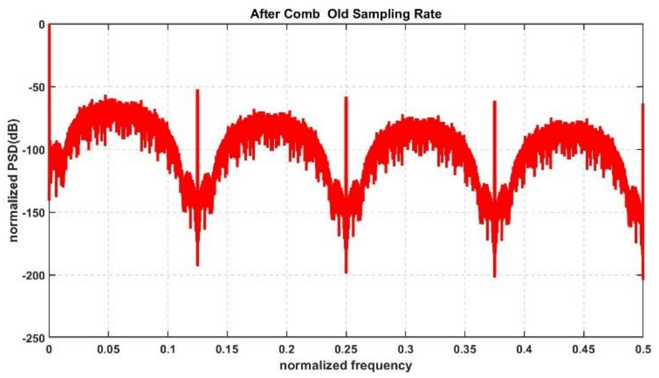


Figure 45 After Comb Filter Normalized on Old Sampling rate

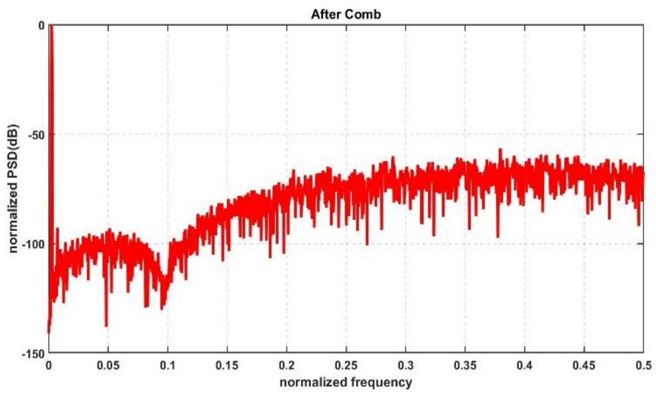


Figure 47 After Comb Filter Normalized on New Sampling rate

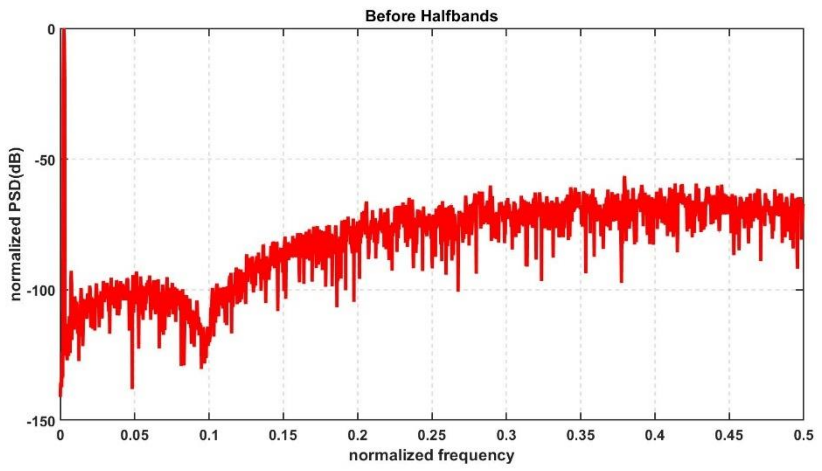


Figure 48 Before the Two Halfbands

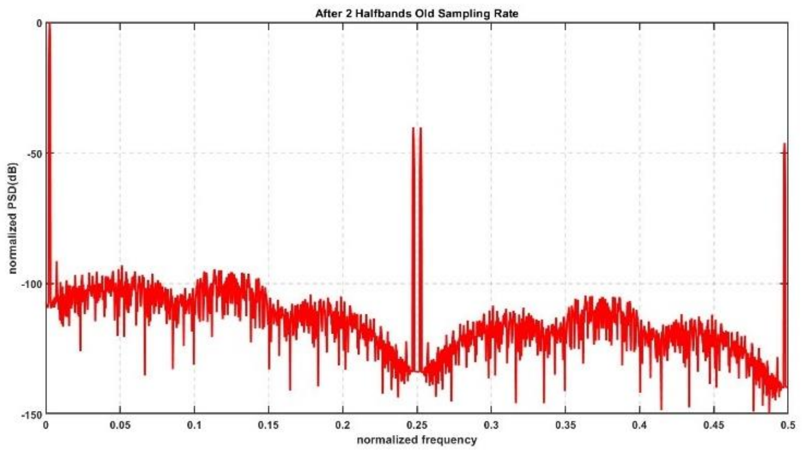


Figure 49 After the Two Half bands Normalized on Old Sampling Rate

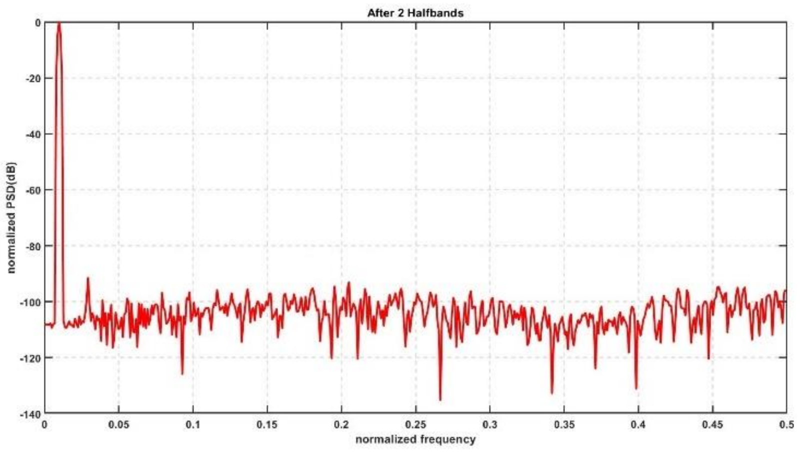


Figure 50 After the Two Half bands Normalized on New Sampling rate

3.4. Final System Architecture

The system consists of four elements each element has two phase shifted outputs I'/Q' the four elements' outputs will be summed together to generate a final I/Q that is constructively interfered at a certain incident angle and destructively interfered at other angles. Then decimated to generate the digitalized I/Q which is ready for further DSP.

3.4.2 Four Elements Single Beam

Full system diagram is shown in figure 51, where the system receives 1-bitstream from the sigma-delta ADC with sampling rate four times IF. Interleaver which generates I/Q components of the signal at half sampling rate and still in the if stage. DDC which works entirely at half of sampling rate and multiplies the two components I/Q by the IF frequency without increasing number of bits of the bitstream received from the sigma-delta ADC. CWM and adder to add phase on each element resulting in coherent signals which adds constructively in the adder achieving the beamforming action. Last stage of our system is the decimation where number of bits increases by 32 bit and down sampled to the signal frequency by eliminating the oversampling ratio completely.

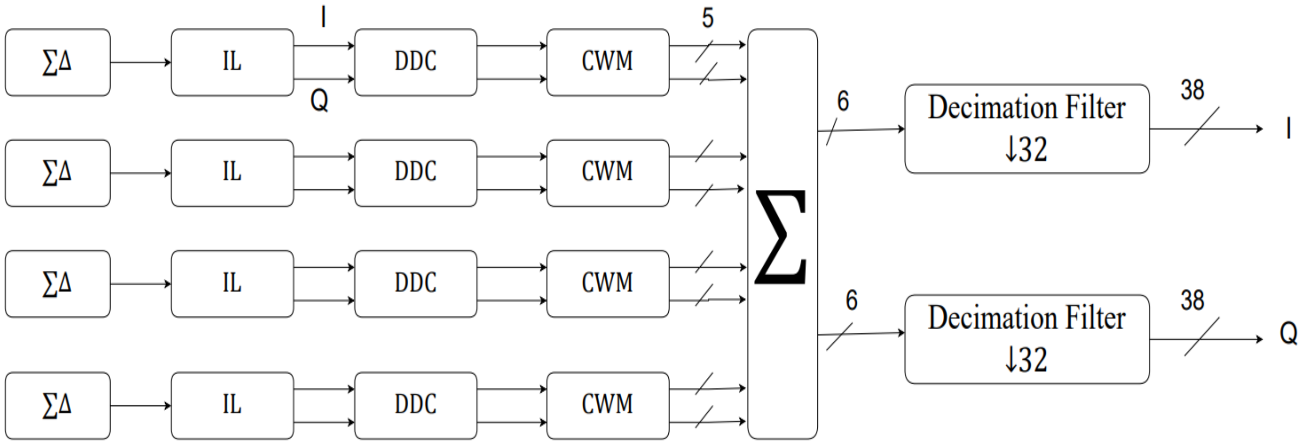


Figure 51 Full System Block Diagram

3.4.3 Four Elements Double Beam

One of the most important features in DPF is muti-beam generation with any SNR penalty. As shown in figure 52 the repetition of hardware is need for each beam starting by the CWM till the output as it is the part differs when steering put IL and DDC are not affected by the angle of steering. The function of IL and DDC is the same for every beam.

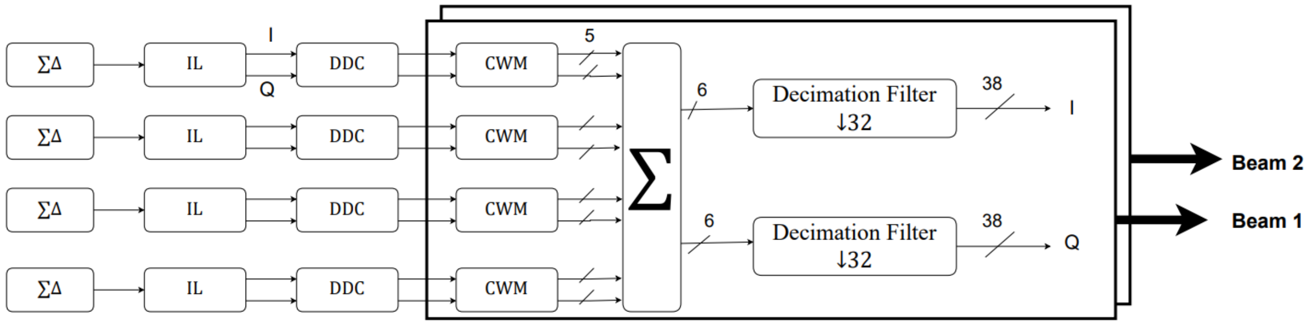


Figure 52 System Diagram For 2 Beams

3.5. Results

To test the validity of our system we need to test the effect of each parameter on SNR, also we need to test the beam steering for a single beam which shows spatial multiplexing done by the system and lastly the multi-beam as it is very important feature in the DBF.

3.5.1 System Dynamic Range

Input amplitude is very important parameter, showing the dynamic range of the system after beamforming and decimation stages as shown in following figures steering at 70°.

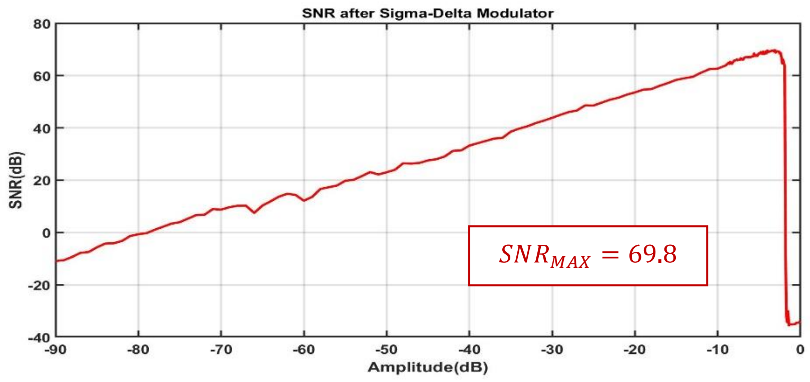


Figure 55 SNR vs Amp After Sigma-Delta

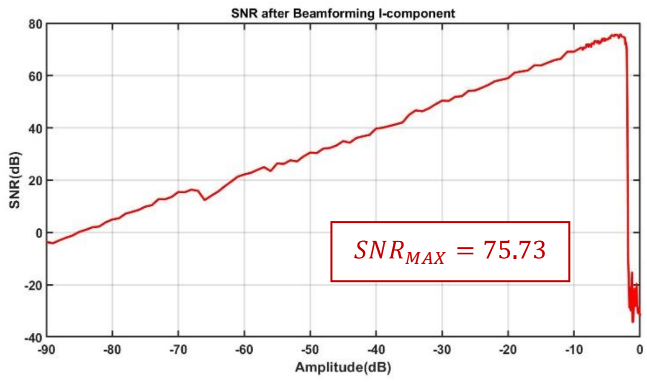


Figure 54 SNR vs Amp After BF I-Comp

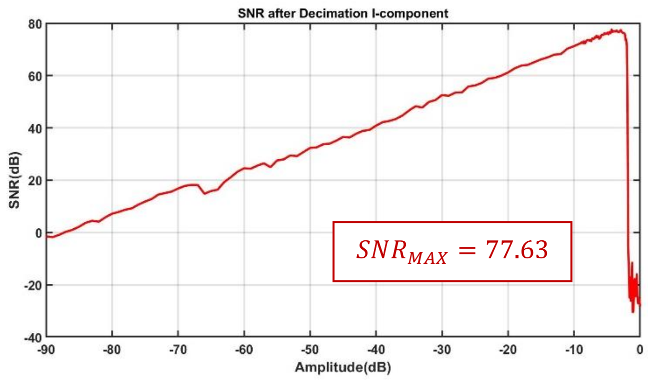


Figure 53 SNR vs Amp After Decimation I-Comp

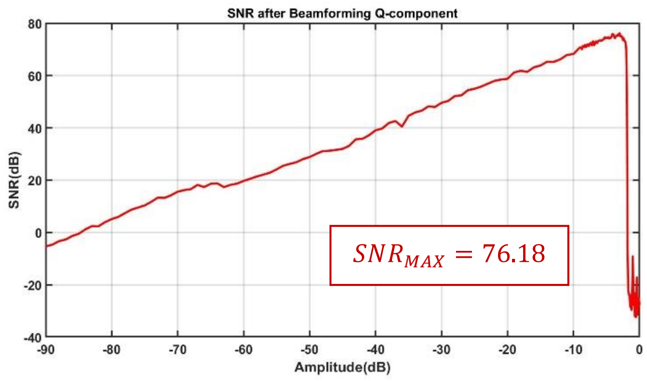


Figure 56 SNR vs Amp After BF Q-Comp

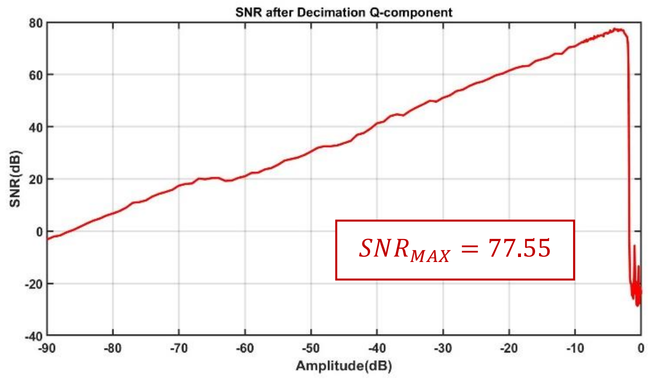


Figure 57 SNR vs Amp After Decimation Q-Comp

3.5.2 Spectrum After Each Stage

Testing the validity of the spectrum after each stage, steering at 70° and Amplitude equals to -4 dB.

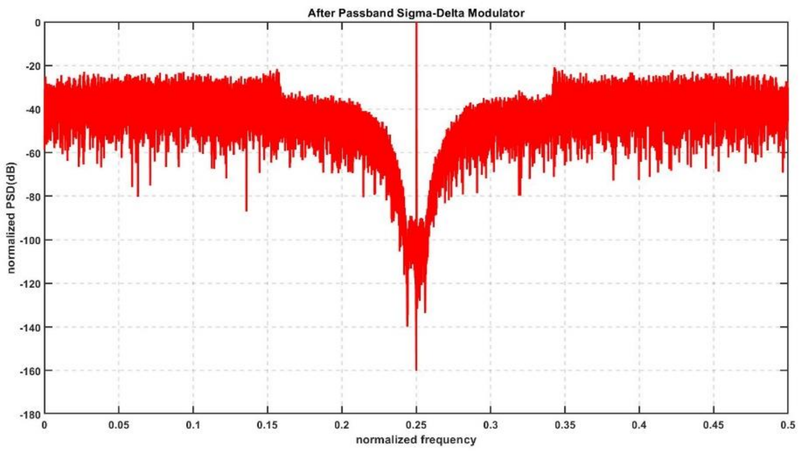


Figure 61 Spectrum After Sigma-Delta

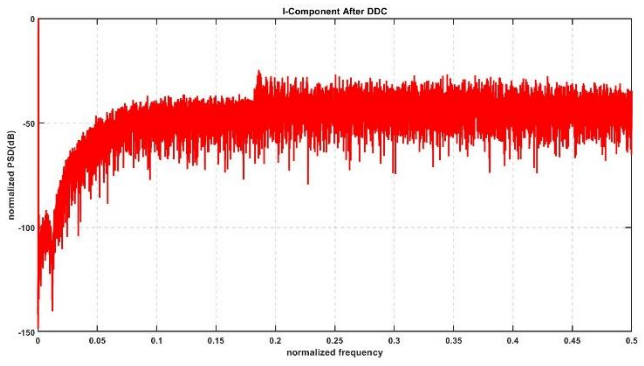


Figure 59 Spectrum After DDC I-Comp

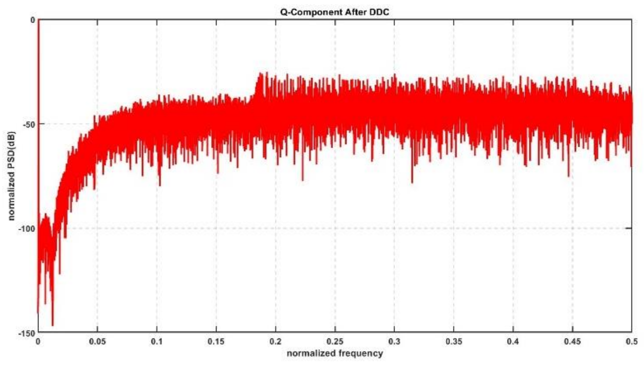


Figure 62 Spectrum After DDC Q-Comp

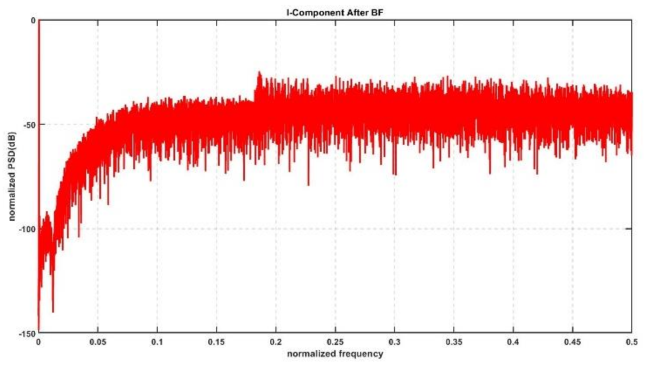


Figure 60 Spectrum After BF I-Comp

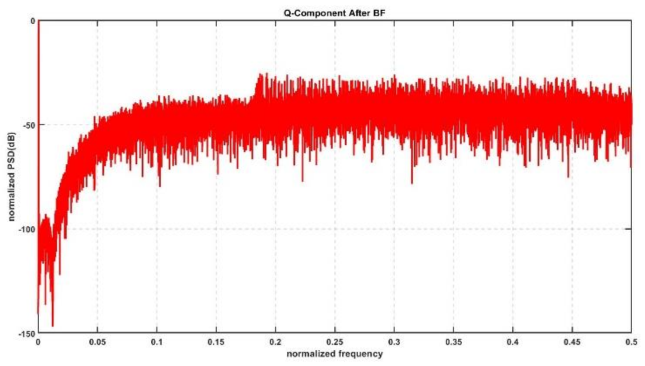


Figure 58 Spectrum After BF Q-Comp

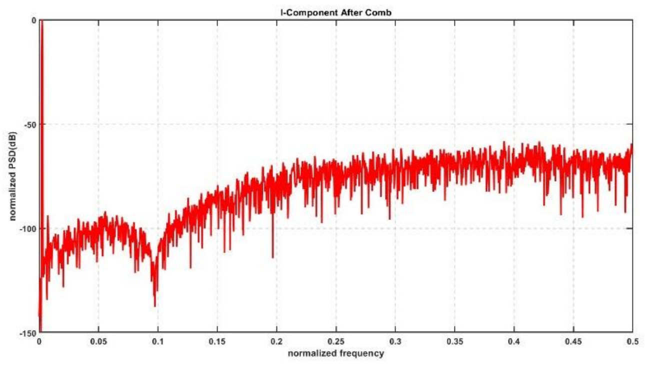


Figure 66 Spectrum After Comb I-Comp

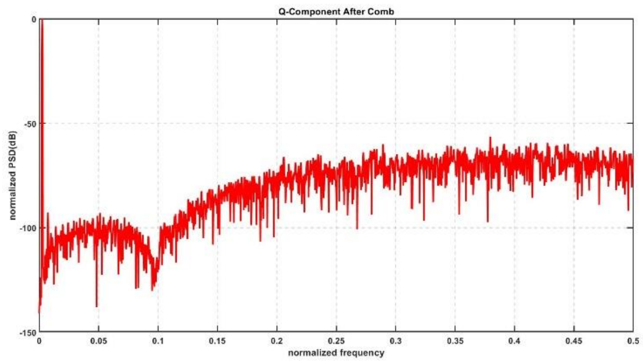


Figure 65 Spectrum After Comb Q-Comp

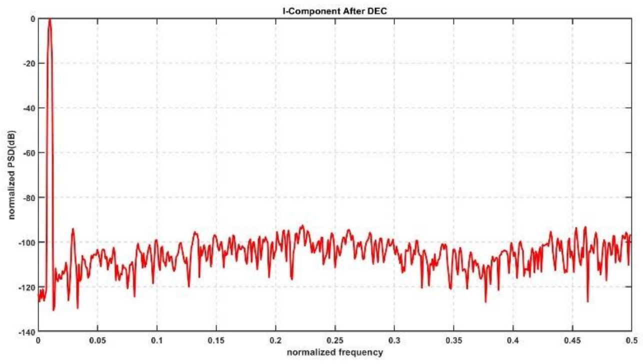


Figure 64 Spectrum After Decimation I-Comp

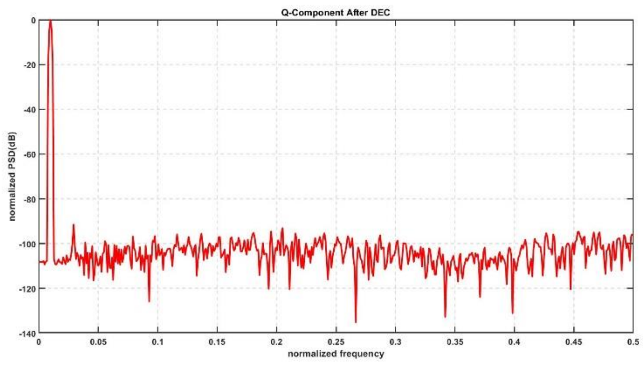


Figure 63 Spectrum After Decimation Q-Comp

3.5.3 Steering Effect on SNR

SNR vs Steering Angle with step 0.5 degrees and input amplitude -4 dB.

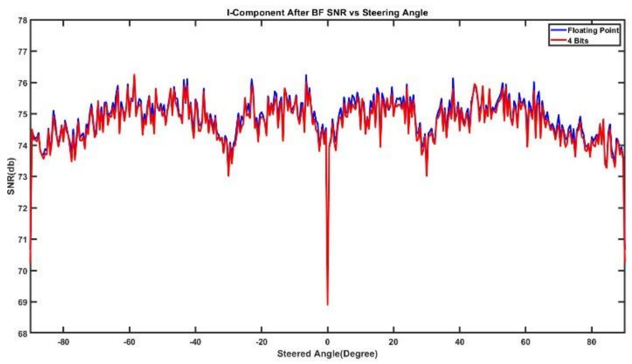


Figure 67 SNR vs Steering Angle After BF I-Comp

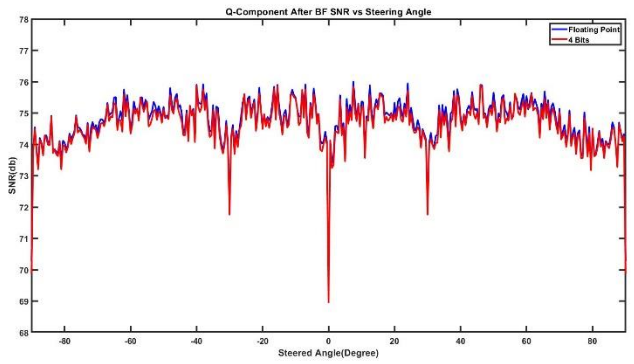


Figure 68 SNR vs Steering Angle After BF Q-Comp

$$SNR_{avg,Floating\ P} = 74.83 \text{ dB}$$
$$SNR_{avg,4-bits} = 74.70\ dB$$

$$SNR_{avg,Floating\ P} = 74.75 \text{ dB}$$
$$SNR_{avg,4-bits} = 74.62\ dB$$

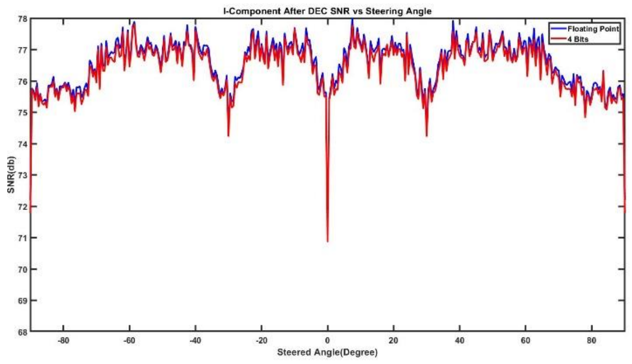


Figure 70 SNR vs Steering Angle After DEC I-Comp

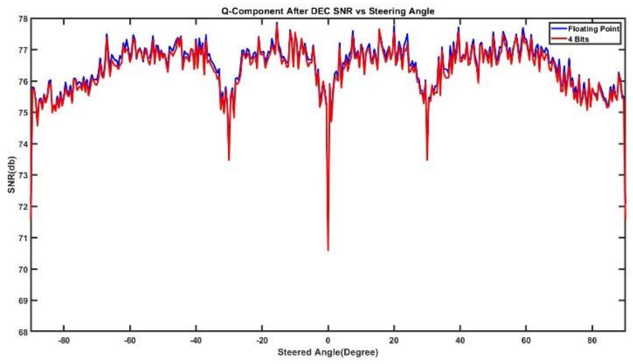


Figure 69 SNR vs Steering Angle After DEC Q-Comp

$$SNR_{avg,Floating\ P} = 76.54 \text{ dB}$$
$$SNR_{avg,4-bits} = 76.39\ dB$$

$$SNR_{avg,Floating\ P} = 76.45 \text{ dB}$$
$$SNR_{avg,4-bits} = 76.32\ dB$$

The downfall at the zero steering angle is because our system uses correlated noise, which leads to have identical signal and noise at the zero-degree steering resulting in no change in SNR.

3.5.4 Baseband Frequency Effect on SNR

SNR vs Freq steered at 70 degrees and amp -4 dB.

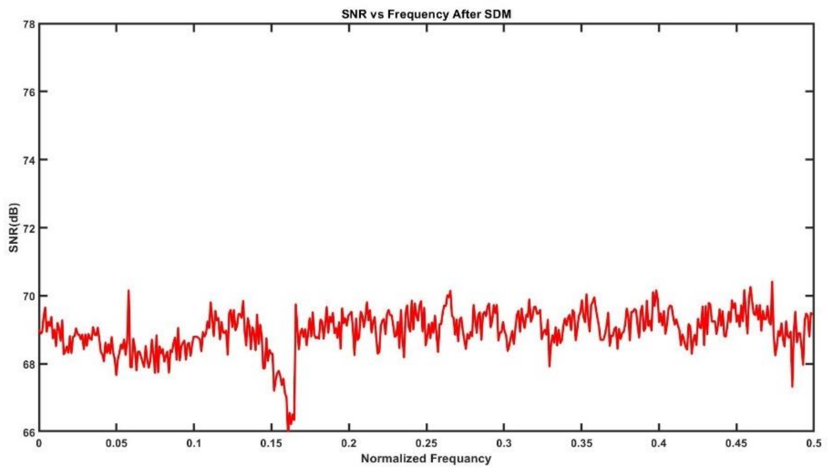


Figure 71 SNR vs Frequency After Sigma-Delta

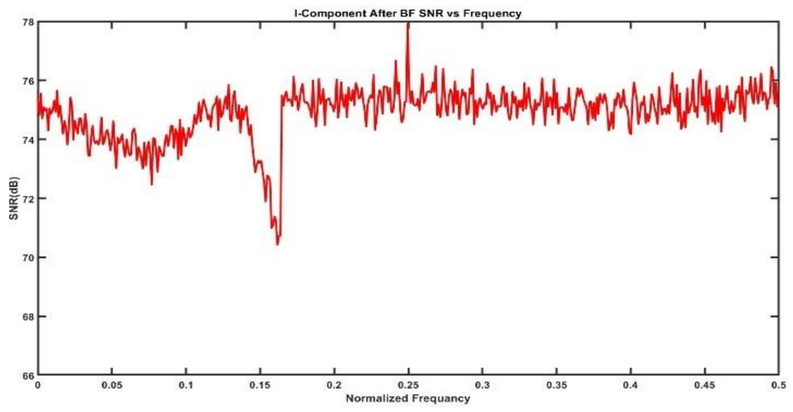


Figure 72 SNR vs Frequency After BF I-Comp

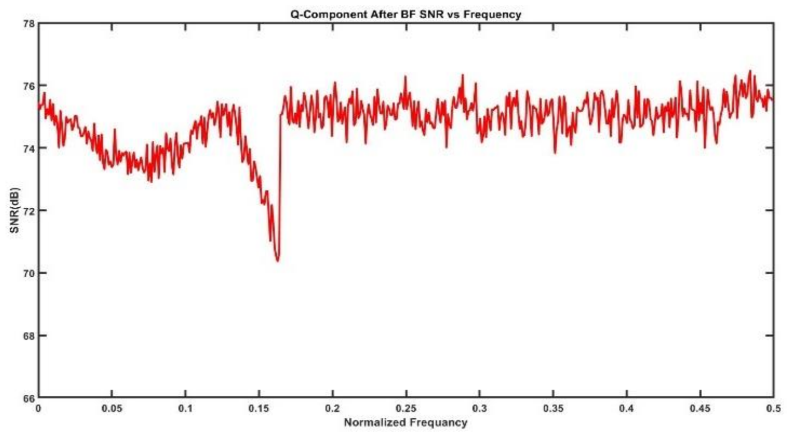


Figure 73 SNR vs Frequency After BF Q-Comp

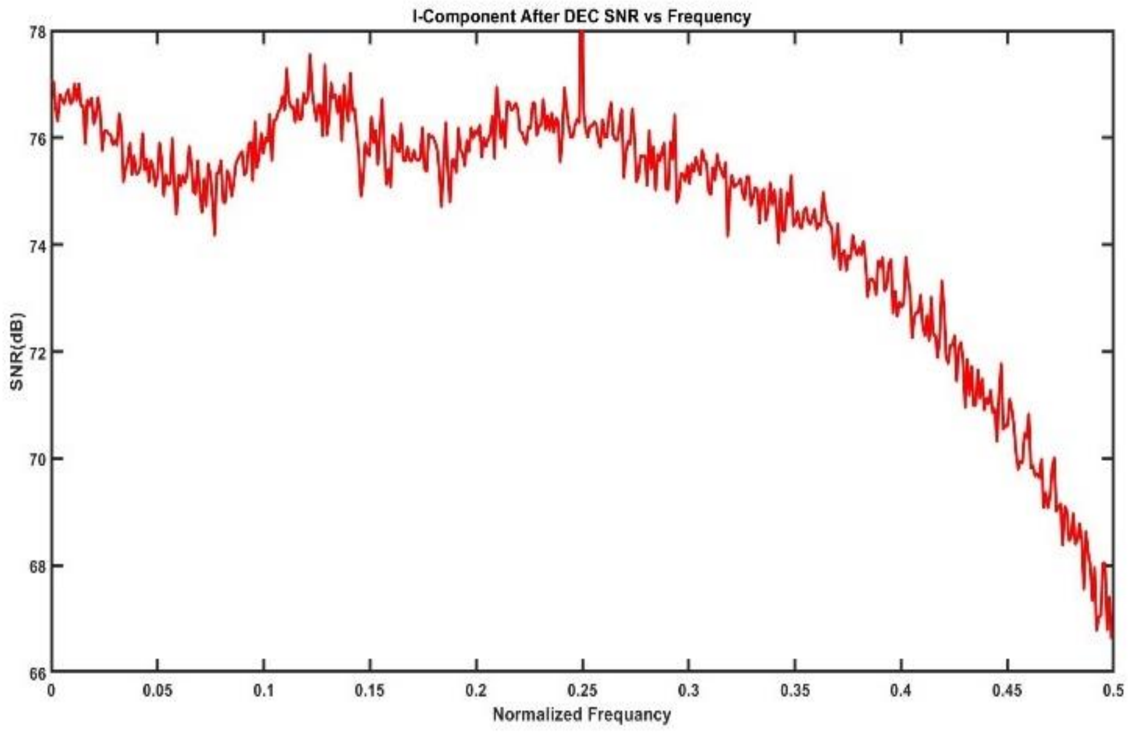


Figure 74 SNR vs Frequency After DEC I-Comp

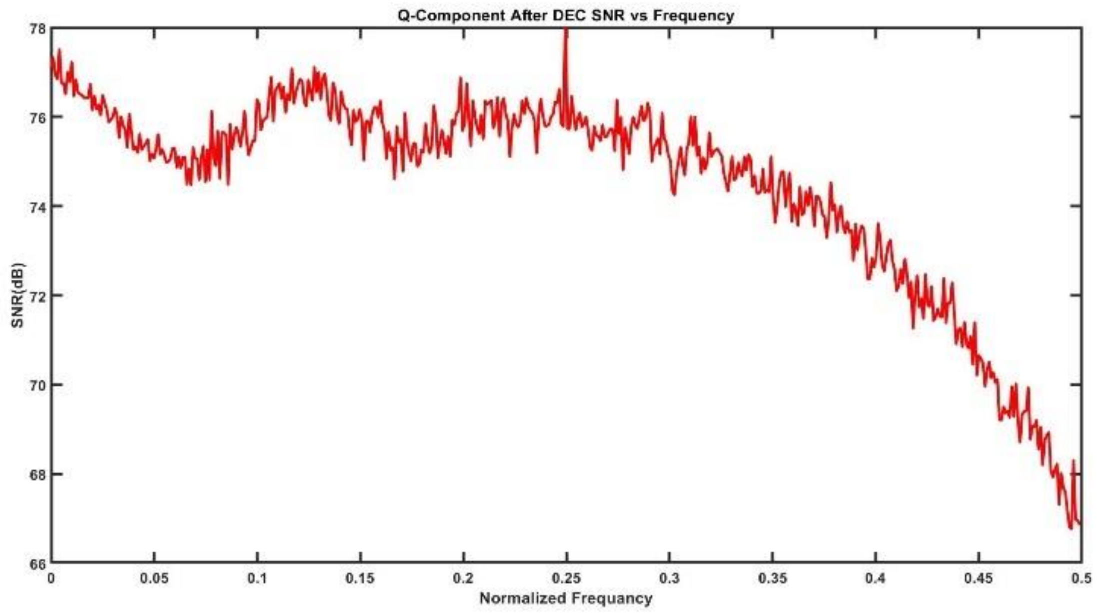


Figure 75 SNR vs Frequency After DEC Q-Comp

3.5.4 Single Beam Steering

As shown in figure 76, steering of the beam for 4-Elemnts array, beam each 10° with step 1°, with side lobe level lower than $-10\ dB$ which happens at $\pm\ 90°$.

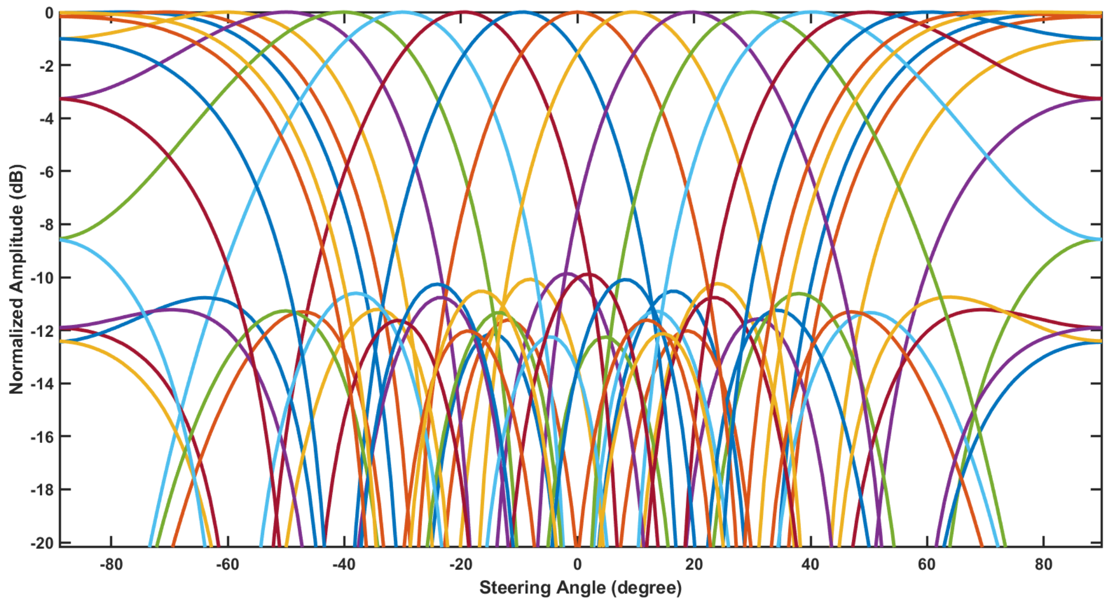


Figure 76 Beam Steering For 4-Elements DBF System

3.5.5 Double Beam

In the shown figures 77 and 78, we can see the double beam of the system steered at -20° and 30°.

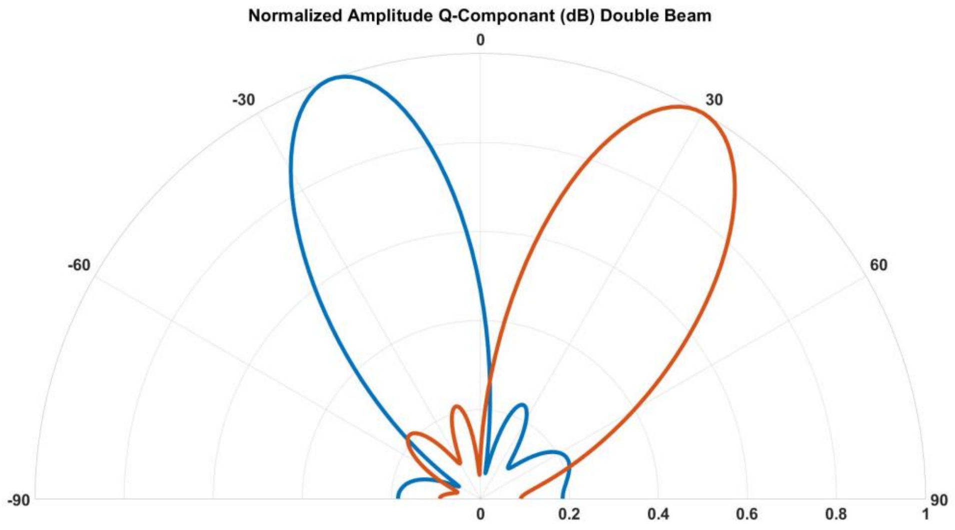


Figure 77 Double Beam For Q-Component

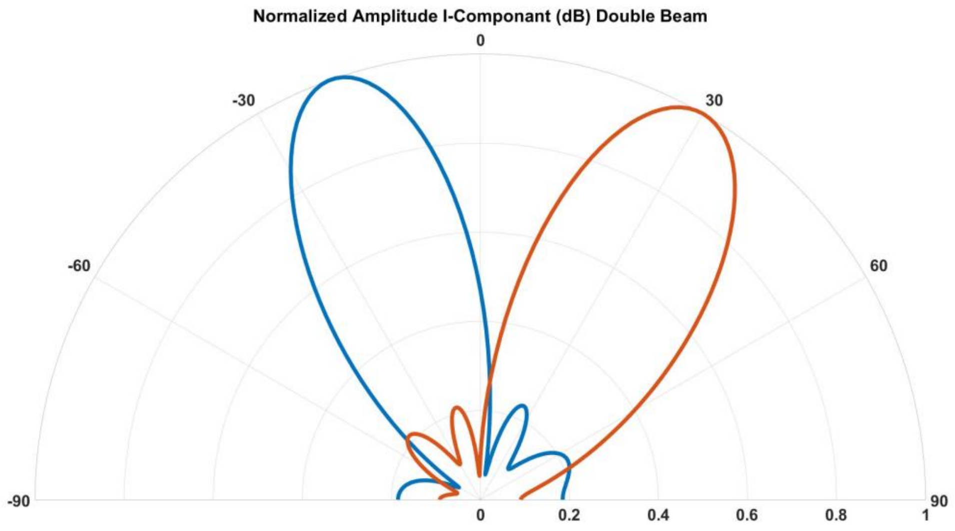


Figure 78 Double Beam For I-Component

Chapter 4: RTL

4.1 Full System

RTL design is to start to put the architectural of the design to meet the system requirements (Functional, interfacing with other blocks, operational speed, power consumption, Area) and then the architectural micro designs are modelled in a hardware Description Language like Verilog/VHDL using synthesizable constructs of the language.

We used Verilog in modelling our design, all the system modules are designed and written fully from scratch with their test benches.

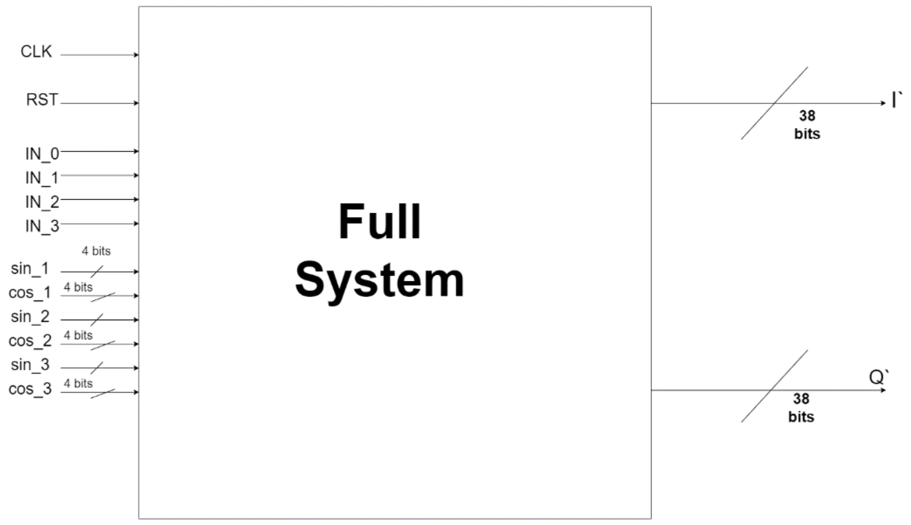


Figure 79 Full system

Inputs to the module:

1. CLK: The main system clock which will be divided to different clocks to serve other internal modules
2. RST: The main system reset which is responsible for resetting the whole system
3. IN: we have four one-bit inputs to the system because we have four elements, and those inputs are from four one-bit Sigma-Delta ADCs
4. Sin & Cos: we have 3 pairs each with 4-bits and they represent the weights of sine and cosine in the CWM (complex weight multiplier)

Outputs of the module:

1. I: is the 38-bits decimated in phase signal.
2. Q: is the 38-bits decimated quadrature signal.

4.2 Clock Divider

Now we are going to talk about the submodules of our system.

Starting by Clock divider:

Behavioral Description:

one of the main blocks is the clock divider which divide the clock into multiple clocks to serve other modules.

The clock divider we designed is Asynchronous clock divider which gives us a couple of advantages:

1. Each stage runs at a lower frequency, resulting in lower power

2.reduced high frequency clock loading

But it also has the disadvantage of Jitter accumulation

Block diagram:

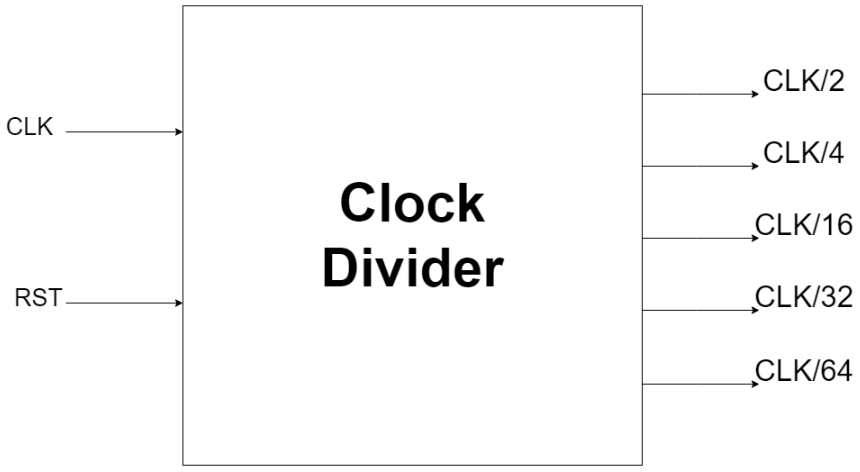


Figure 80 Clock Divider

Inputs:

1. CLK: the main clock to the system which the clock divider generated the other clocks we need
2. RST: the main reset signal

Outputs:

1. CLK/2
2. CLK/4
3. CLK/16
4. CLK/32
5. CLK/64

Behavioral Simulations:

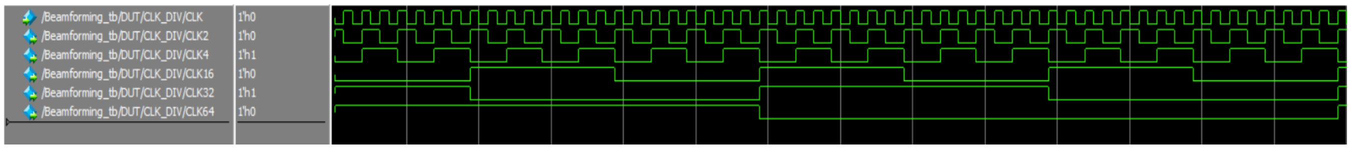

/Beamforming_tb/DUT/CLK_DIV/CLK 1'h0
/Beamforming_tb/DUT/CLK_DIV/CLK2 1'h0
/Beamforming_tb/DUT/CLK_DIV/CLK4 1'h1
/Beamforming_tb/DUT/CLK_DIV/CLK16 1'h0
/Beamforming_tb/DUT/CLK_DIV/CLK32 1'h1
/Beamforming_tb/DUT/CLK_DIV/CLK64 1'h0

Figure 81 Clock Divider wave forms

So, the Clocks are generated from the main Clock, and that's obvious as in figure 81 as one period of CLK/2 has 2 periods of the main clock and CLK/4 has 4 periods of the main clock and so on.

4.3 Interleaver

Behavioral Description:

The interleaver main purpose is to divide the input signal into a pair of signals, one called in-phase and the other is called quadrature.

This is achieved by down sampling the signal by 2 to generate the first output, and the second output is generated by delaying the signal by one period then down sample it.

Block diagram:

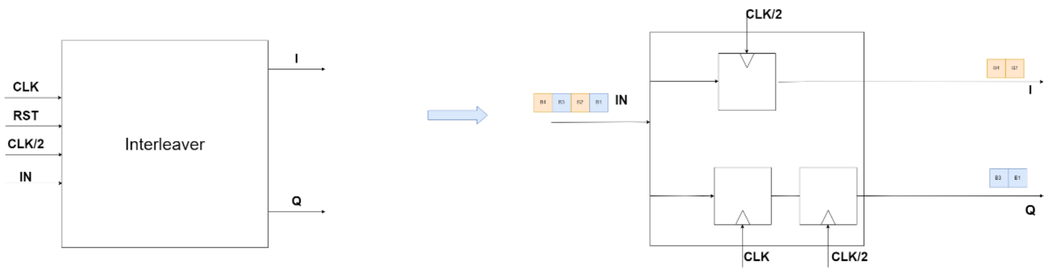


Figure 82 Interleaver

Inputs:

1. CLK: The main system clock which is used by the register that delay the input by one period
2. RST: the main reset signal
3. IN: the one-bit input signal that will be divided
4. CLK/2: the version of the main clock that is generated by the clock divider and is used by the registers to down sample the input signal

Outputs:

1. I: The one-bit in phase output
2. Q: The one-bit quadrature output

Behavioral simulation:

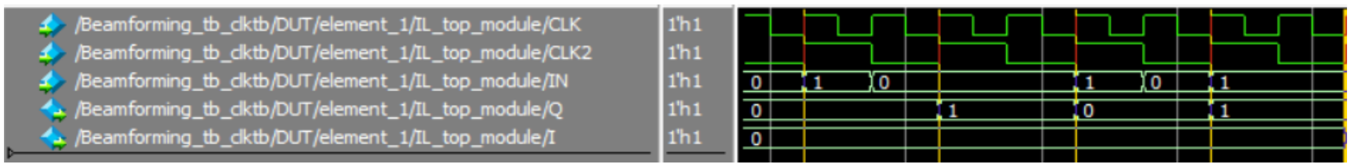

/Beamforming_tb_clktb/DUT/element_1/IL_top_module/CLK
/Beamforming_tb_clktb/DUT/element_1/IL_top_module/CLK2
/Beamforming_tb_clktb/DUT/element_1/IL_top_module/IN
/Beamforming_tb_clktb/DUT/element_1/IL_top_module/Q
/Beamforming_tb_clktb/DUT/element_1/IL_top_module/I
1'h1
1'h1
1'h1
1'h1
1'h1

Figure 83 Interleaver waveforms

IN= {1 0 0 0 1 0}

As we could see in figure 83 the interleaver works as we wanted, if we examined the first six inputs, we could see that they are distributed between the two outputs where the ‘Q’ signal takes the odd numbered bits and the ‘I’ signal takes the even ones.

4.4 DDC

Behavioral Description:

The DDC purpose is to bring the signal to the baseband, and this is done by multiplying the signal with the IF which is the clk/4 we generated in the clock divider.

Looking at table 4 and table 5 we realized that the multiplication can be avoided using an XNOR gate instead as our signal is only one bit so we can represent -1 as 0 and that’s what we used in our design.

LO	I	I_DDC
-1	-1	1
-1	1	-1
1	-1	-1
1	1	1

Table 4 System Truth table

LO	I	I_DDC
0	0	1
0	1	0
1	0	0
1	1	1

Table 5 XNOR Truth table

Block diagram:

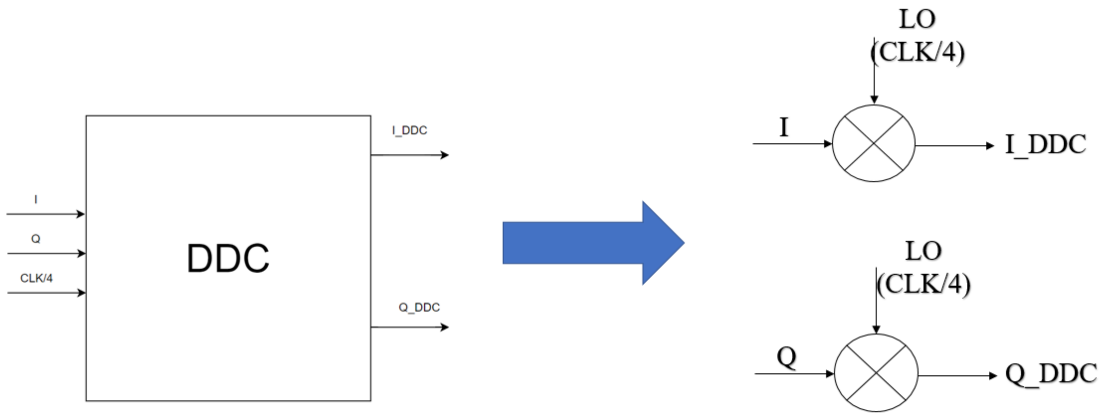


Figure 84 DDC

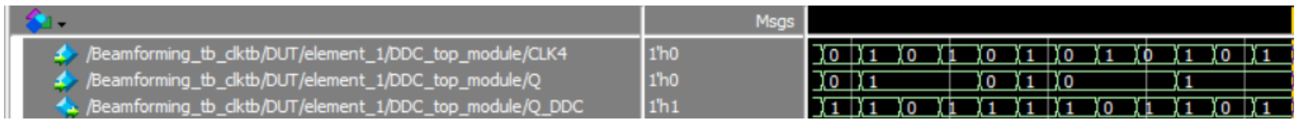
Inputs:

1. I: one-bit in-phase signal which we generated from the Interleaver
2. Q:one-bit quadrature signal which we generated from the interleaver
3. CLK/4: it's the frequency divided by 4 which is the IF we generated from the clock divider

Outputs:

1. I_DDC: one-bit in-phase signal in the baseband
2. Q_DDC: one-bit quadrature signal in the baseband

Behavioral simulation:



```
                                                      Msgs
/Beamforming_tb_clktb/DUT/element_1/DDC_top_module/CLK4    1'h0   0 1 0 1 0 1 0 1 0 1 0 1
/Beamforming_tb_clktb/DUT/element_1/DDC_top_module/Q       1'h0   0 1     0 1 0     1
/Beamforming_tb_clktb/DUT/element_1/DDC_top_module/Q_DDC   1'h1   1 1 0 1 1 1 1 0 1 1 0 1
```

Figure 85 DDC wave forms

As we can see from figure 85 the results follow table 5 of and as the function of the DDC is to bring the signal to the baseband we draw the spectrums (figure 86, 87) to check that

The spectrum before the DDC:

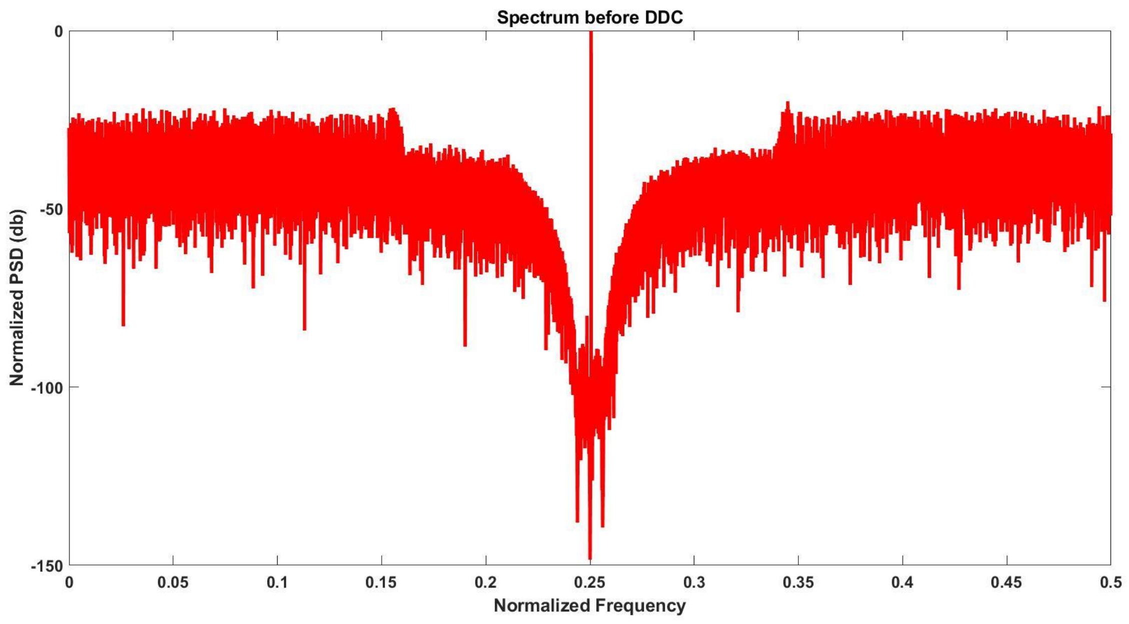


Figure 86 Spectrum before DDC

The spectrum after the DDC:

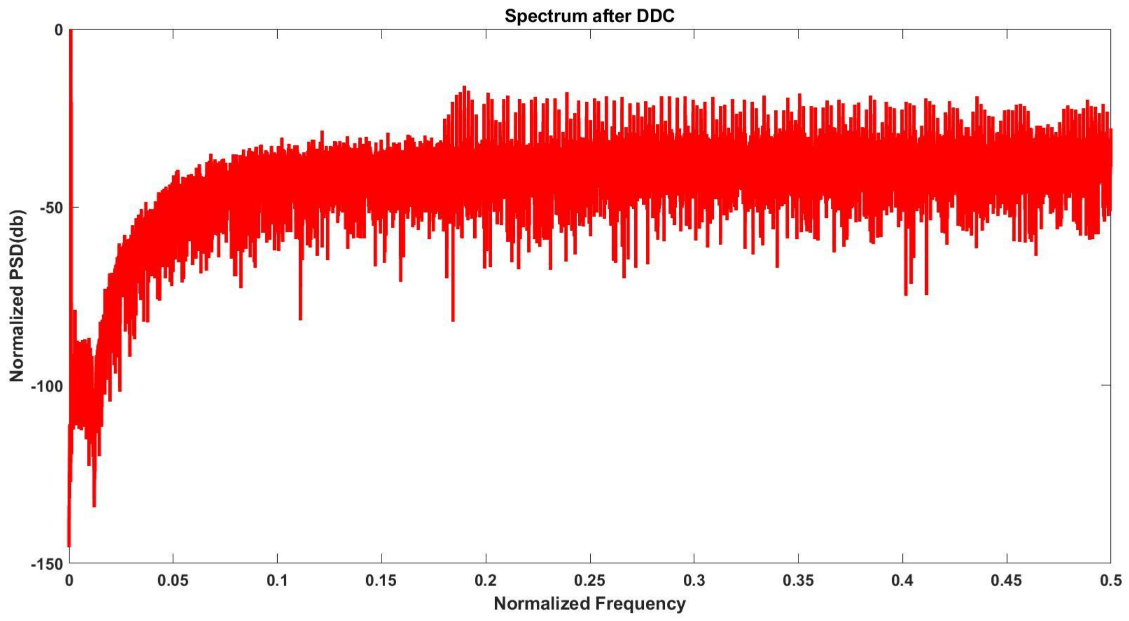


Figure 87 Spectrum after DDC

Now we can see that the signal is now at the baseband as we desired after DDC

4.5 CWM

Behavioral Description:

In CWM we want to achieve those two equations:

$$I' = I\cos(\theta) + Q\sin(\theta)$$

$$Q' = Q\cos(\theta) - I\sin(\theta)$$

Where θ is the phase shift that we need for each element.

	Phase shift θ	$\cos(\theta)$	$\sin(\theta)$
Element 0	0	1	0
Element 1	$\pi\sin(\psi)$	$\cos(\pi\sin(\psi))$	$\sin(\pi\sin(\psi))$
Element 2	$2\pi\sin(\psi)$	$\cos(2\pi\sin(\psi))$	$\sin(2\pi\sin(\psi))$
Element 3	$3\pi\sin(\psi)$	$\cos(3\pi\sin(\psi))$	$\sin(3\pi\sin(\psi))$

Table 6 Weights table

From table (6) we deduce that the needed inputs for each element is a pair of sin & cos and for element 0 the equation will be simplified then we can see that we will have 3 sin & cos pair as inputs for the whole system and those pair what we are going to use in the CWM to achieve the equations.

The equations are achieved by multiplication and additions and since we multiply by the output of the DDC which is a one-bit signal we can use a multiplexer instead of a multiplier which will decrease in area and power.

Block diagram: The CWM main components are 4 multiplexers and 2 adders

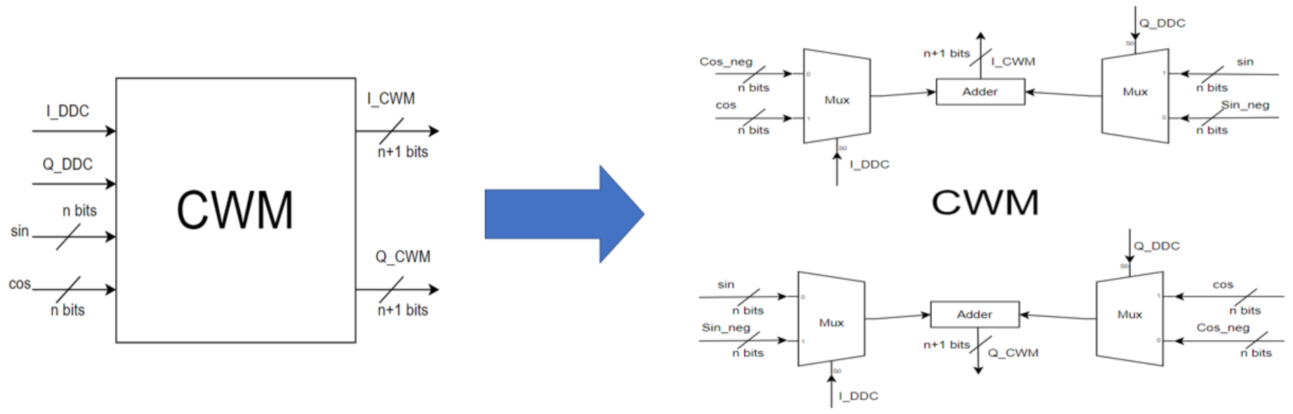


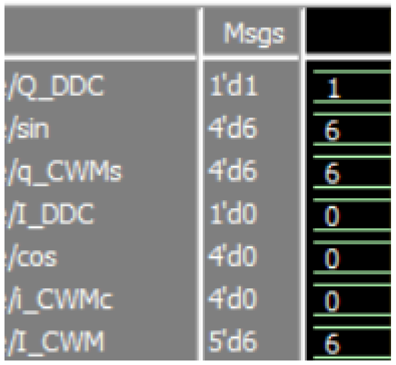
Figure 88 CWM

Inputs:

1. I_DDC: one-bit in-phase DDC output
2. Q_DDC: one-bit quadrature DDC output
3. Sin: 4-bits weight of the sin in the CWM
4. Cos: 4-bits weight of the cos in the CWM

Outputs:

1. I_CWM: a 5-bits in-phase CWM output which achieve the wanted equation.
2. Q_CWM: a 5-bits quadrature CWM output which achieve the wanted equation.



```
        Msgs
/Q_DDC  1'd1   1
/sin    4'd6   6
/q_CWMs 4'd6   6
/I_DDC  1'd0   0
/cos    4'd0   0
/i_CWMc 4'd0   0
/I_CWM  5'd6   6
```

Figure 89 CWM waveform

Behavioral simulation:

To check that the module is working as intended we will see if the output of the simulation achieve the wanted equations.

We are going to see if the first equation is achieved,

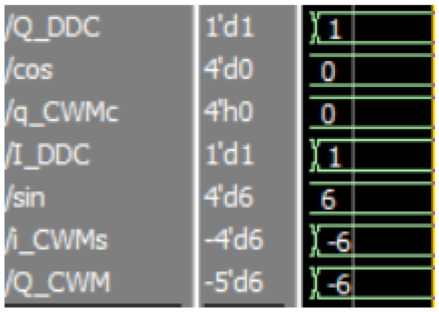
$$I' = I\cos(\theta) + Q\sin(\theta)$$

From the simulation(figure 89) when 'Q' is 1, 'sin' is 6 then Q multiplied by sin is equal 6, and 'I' is 0, 'Cos' is 0 then I multiplied by cos is 0, since $Q \times sin$ is equal 6 and $I \times cos$ is equal 0 then $I' = I\cos(\theta) + Q\sin(\theta)$=6 as we can see in figure 89.

Where Q_DDC is Q, q_CWMs is Q*sin, I_DDC is I, i_CWMc is ICOS and I_CWM is I`

After making sure that the first equation is achieved let's check the second equation,

$$Q' = Q\cos(\theta) - I\sin(\theta)$$



```
/Q_DDC  1'd1   1
/cos    4'd0   0
/q_CWMc 4'h0   0
/I_DDC  1'd1   1
/sin    4'd6   6
/i_CWMs -4'd6  -6
/Q_CWM  -5'd6  -6
```

Figure 90 CWM waveform

From the simulation(figure 90) when 'Q' is 1, 'cos' is 0 the Q multiplied by cos is equal 0, and 'I' is 1, 'sin' is 6 then '-I' multiplied by 'sin' is -6, since $Q \times cos$ is equal 0 and $-I \times sin$ is equal -6 then $Q' = Q\cos(\theta) - I\sin(\theta)$=-6 as we can see in figure 90.

Where Q_DDC is Q, q_CWMc is $Q \times cos$, I_DDC is I, i_CWMs is $I \times sin$ and Q_CWM is Q`

4.5 CWM Element Zero

Behavioral Description:

After we discussed all the modules that we used to create a single element for the beamformer but there is something we noticed that we decided to make use of, if we took a close look at table(7) we are going to notice that the element number 0 has no phase shift therefor it's weights are constants so instead of using the CWM we designed which had 4 multiplexers and 2 adders we decided to design another CWM for element 0 which will be more efficient in terms of area and power and since there are no adders in this version of CWM so the equation can be reduced to

$$I' = I$$

$$Q' = Q$$

	Phase shift θ	$\cos(\theta)$	$\sin(\theta)$
Element 0	0	1	0
Element 1	$\pi\sin(\psi)$	$\cos(\pi\sin(\psi))$	$\sin(\pi\sin(\psi))$
Element 2	$2\pi\sin(\psi)$	$\cos(2\pi\sin(\psi))$	$\sin(2\pi\sin(\psi))$
Element 3	$3\pi\sin(\psi)$	$\cos(3\pi\sin(\psi))$	$\sin(3\pi\sin(\psi))$

Table 7 CWM weights

Block diagram:

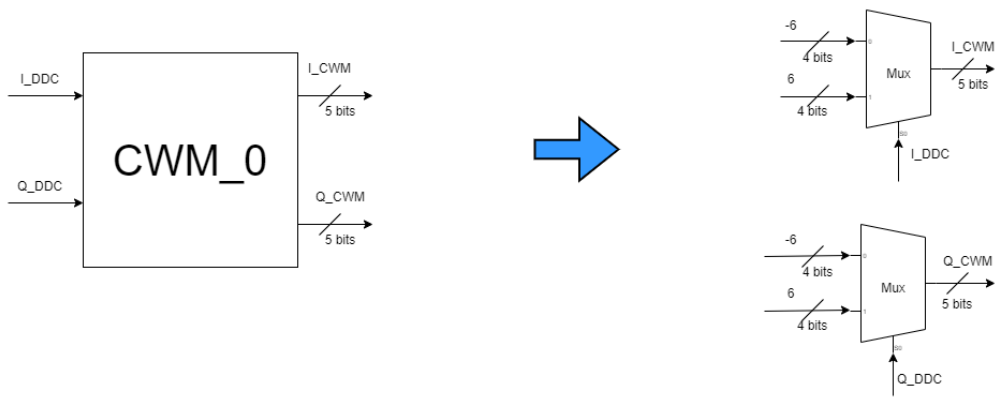


Figure 91 CWM_0

So, after the modifications the CWM now only contains two multiplexers

Inputs:

1. I_DDC: one-bit in-phase DDC output
2. Q_DDC: one-bit quadrature DDC output

Outputs:

1. I_CWM: 5-bits in-phase CWM output

2. Q_CWM: 5-bits quadrature CWM output

Behavioral simulation:

To check that the module is working as intended we will see if the simulation achieve the equations.

$$I' = I$$

$$Q' = Q$$

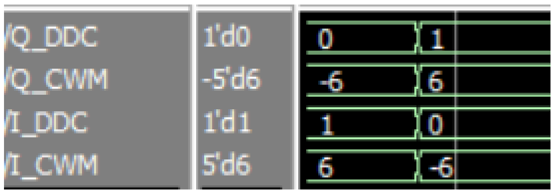


Figure 92 CWM_0 waveform

As we can see in the figure 92 when 'Q' is 0 'Q`' is -6 where '6' here is equal to '1' but scaled and when 'I' is 1 'I`' is 6 and vice versa.

Where Q_DDC is Q, Q_CWM is Q`, I_DDC is I and I_CWM is I`

After we discussed all the submodules now, we are ready to connect them together

Element_0:

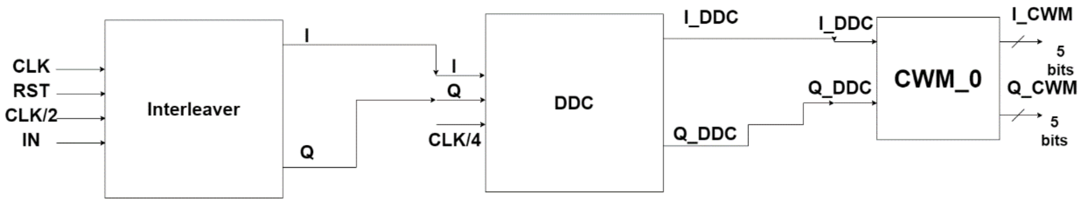


Figure 93 Element 0

Element:

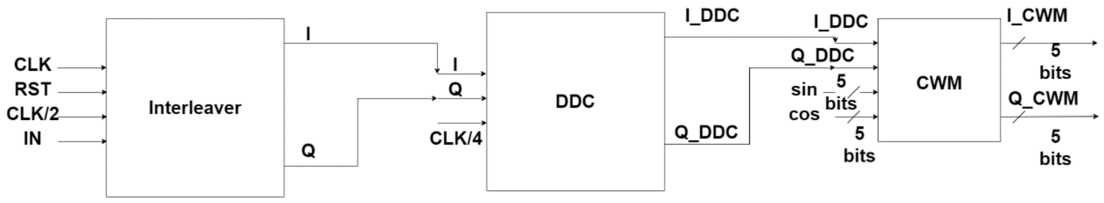


Figure 94 Element

4.6 Beamformer

Now after we assembled element_0 and element we are ready to create the 4-elements beamformer

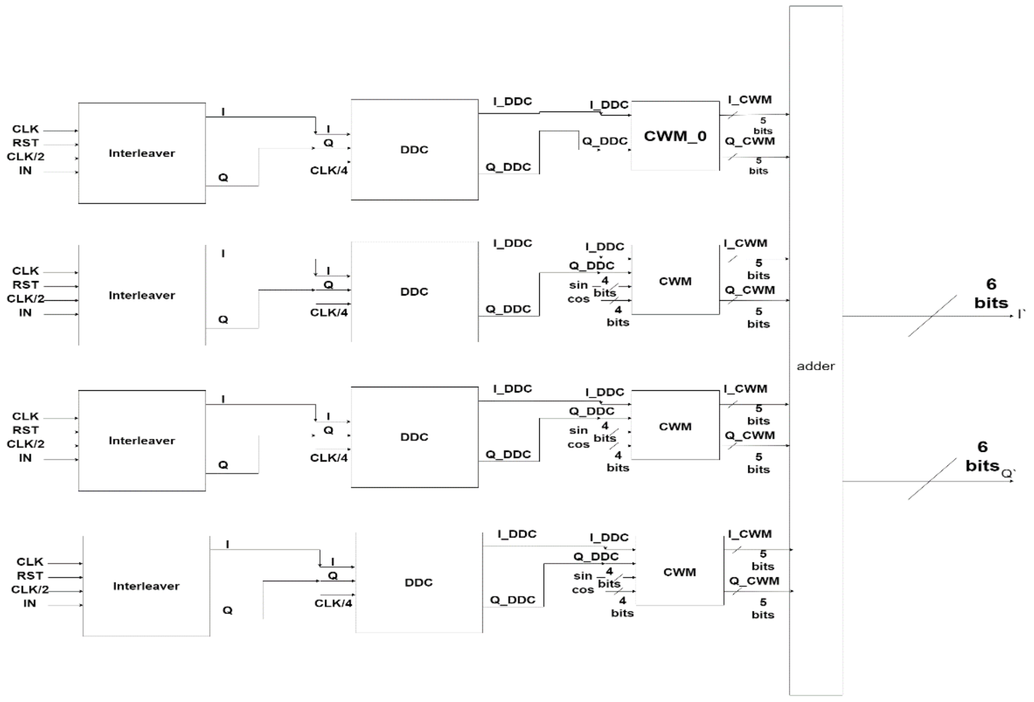


Figure 95 Beamformer

Spectrum after adding the 4 elements together (figure 96)

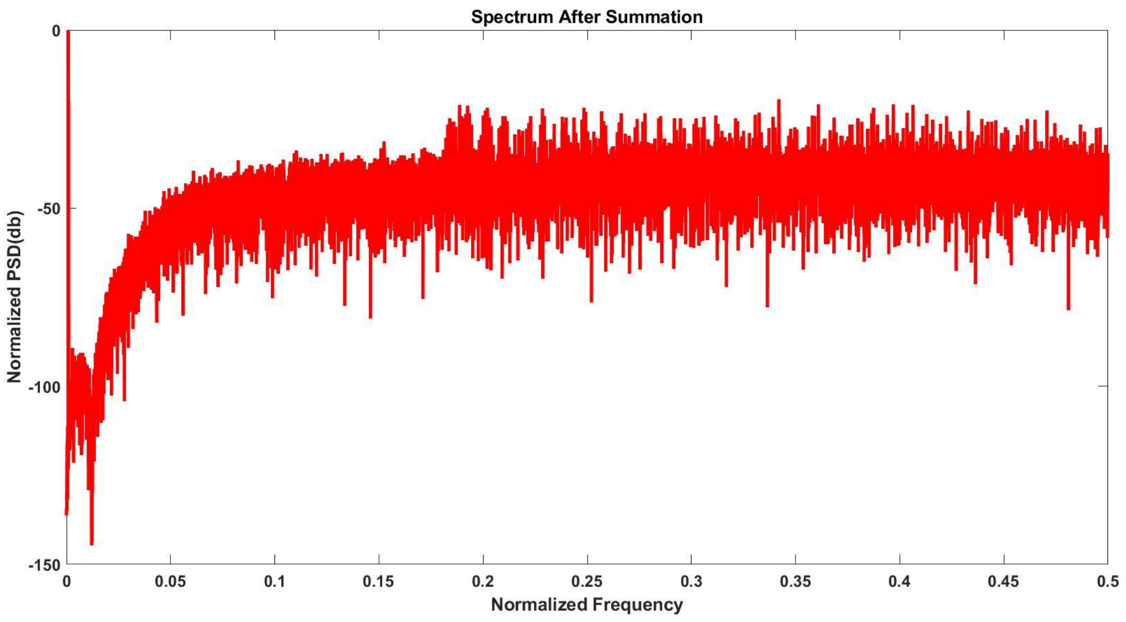


Figure 96 Spectrum after summation

4.7. Decimation filters

The decimation filter consists of three stages a comb filter and two half-band filters which down sample the frequency by 32, the comb filter reduces the sampling frequency by 8 and each half band divide the frequency by 2.

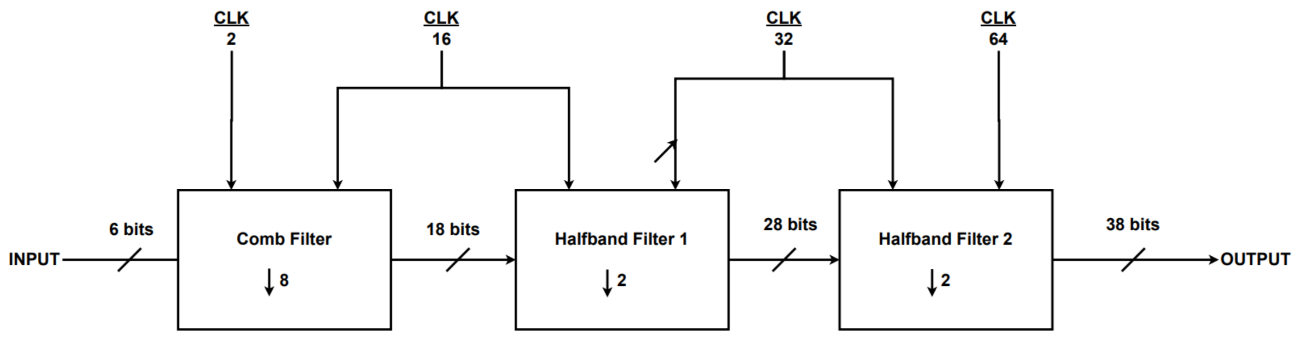


Figure 97 Decimation Diagram

As shown in figure 97 the decimation needs four clocks from the clock divider which are $\frac{CLK}{2}, \frac{CLK}{16}, \frac{CLK}{32}$ and $\frac{CLK}{64}$, the comb filter increases the number of bits by 12 bit and each Half-Band (HB) increases the number of bits by 10 bits.

4.7.1. Comb filter

The Comb filter has two input clocks which are CLK/2 and CLK/16 with an asynchronous Reset.

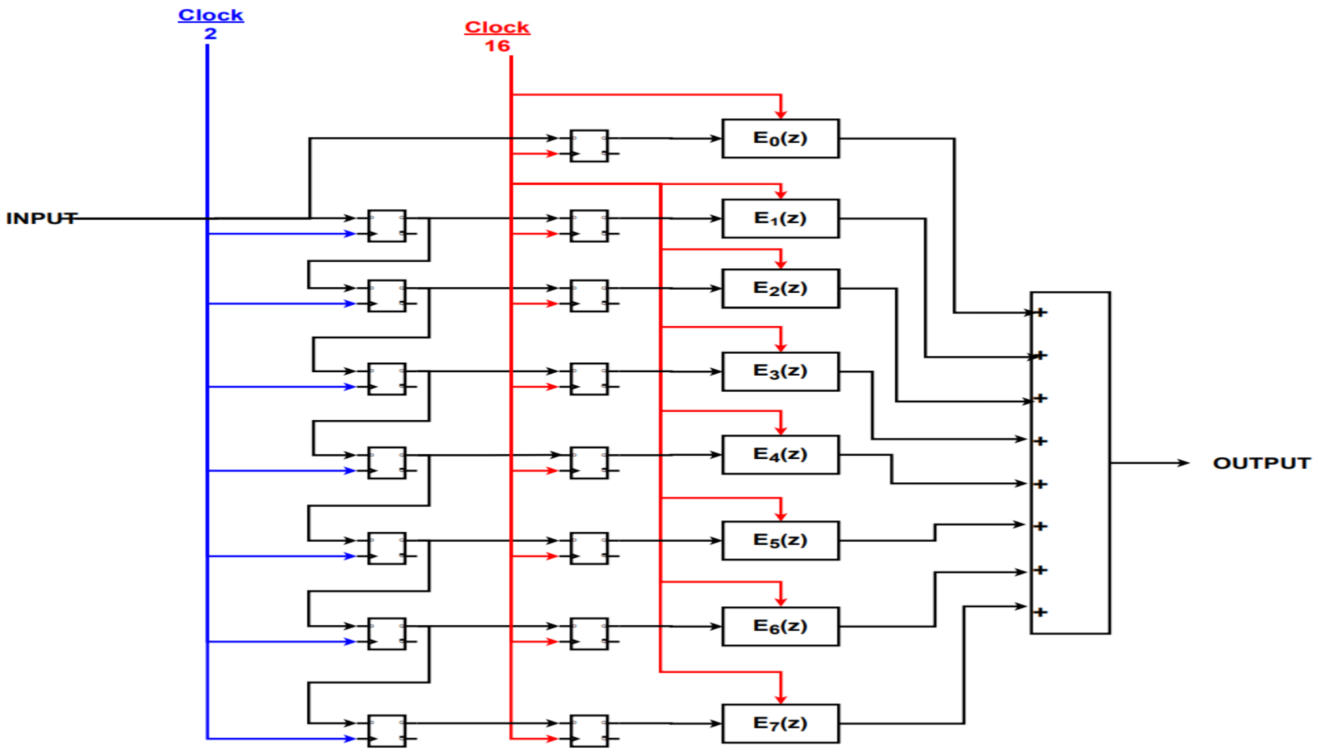


Figure 98 Comb filter diagram

The diagram shown in figure 99 shows the implementation of the comb filter, where E (z) is FIR filter.

The system has 8 FIR filters, these filters have symmetric coefficients so we can get advantage of this by adding each two nodes with the same coefficients together before multiplication to reduce the number of multipliers to almost half.

The final comb design with Polyphase representation will have 7 registers operating at Clock/2 and 29 register operating at CLK/16, with a set of adders and multipliers. Some multipliers can be implemented using shifters only.

This Comb filter increases the number of bits by 12-bits so in our design its input is represented in 6 bits and its output in 18 bits.

4.7.2. Half-band filter

Each HB has two input clock; CLK/16 and CLK/32 for the first HB, CLK/32 and CLK/64 for the second HB with an asynchronous Reset.

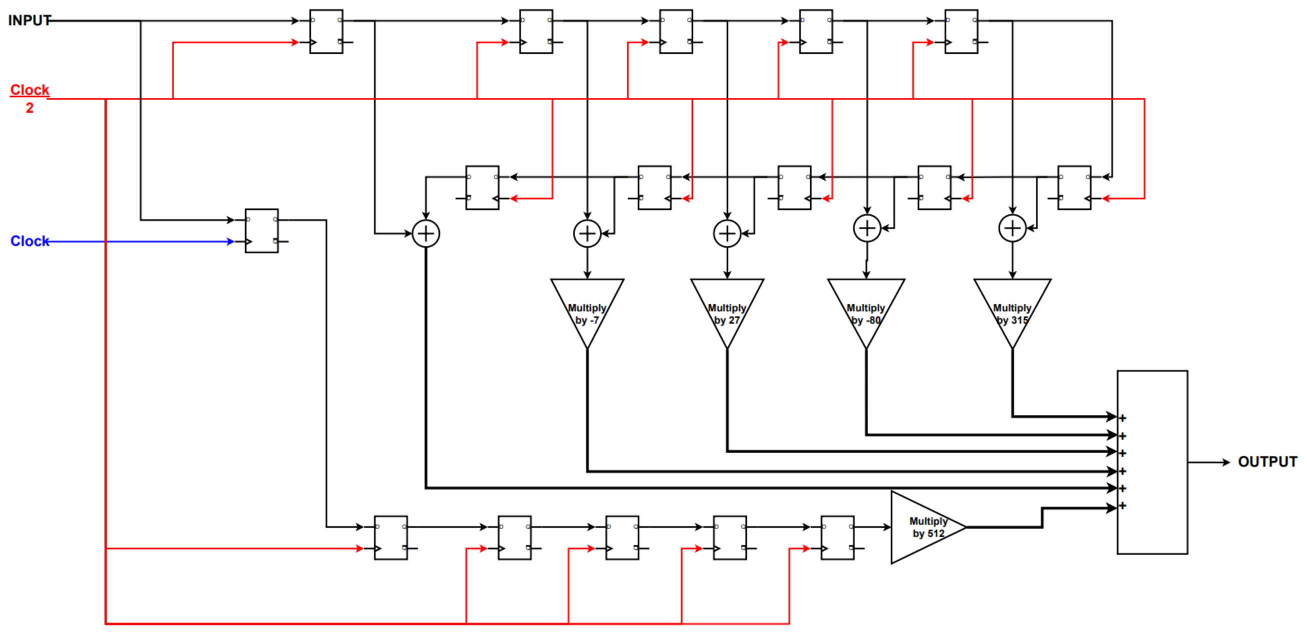


Figure 99 Half-band Diagram

The diagram shown in figure 98 shows the internal digital implementation of the half-band filter. We take the advantage of the symmetric coefficients and add the nodes with the same coefficients together before multiplying by a gain to reduce the number of multipliers used to half.

Some of these multipliers are implemented using shifters only.

The Polyphase representation allows us to make most of registers works with the low clock frequency, this HB consists of 1 register only that works with the high clock and 15 register works with the half of that clock.

Each HB increases the number of bits by 10-bits, so the first HB filter's output is 28-bits and the second HB filter's output is 38-bits.

4.7.3 Simulation Results

Figure 100 shows the simulation results, the first waveform is the Decimation input which is represented in only 6 bits. The second waveform is the Comb filter output which is represented in 18 bits. The third one is the first half-band output represented in 28 bits. The last waveform is the final decimation output after the second half-band represented in 38 bits.

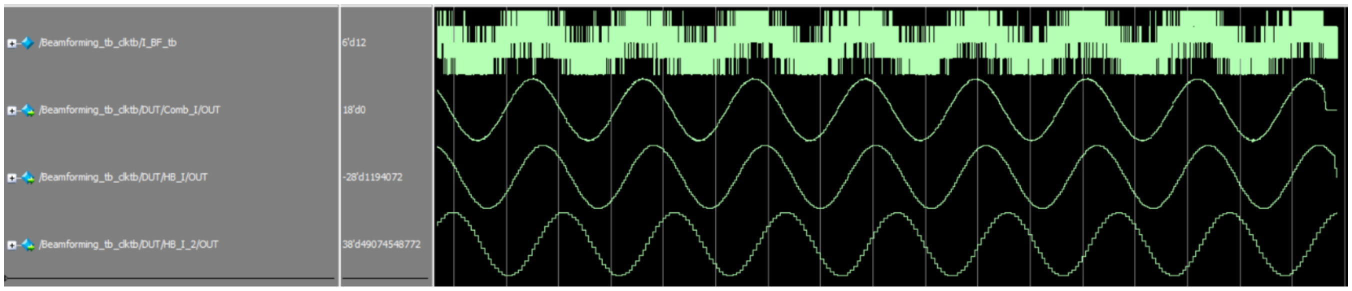

/Beamforming_tb_clktb/I_BF_tb
6'd12
/Beamforming_tb_clktb/DUT/Comb_I/OUT
18'd0
/Beamforming_tb_clktb/DUT/HB_I/OUT
-28'd1194072
/Beamforming_tb_clktb/DUT/HB_I_2/OUT
38'd49074548772

Figure 100 Decimation Simulation

Figure 101 show the output spectrum with SNR equals to 76 dB which is almost equal to the MATLAB model output SNR.

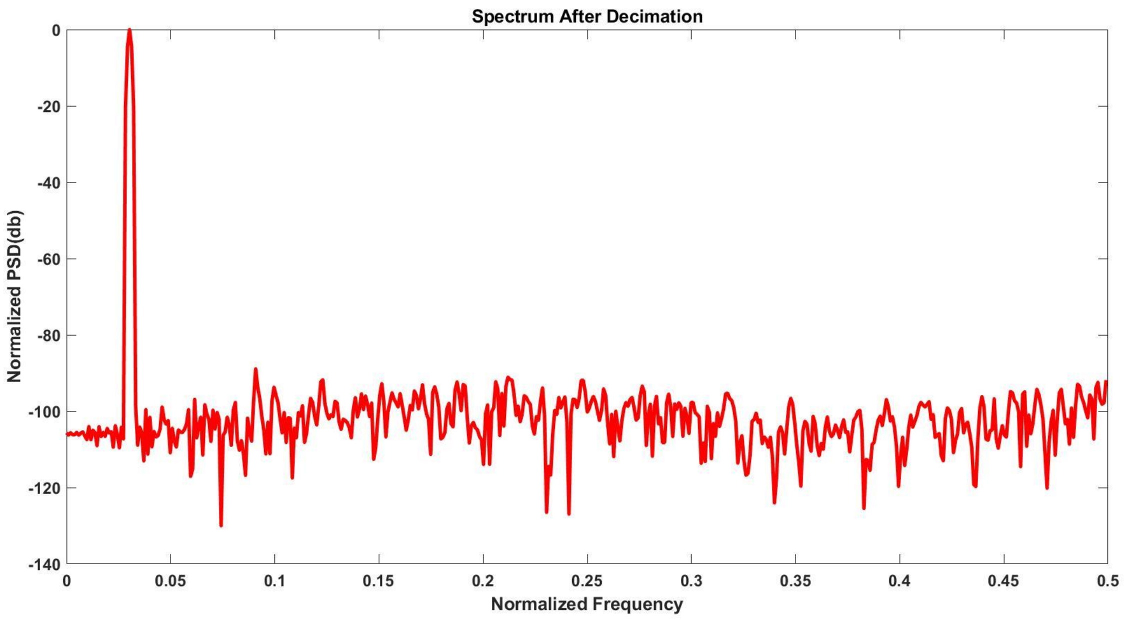


Figure 101 Output Spectrum

Chapter 5: Logic Synthesis

5.1. Synthesis Overview

Synthesis is the process that translate, and map RTL code written in HDL into a technology specific gate-level netlist, optimized for a set of pre-defined constrains using synthesis tool.

We used Synopsys Design Compiler tool.

The Technology library (standard cell library) is a collection of different combinational and sequential logic cells.

We used an educational technology library "NanGate 45nm Open Cell Library".

For delay calculations, Synopsys tool supports several delay models, but the Non-linear delay model (NLDM) is the most delay model used due to its high efficiency.

Standard cells are characterized in the technology library at specific operating condition: Process, Voltage and Temperature (PVT), so we need to consider during synthesis and timing analysis the PVT variations, these variations are shown in figure 102.

The temperature variation on a chip is a fact of life that can't be avoided. The threshold voltage is directly proportional with the chip temperature, so as temperature increases the threshold voltage increases then the propagation delay increase.

The design's supply voltage can vary from the ideal value during day-to-day operation. MosFet current is directly proportional to supply voltage, so as voltage decrease the current will decrease then the propagation delay increase.

The process variation accounts for deviations in the semiconductor fabrication process parameters which can be impurity densities, oxide thicknesses and diffusion depths which introduces variations in the transistor parameters such as threshold voltage and transistor dimensions (W/L).

Synthesis tool must consider these variations across the chip which directly affects the propagation delay, so it must consider the worst case and the best-case delays for the expected variations.

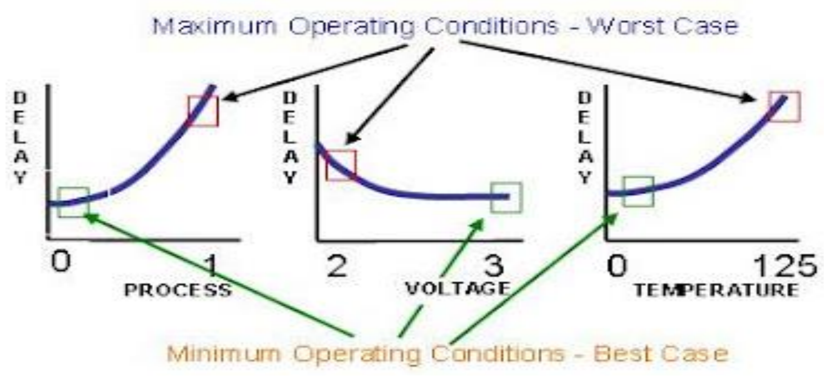

Maximum Operating Conditions - Worst Case
DELAY
DELAY
DELAY
0
1
PROCESS
2
3
VOLTAGE
0
125
TEMPERATURE
Minimum Operating Conditions - Best Case

Figure 102 PVT variations

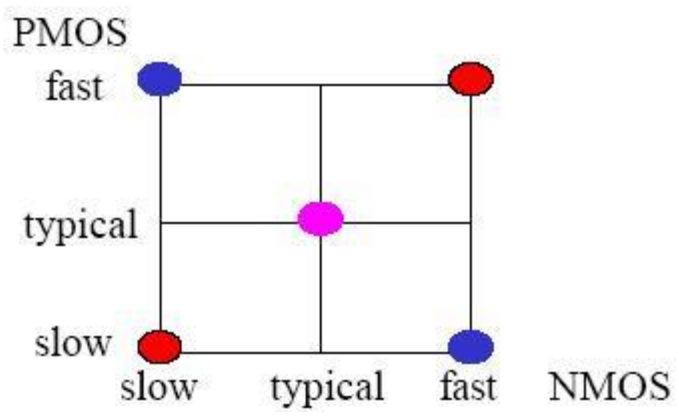


Figure 103 Design corners

As shown in figure 103, any combination of NMOS and PMOS variations will lead to what is known as a "design corner". Specifically, a design condition in which the two MOSFET types lie in a certain extreme, straining the performance of a circuit in a different way. There are four proper corners, namely: FF, FS, SF, and SS. FS means that the NMOS devices are fast, while the PMOS devices are slow. FF and SS are symmetric corners that affect both devices equally. FS and SF are skewed corners that affect the two devices differently. Colloquially, the design conditions TT, FT, ST, TS, and TF are also sometimes termed "corners"; although in reality they always represent less strenuous situations than the four true corners.

We need to specify at least two libraries; worst-case library (SS) in which cells are characterized for maximum delay and best-case library (FF) in which cells are characterized for minimum delay.

5.2 Describing our design

Firstly, Design compiler read the designs, check for any un-synthesized syntax and link the designs. As mentioned, our design has an asynchronous reset with one main clock, which is frequency divided to another four generated clocks; clock/2, clock/16, clock/32 and clock/64. The design has 4 1-bit inputs and 8 4-bit inputs. We set a driving cell for input ports all with BUF_X32 which has the highest driving power. We set input delay for all input ports that is 10% of the main clock period and set output load at all output ports. We set a wire-load model from the library to calculate an approximate value of the wire delay.

Figure 104 show the hierarchal design inside Synopsys Design compiler tool.

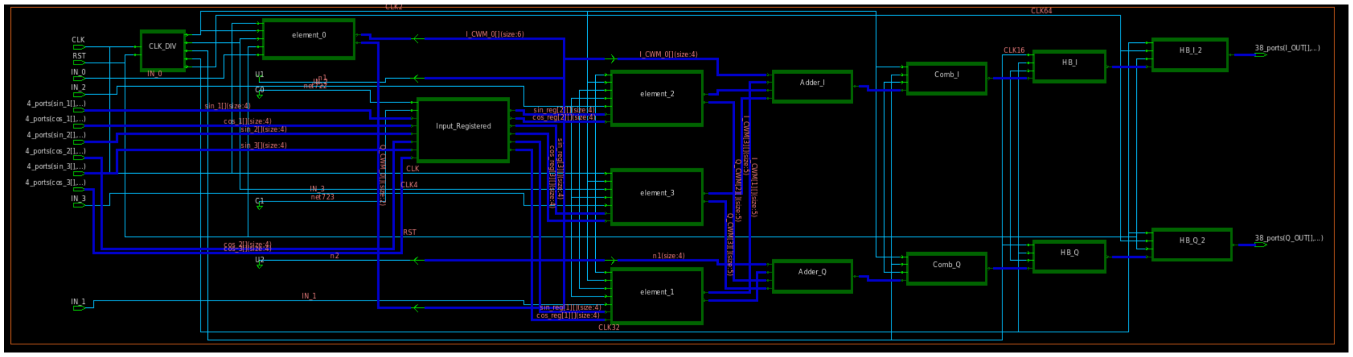


Figure 104 Hierarchal design inside Synopsys Design compiler tool

5.3. Synthesis Results

After successively compiling the design, we get:

A Clock frequency of 960 MHz with no violations.

Total cell area of $25568\ \mu m^2$ with only $750\ \mu m^2$ in the beamforming and $24818\ \mu m^2$ in decimation filter.

Total power of $1.7391\ mW$, with dynamic power equals to $1.4543\ mW$ and cell leakage power equals to $284.8677\ \mu W$ (the beamforming module only consumes $406\ \mu W$).

5.3.1. First Pipelining

The synthesis tool can't get a clock frequency more than 960 MHz because of the critical path that starts from the Interleaver at the beginning of the design and ends at the first register in the comb filter as shown in figure 105.

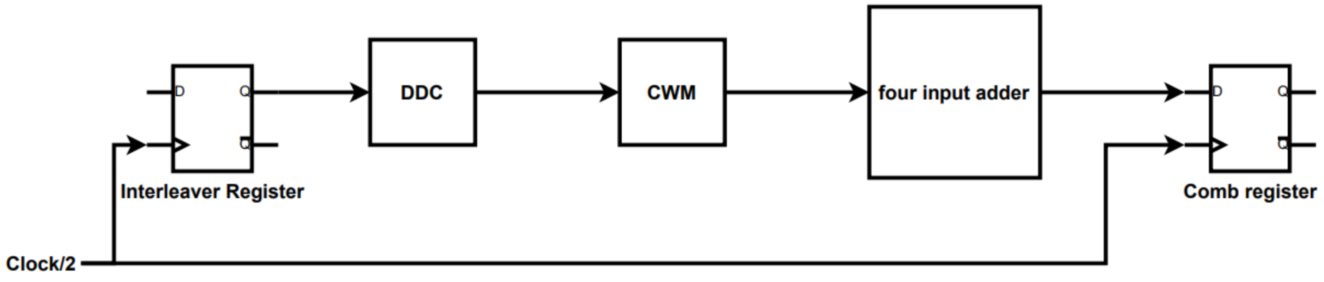


Figure 105 The critical path between interleaver and comb register

After analyzing the timing reports, pipelining was the best decision in order to have higher clock frequency to increase the throughput. We insert only one register after the CWM as this is almost half the critical path as in the diagram shown in figure 106.

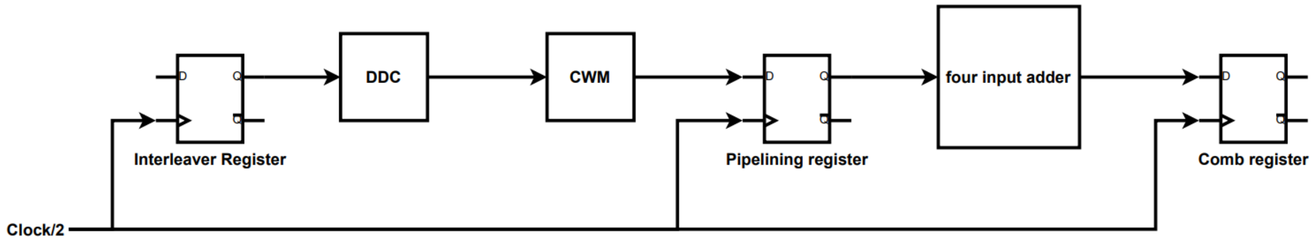


Figure 106 Pipelining by using 1 register after CWM

With this design the clock frequency increases to reach 1.38 GHz instead of 960 MHz.

The total cell area increases to be $25637\ \mu m^2$, the beamforming module area increases to be $836\ \mu m^2$ instead of $750\ \mu m^2$.

The power increases significantly to be $2.4475\ mW$ instead of $1.7391\ mW$.

The static power becomes remains almost the same, but all the increase was in the dynamic power.

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

As shown in this equation the switching power is directly proportional to the clock frequency, the frequency increases about 43.75% then the switching power is predicted to be 43.75% more than before and that's what actually happen, and the total dynamic power becomes $2.1627\ mW$.

5.3.2. Second Pipelining

The critical paths are still that two paths; the first path begin from the interleaver register and ends at the pipelining register and the other path that begin from that pipelining register and ends at the comb register, the two paths have almost the same propagation delay. So, we decide to do more pipelining to increase the throughput more.

In the first path, after analyzing the reports, to get the best delay we decide to add a pipelining register inside the CWM between the MUX and the Adder.

In the second path we can divide the four-input adder into three two-input adders and put a pipelining register between them.

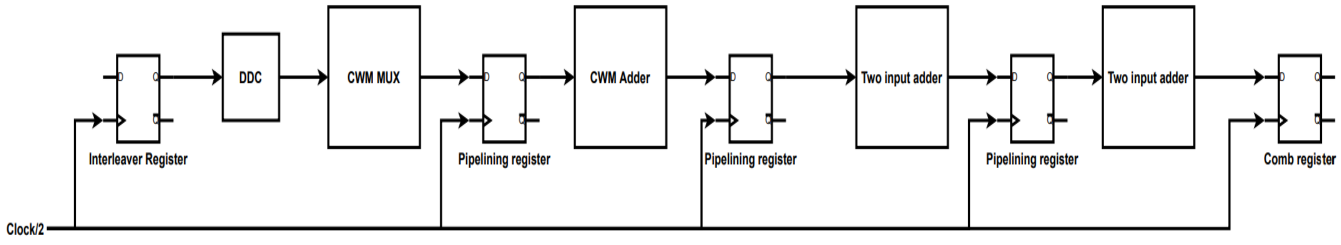


Figure 107 Pipelining inside the CWM and dividing 4 input adder and pipeline it

After this pipelining as in figure 107s the clock frequency increases to be 2.12 GHz instead of 1.38 GHz before. The area increased more to be $27511\ \mu m^2$ with an increase in beamforming area to be $1253\ \mu m^2$ instead of $836\ \mu m^2$.

The total power becomes $4.0739\ mW$ instead of $2.4475\ mW$, with small increase in static power to be $304\ \mu W$ instead of $284.8677\ \mu W$, but with significantly increase in dynamic power because of the reasons mentioned before to be $3.7699\ mW$ instead of $2.1627\ mW$.

5.3.3. Adding Pipelined ripple carry adder

Now the paths have almost equal delays so in order to increase the clock frequency again more pipelining is done in multiple paths and replacing the adders with pipelined ripple carry adders.

With this design the clock frequency increased to be $2.24\ GHz$ which the maximum we can get by this technology, due to its registers' min pulse width constrains.

The total cell area increased to be $30333\ \mu m^2$ instead of $27511\ \mu m^2$ with an increase in beamforming area to be $3418\ \mu m^2$ instead of $1253\ \mu m^2$.

The Total power becomes $7.0448\ mW$ instead of $4.0739\ mW$, with total dynamic power of $6.7134\ mW$.

Table 8 shows a comparison between the 4 designs according to the operating frequency, total cell area and total power consumed.

Design	Frequency	Area	Power
No Pipeline	$960\ MHz$	$25568\ \mu m^2$	$1.7391\ mW$
1-Register Pipeline	$1.38\ GHz$	$25637\ \mu m^2$	$2.4475\ mW$
3-Register pipeline	$2.12\ GHz$	$27511\ \mu m^2$	$4.0739\ mW$
With pipelined adder	$2.24\ GHz$	$30482\ \mu m^2$	$7.1553\ mW$

Table 8 Synthesis Comparison

Chapter 6: Physical Synthesis

6.1 Definition:

A design that satisfies timing constraints after logic synthesis will not necessarily meet timing constraints after place-and- route due to wire delays. Physical synthesis has been emerged as a necessary weapon for design closure. It is a core component of modern VLSI design methodologies for ASIC.

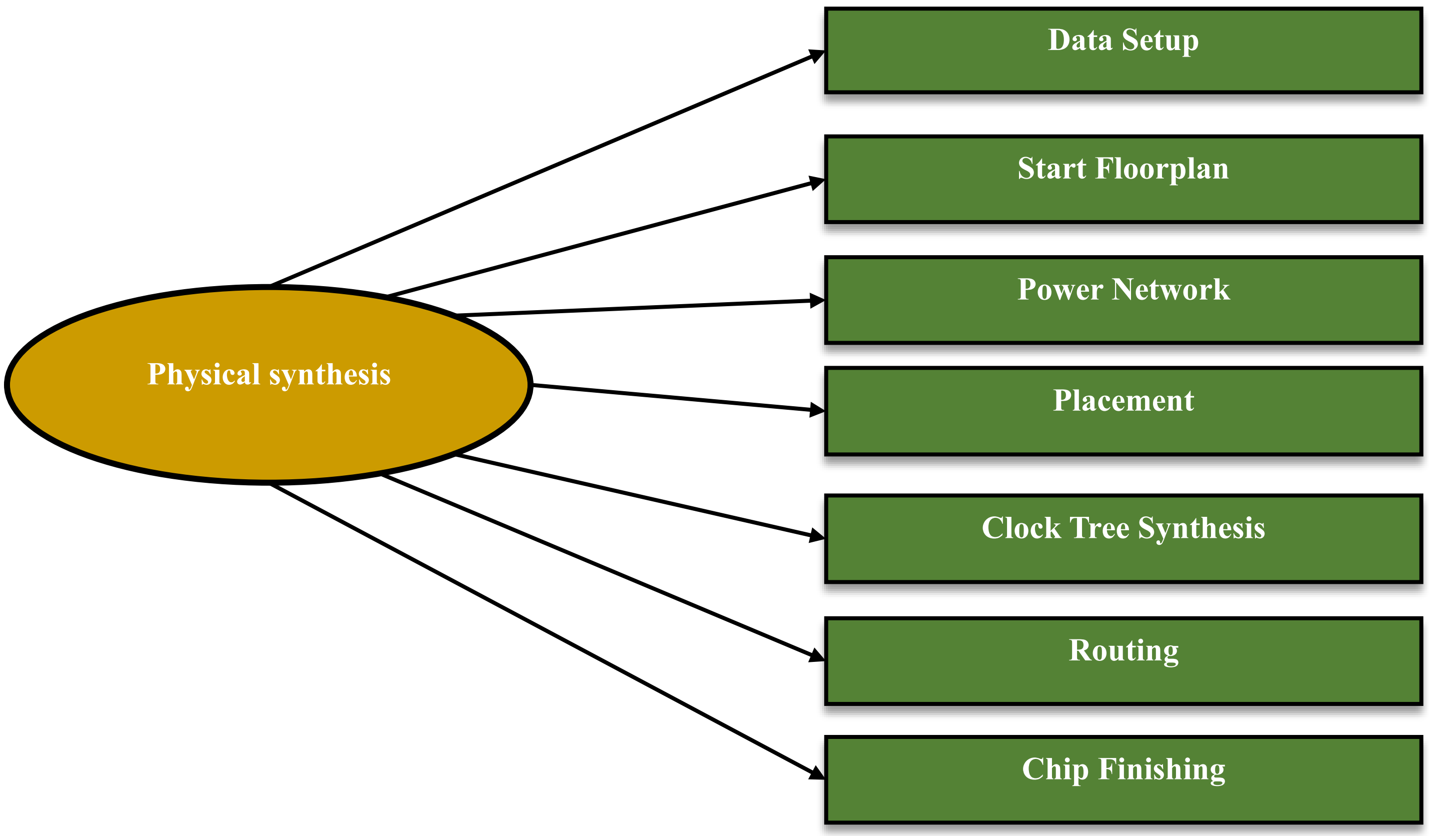


Figure 108 Physical Synthesis stages

6.2 Stages:

6.2.1 Data Setup:

To do PNR we used icc tool in Synopsys EDA. To be able to start physical synthesis we must define 2 kinds of libraires:

1. Logical Libraries:
   They are the same used in logical synthesis to provide timing and functionality information for all standard cells. They are the same used in logic synthesis.
2. Physical libraries:
   In icc tool the main physical library is Milkyway library which is used to store the design and its

associated library information. It also contains physical information about standard cells. The technology file used in the design is FreePDK45_10m.tf.

6.2.2 Floorplan:

The main goal is to create a floorplan that is likely to be routable and achieve timing closure. Floorplans define several parameters:

1. Core size:
   As the cells area we got from logical synthesis was 28209.832 μm^2 so we define the core size is $400\mu m \times 400\mu m$ to be able to route in flexible way and to decrease congestion as much as we can.
2. Aspect ratio:
   As the height and the width of core are equal which mean we have unity aspect ratio
3. Core utilization:
   We start our design by 0.5 utilization, but it didn't work at the selected frequency then we increase it to 0.85.
4. Power planning (pre routing):
   Define the area which will be used later for power ring and its value is $50\mu m \times 50\mu m$ from each side.

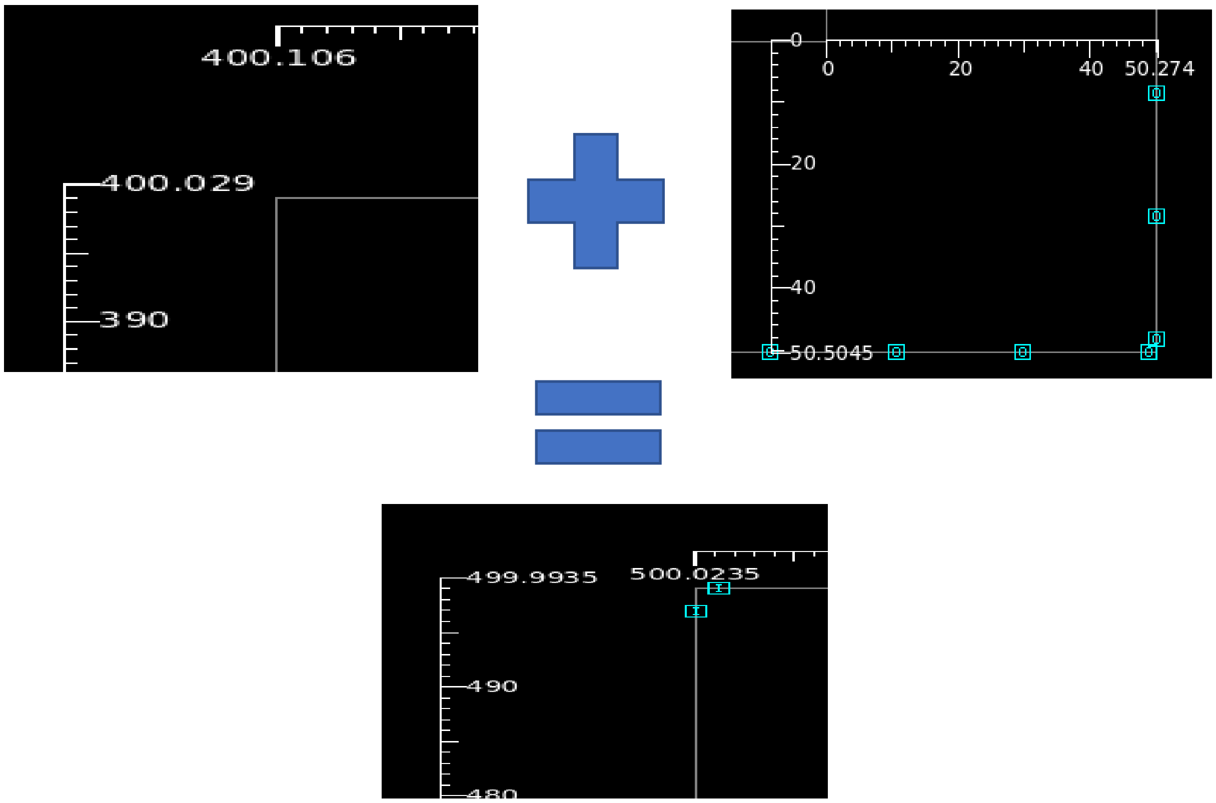


Figure 109 The dimensions of core and chip

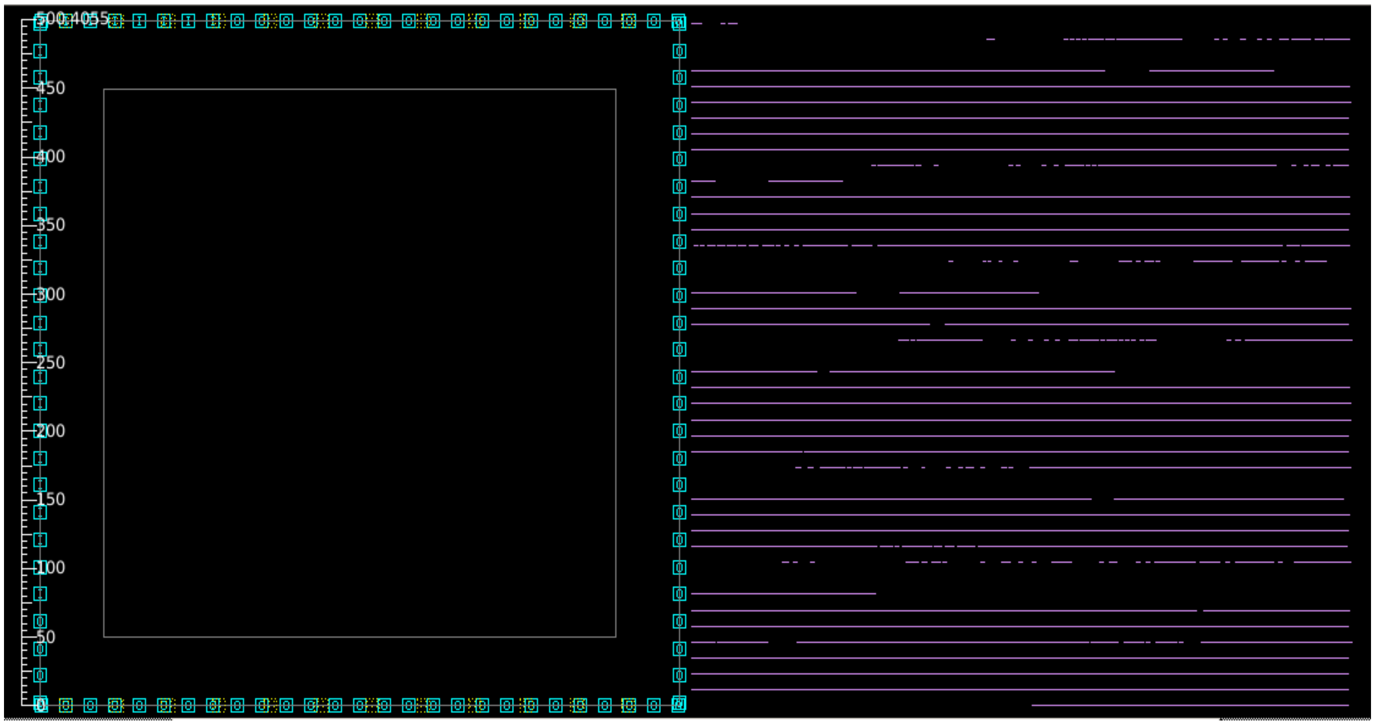


Figure 110 The chip before place any cells

6.2.3 Power Network:

In this stage we define the shape of power network, so we define 2 types of power distribution:

1. Power Grid:
   To connect the power pin to the chip. We use metal layers 10 and 9
2. Powe Mesh:
   To allow multiple paths from power and ground sources to destination (Standard cells). We use metal layers 8 and 7.

We had to check the most important phenomena which is IR drop. It's the drop of supply voltage over the length of the supply line.

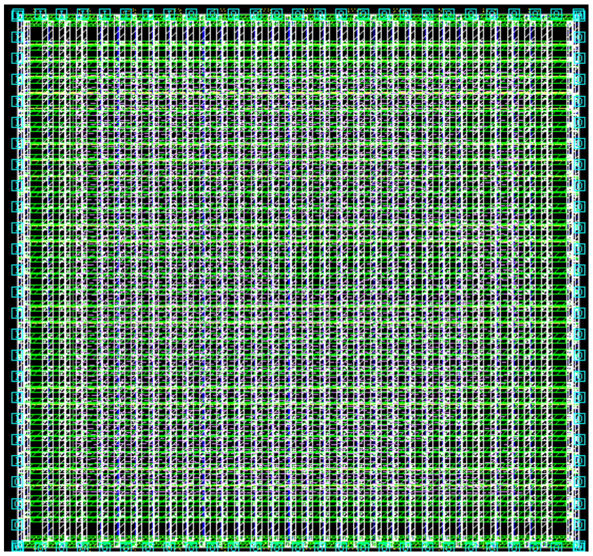


Figure 111 Power network

As we use Nangate45nm library the voltage is swing between 1.25V and 0.95V with typical voltage equal to 1.1V so it’s only available to vary $\pm 150mV$.

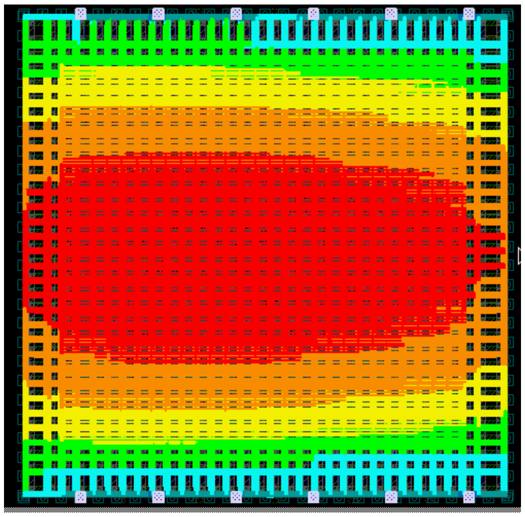
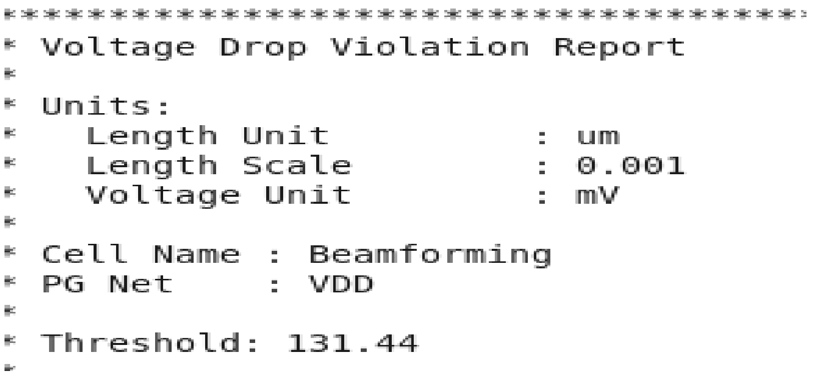


Figure 112 PNA Voltage map

As we see in fig (113) the IR drop voltage is 131.44 mV which mean we are in the safe side.



```
*************************************
* Voltage Drop Violation Report
*
* Units:
*    Length Unit            : um
*    Length Scale           : 0.001
*    Voltage Unit           : mV
*
* Cell Name  : Beamforming
* PG Net     : VDD
*
* Threshold: 131.44
*
```

Figure 113 Value of IR drop from reports

6.2.4 Placement:

Placement is the stage of the design flow, during which each instance is given an exact location. First step is that we should check the readiness of placement of floorplan, netlist, and design constraints. Second step is to report all cells placed and legal sites for cell placement.

We check for any the whole design if there is any input which is connected to VDD or VSS to put tie cells to prevent any error due to IR drop as shown in Fig (115)

After place cells we check timing constrains, we have no setup violation, and we have multiple hold violation as shown in fig (116) and that’s correct as we put many cells which will be solved in next stage.

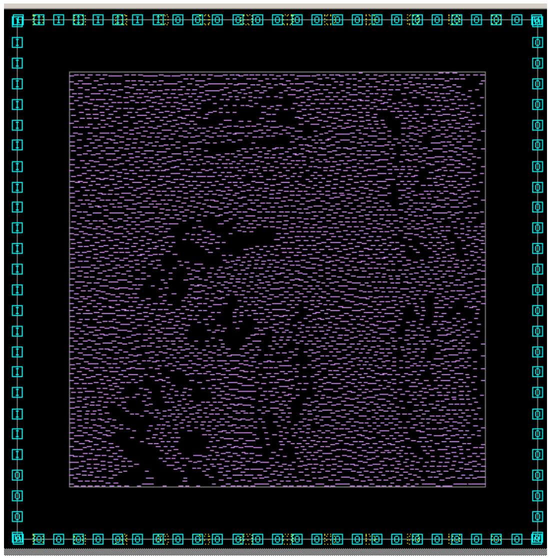
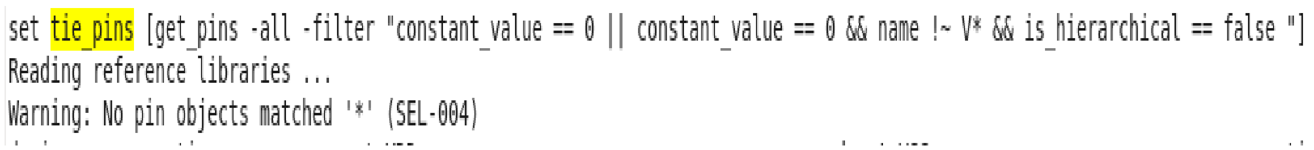
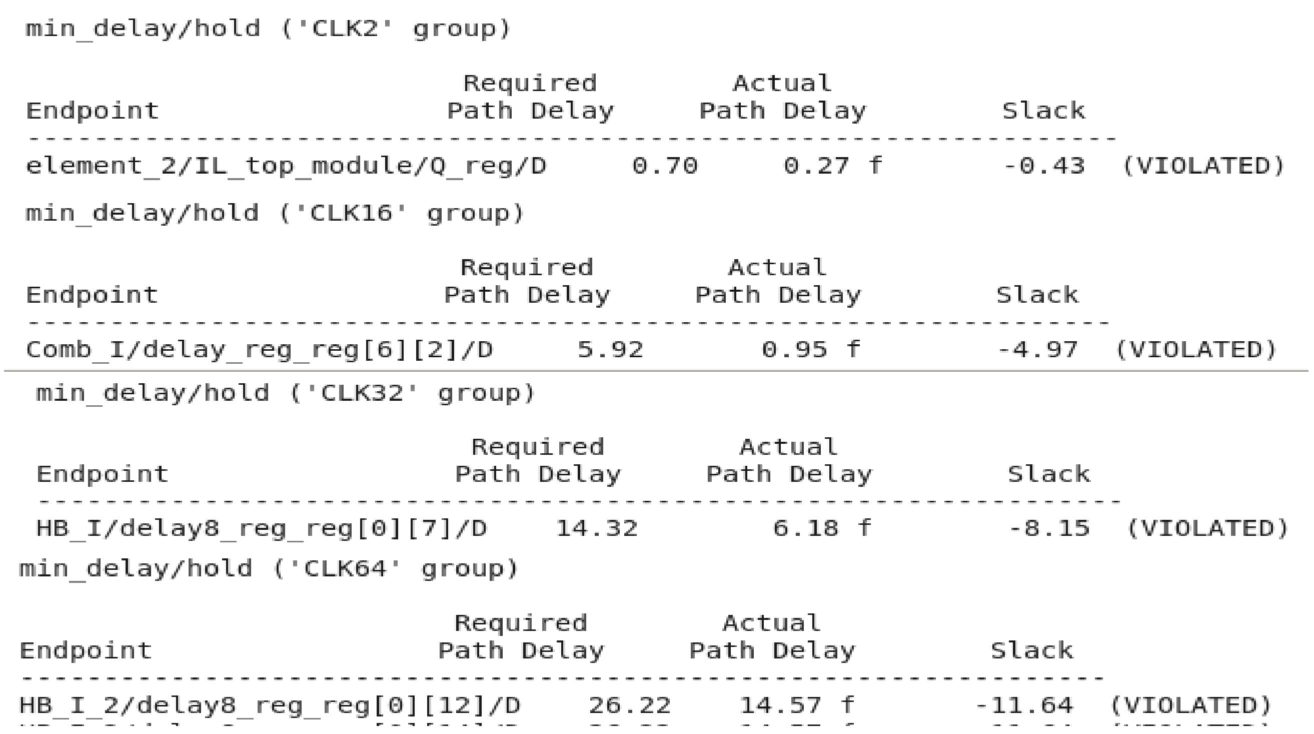


Figure 114 Cells after placement



```
set tie_pins [get_pins -all -filter "constant_value == 0 || constant_value == 0 && name !~ V* && is_hierarchical == false "]
Reading reference libraries ...
Warning: No pin objects matched '*' (SEL-004)
```

Figure 115 Check for tie cells



```
min_delay/hold ('CLK2' group)

                              Required        Actual
Endpoint                     Path Delay     Path Delay        Slack
-------------------------------------------------------------------
element_2/IL_top_module/Q_reg/D     0.70     0.27 f       -0.43  (VIOLATED)
min_delay/hold ('CLK16' group)

                              Required        Actual
Endpoint                     Path Delay     Path Delay        Slack
-------------------------------------------------------------------
Comb_I/delay_reg_reg[6][2]/D     5.92       0.95 f        -4.97  (VIOLATED)
```

```
min_delay/hold ('CLK32' group)

                              Required        Actual
Endpoint                     Path Delay     Path Delay        Slack
-------------------------------------------------------------------
HB_I/delay8_reg_reg[0][7]/D    14.32        6.18 f        -8.15  (VIOLATED)
```

```
min_delay/hold ('CLK64' group)

                              Required        Actual
Endpoint                     Path Delay     Path Delay        Slack
-------------------------------------------------------------------
HB_I_2/delay8_reg_reg[0][12]/D    26.22    14.57 f       -11.64  (VIOLATED)
```

Figure 116 Hold time violation after placement

6.2.5 Clock Tree Synthesis (CTS):

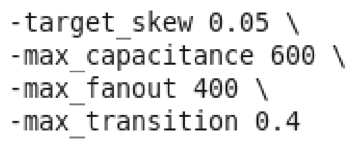
In the stage we go through multiple steps that will be shown.

1. Check design:
   We firstly check that the design is placed, and the clocks have been defined

2. Set driving cell:
   Driving cell used is CLKBUF_X3, 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.[15]

3. Set clock tree options:
   as shown in fig (117) the options used after hundreds of simulations because of the tradeoff between all parameters.

4. Set clock tree references:
   We use INV_X16 to drive the clock after driving cell to all flip flops in the design.[15]



```
-target_skew 0.05 \
-max_capacitance 600 \
-max_fanout 400 \
-max_transition 0.4
```

Figure 117 Clock tree options

5. Non default routing:
   We change the width of metals to harden the clock and make the clock routs less sensitive to cross talk or EM effects, so we doubled both spacing and width.

6. Clock placement:
   Performs clock tree synthesis, routing of clock nets, extraction, optimization, and hold time violation fixing on the design. There is also an option to perform interclock delay balancing.

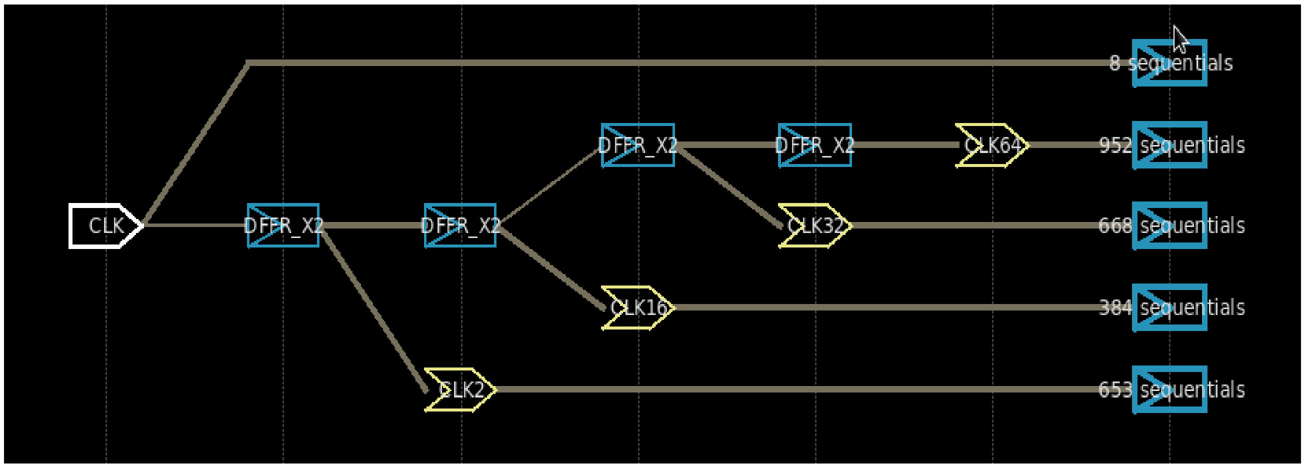


Figure 118: Clock tree and the fanout in each stage

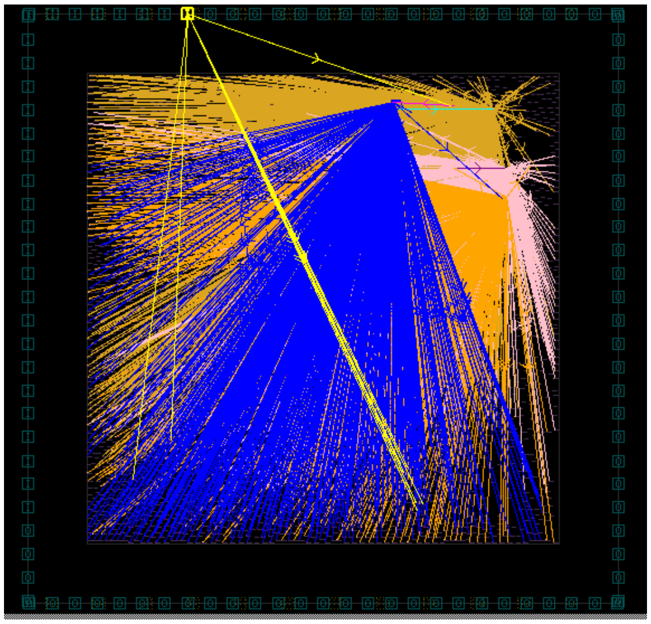
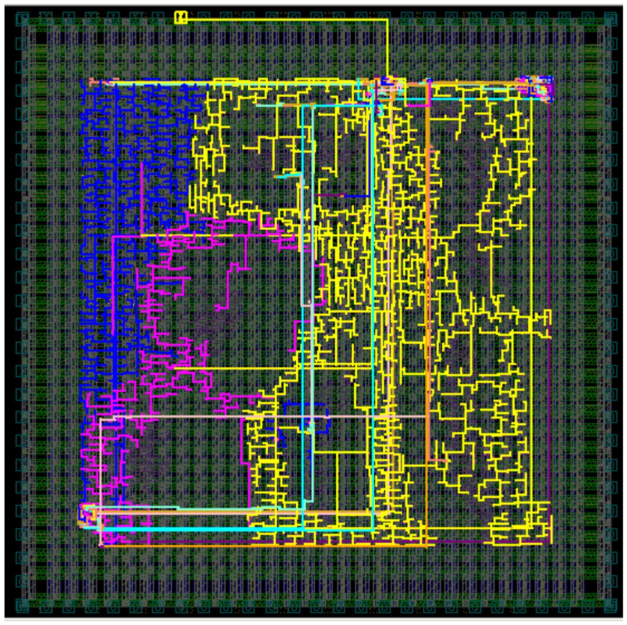



Figure 119 Clock tree before and after routing

6.2.6 Routing:

Routing creates physical connection to all clock and signal pins through metal interconnection. Routed paths must meet setup and hold timing, maximum cap/trans, and clock skew requirements. The metals used for routing are metal 1 and metal 2. In the library used there are many wire models, The used wire model in this work is arnoldi wire model because we have a huge number of driving cells which used in clock tree synthesis. [16]

6.2.7 Finishing:

At this stage we insert filler cells to connect every row with same VDD and same VSS as shown in fig (120), we also had to check:

1. If there is any floating port
2. Verify LVS vs schematic
3. Get .gds file
4. Get netlist file with physical cells
5. Extract parasites but we couldn't make that because the open-source library can't do that.

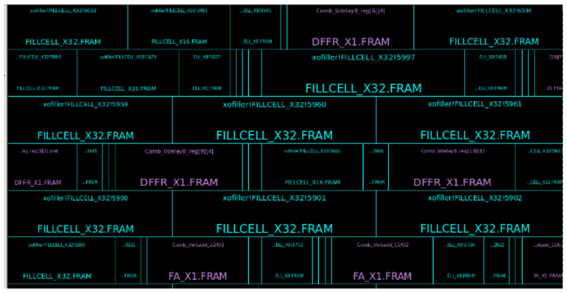


Figure 120 Tie cells

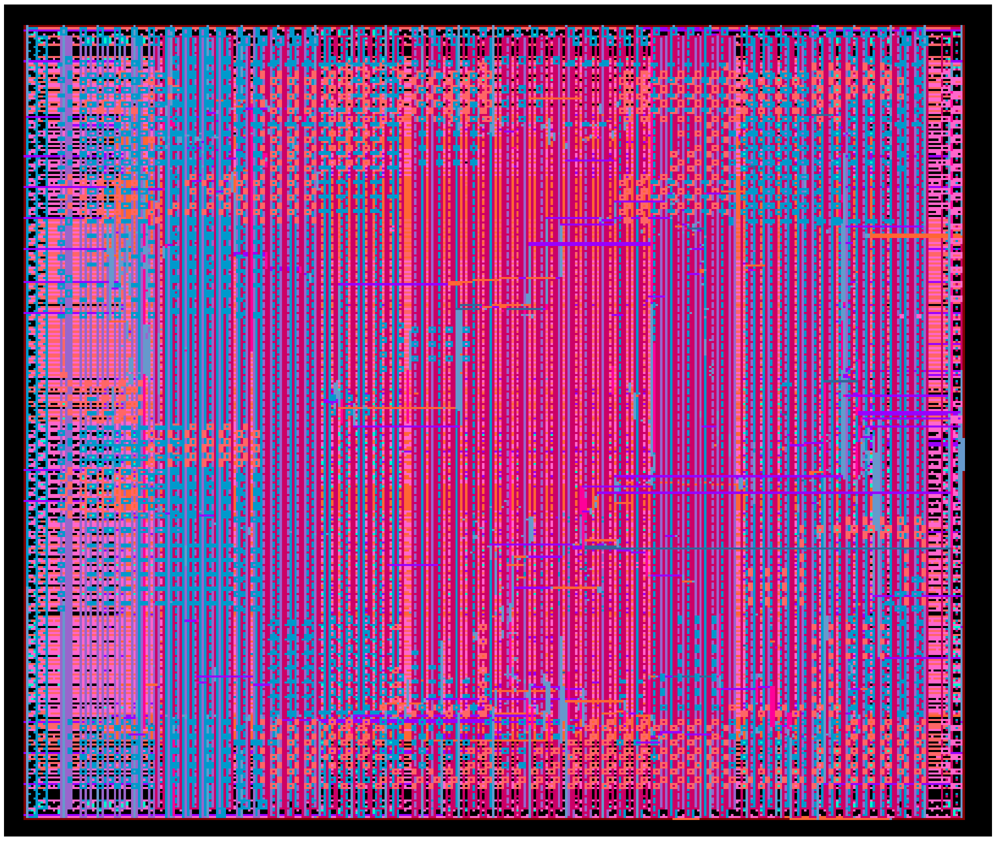


Figure 121 GDS cell

6.3 Results:

We report both area and power after every stage, as shown in table (9) that at the first 3 stages the area and the power are the same as the output are and power from synthesis as we just define core, power network, etc. but not place any cells.

We notice that the clock tree synthesis stage added more than $800\mu m^2$ because of the inverters and buffers used to drive the clock to standard cells.

Table 9 Power and area in every stage:

Stage	**Area (μm^2)**	**Power (μW)**
Define libraries	28209.832	4647.3
Floor plan	28209.832	4647.3
Power network	28209.832	4647.3
Placement	28440.72	4962.9
CTS	29248.03	9702.6
Routing	29343.79	9688.9
Finishing	29343.79	9744

Compared to Flynn paper we have multiple differences shown in Table (10):

Table 10 The comparison between this work and the last work published:

Parameter	**This work**	**Sunmin Jang 2018 [7]**
Technology	45 nm CMOS	40 nm CMOS
Input frequency	325 MHz	1 GHz
Sampling rate	1.3 GHz	4 GHz
OSR	32	20
Bandwidth	20 MHz	100 MHz
Power (μW)	9744	312
Area (μm^2)	29343.79	0.22

As shown from table above that our results are not making any improvement and that because of several reasons.

- The library used in this work is educational libraries not commercial, so we had limits on frequency used
- The library used in this work had problems but the most two critical problems:
  - ➢ The minimum pulse width problem we found that the simple flip flop used had 50% duty cycle equal to 0.244 nsec, which means that the highest frequency we can get using this library is 2.05Ghz.[14]
  - ➢ There is no wiring area in this library.

Chapter 7: Formal Verification

7.1 Definition

Formal verification is a step that uses mathematical modeling in order to prove or disprove the design correctness with static analysis (this is known by formal method) by comparing the initial design (reference) with the target design (implementation). It is shown from fig (117) that this step is done twice in our design. The first time is after the logical synthesis where the verification is done between the RTL (initial design) and the synthesized netlist (target design). The second time we use the formal verification is after the physical synthesis where the verification is done between the synthesized netlist (initial design) and the post routed netlist (target design). These two verifications and their results will be discussed later in this chapter.

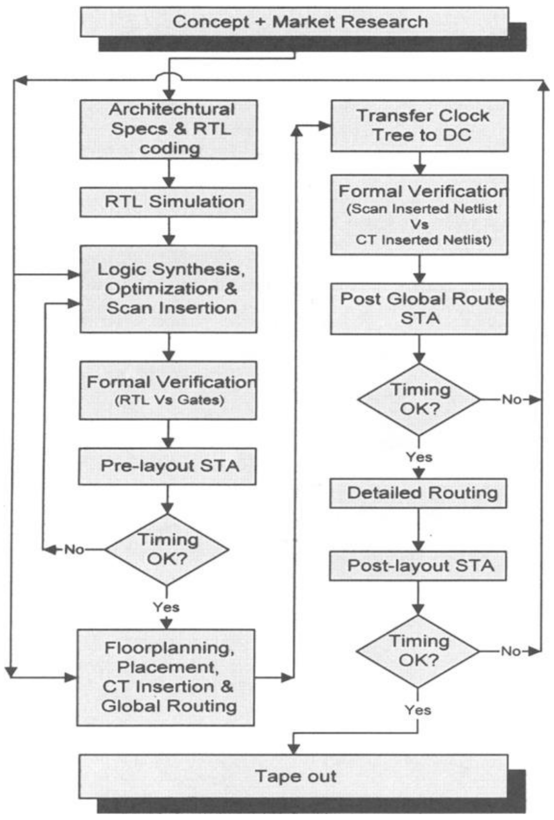

Concept + Market Research
Architechtural Specs & RTL coding
RTL Simulation
Logic Synthesis, Optimization & Scan Insertion
Formal Verification (RTL Vs Gates)
Pre-layout STA
Timing OK?
No
Yes
Floorplanning, Placement, CT Insertion & Global Routing
Transfer Clock Tree to DC
Formal Verification (Scan Inserted Netlist Vs CT Inserted Netlist)
Post Global Route STA
Timing OK?
No
Yes
Detailed Routing
Post-layout STA
Timing OK?
No
Yes
Tape out

Figure 122 Asic verification flow [18]

Formal verification consists of two main blocks which are the equivalence checker and the model checker. The equivalence checker is responsible to check if a design is logical equivalent to another while the model checker is responsible to check if the design adheres to a specified set of logical properties.

We used the Formality tool (which is introduced by Synopsys) in order to apply the formal verification step. This tool's purpose is to detect the unexpected differences that may occur to the design during the development and it is an equivalence checker tool. [18]

7.2 The difference between the formal and functional verifications

The functional verification is done by simulation using Modelsim and other tools. This type of verification is incomplete as it tests certain cases in the testbench. As well as it takes a lot of effort in order to write a testbench that can test the critical points and consumes a reasonable time for one simulation run.

As for the formal verification that uses the formal methods. This type overcomes the insufficient simulations and the process of tracking and hunting the bugs. It gives us 100% coverage also it improves the verification quality and reduces the time and verification effort.

7.3 Steps for the using the formality tool

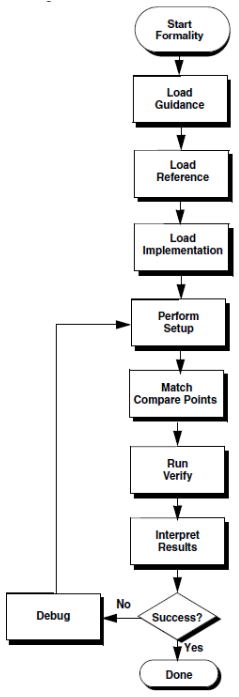


Figure 123 Steps of the equivalence check using formality [17]

From fig (118) we can see that the steps of formality to do the verification are:

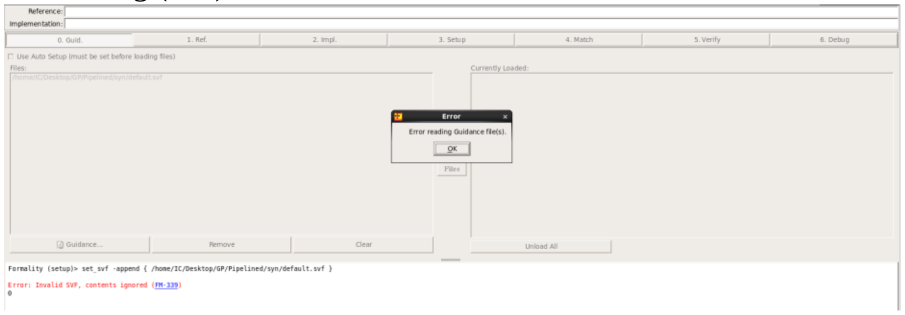
1- Start Formality: in this step we write in the terminal of the VM the word formality or fm_shell then start_gui to open the window of the tool.
2- Guidance: in this stage we insert the svf file of the design. This file is known as automated setup files. This file provides setup information about design changes caused during design development.
3- Load Reference: in this stage we load Verilog files and set the top design. In case the Verilog files are netlist the set top design will suffer from error to solve it we need first to insert the used db files then set top.

4- Load Implementation: in this stage we load Verilog files and set the top design. In case the Verilog files are netlist the set top design will suffer from error to solve it we need first to insert the used db files then set top.
5- Perform Setup: in this stage we supply the formality by information in order to take in consideration the design specific issues that the guidance step don't care about.
6- Match the compare points
7- Run verification
8- Interpret the results: the results of the verification are PASS (all compare points are equivalent), FAIL (some compare points aren't equivalent), or INCONCLUSIVE (some compare points are either unverified or aborted)
9- Debug: we enter this step (stage) in case that the verification failed (the results contain FAIL or INCONCLUSIVE) in order to know what are the problems and try to reach solutions for them by comparing the schematic of both the Reference and the Implementation designs. [17]

7.4 Formal verification after logical synthesis

In this part we apply the steps and represent faced problems and how we solve them. This verification is RTL to netlist verification.

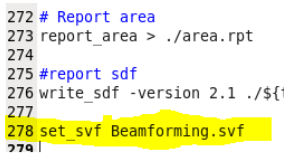
1- Start formality as stated before.
2- Load .svf (guidance): file here we faced an error while loading the .svf file resulted after synthesis as shown in fig (119).



```
Formality (setup)> set_svf -append { /home/IC/Desktop/GP/Pipelined/syn/default.svf }

Error: Invalid SVF, contents ignored (FM-339)
0
```

Figure 124 Error while loading the .svf file

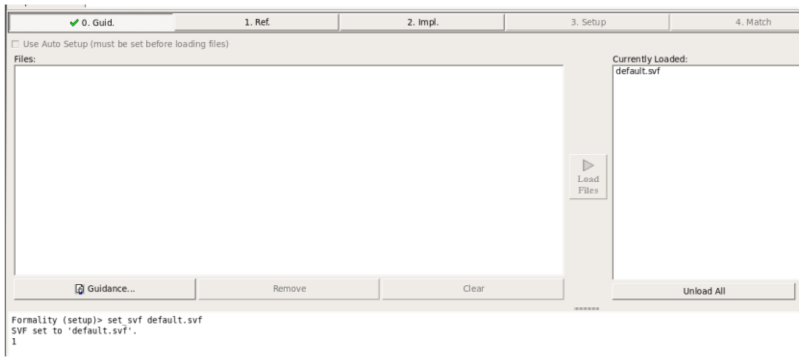
We solve this error by adding a line that generates the required .svf file at the end of .tcl file of synthesis, this line is represented in fig (120) as a highlighted line.



```
# Report area
report_area > ./area.rpt

#report sdf
write_sdf -version 2.1 ./${top_module}.sdf

set_svf Beamforming.svf

```

Figure 125 Line used to generate needed .svf file

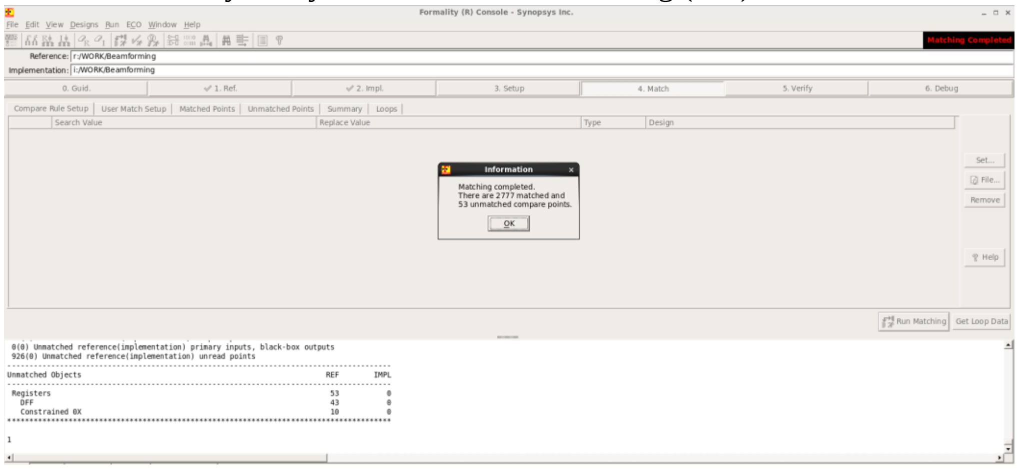
By doing this the file will be loaded successfully as shown in fig (121):



```
Formality (setup)> set_svf default.svf
SVF set to 'default.svf'.
1
```

Figure 126 .svf file loaded successfully

3- Reference: Loading all RTL design files and setting the beamforming as the top design.
4- Implementation: Loading the output netlist from logical synthesis and loading the .db files and setting the beamforming as the top design.
5- Setup: In this step we do nothing.
6- Matching: In the first we don't use .svf file this caused unmatched points as we don't consider the optimizations done by the synthesis tool as shown in fig (122).



Information

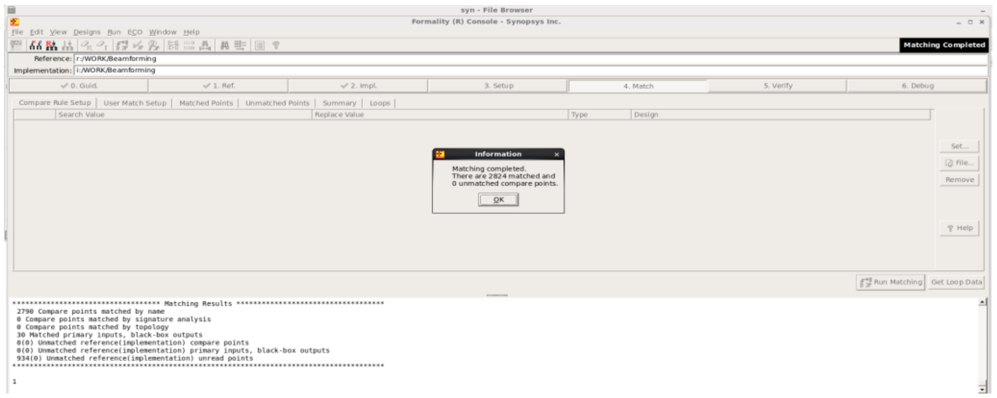
Matching completed.
There are 2777 matched and
53 unmatched compare points.

OK

```
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
 926(0) Unmatched reference(implementation) unread points
--------------------------------------------------------------------------------
Unmatched Objects                                                   REF       IMPL
--------------------------------------------------------------------------------
 Registers                                                           53          0
   DFF                                                               43          0
   Constrained 0X                                                    10          0
********************************************************************************

1
```

Figure 127 The result of matching with no svf file

After using the svf file the unmatched points disappear as shown in fig (123).



Information

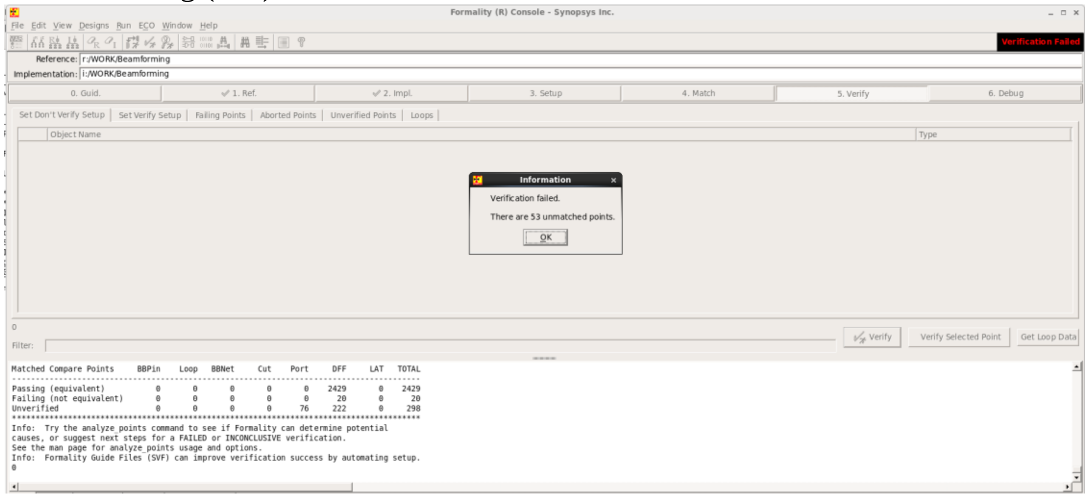
Matching completed.
There are 2824 matched and
0 unmatched compare points.

OK

```
*********************************** Matching Results ***********************************
 2790 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 30 Matched primary inputs, black-box outputs
 0(0) Unmatched reference(implementation) compare points
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
 934(0) Unmatched reference(implementation) unread points
****************************************************************************************

1
```

Figure 128 The result of matching after applying svf file

7- Verification: In the first we don't use .svf file this caused unmatched points and verification failed as shown in fig (124).



Verification failed.

There are 53 unmatched points.

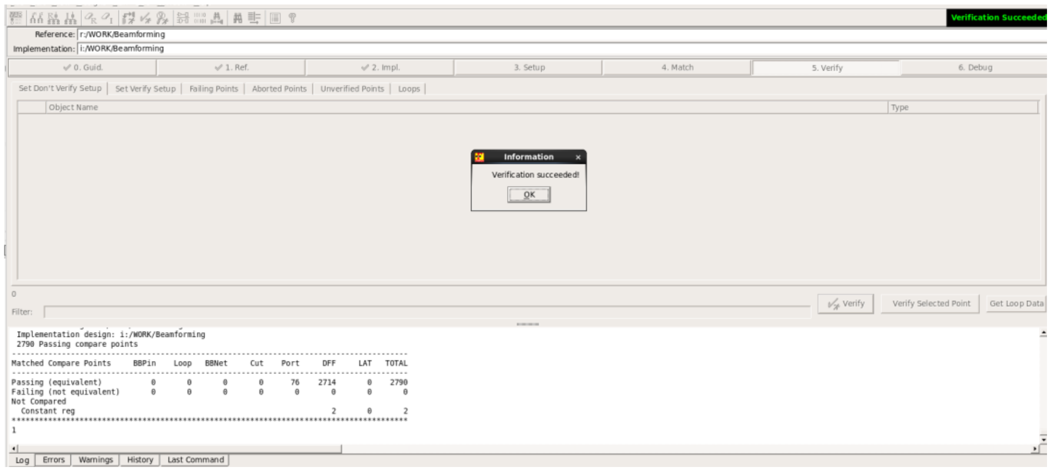
```
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
-----------------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0       0    2429       0    2429
Failing (not equivalent)       0       0       0       0       0      20       0      20
Unverified                     0       0       0       0      76     222       0     298
*****************************************************************************************
Info:  Try the analyze_points command to see if Formality can determine potential
causes, or suggest next steps for a FAILED or INCONCLUSIVE verification.
See the man page for analyze_points usage and options.
Info:  Formality Guide Files (SVF) can improve verification success by automating setup.
0
```

Figure 129 The result of verification with no svf file

After using the svf file we suffer from another problem which is the process take along time without proceeding and stops at 93%.

We solve this by changing the compile ultra in the tcl file of synthesis to compile only. As the ultra-feature can cause the multipliers and the adders to be divided then recombined so that the formality might be stopped due to uncertain mismatches.

By using the svf file and changing to compile the verification succeeded as shown in fig (125).



Verification succeeded!

```
 Implementation design: i:/WORK/Beamforming
 2790 Passing compare points
-----------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
-----------------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      76    2714       0    2790
Failing (not equivalent)       0       0       0       0       0       0       0       0
Not Compared
  Constant reg                                                         2       0       2
*****************************************************************************************
1
```

Figure 130 The result of verification after applying svf file

8- Interpret results: From the fig (125) we have 2 constant registers which are outputs of CWM_0 (I and Q regs) and all the other points are passing points. On the other hand, we see from fig (124) that we have failing points and unverified points due to reaching the maximum limit of failing points which is 20.

9- Debug: We used this step-in case of not using the svf file in order to compare between the implementation and reference schematics. For example, for a failing point what is presented in fig (126) where you can see that the 2 net names in implementation and reference are having different values and names. While an example for a passing point is presented in fig (127) where you can see that the 2 net names in implementation and reference are having the same values and names.

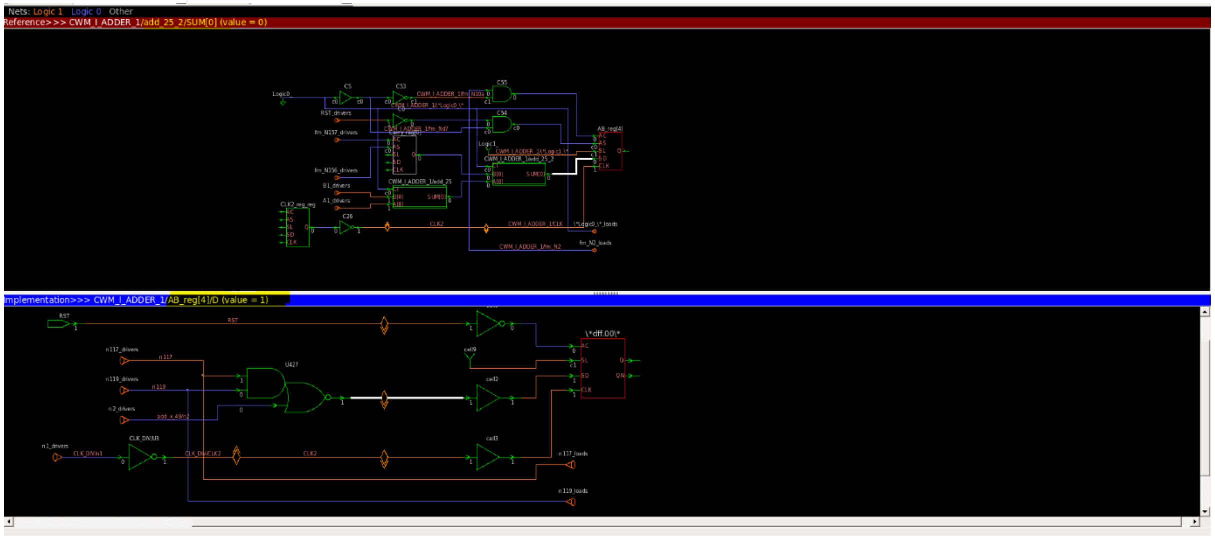

Nets: Logic 1 Logic 0 Other
Reference>>> CWM_I_ADDER_1/add_25_2/SUM[0] (value = 0)
Implementation>>> CWM_I_ADDER_1/AB_reg[4]/D (value = 1)

Figure 131 Example for a failing point

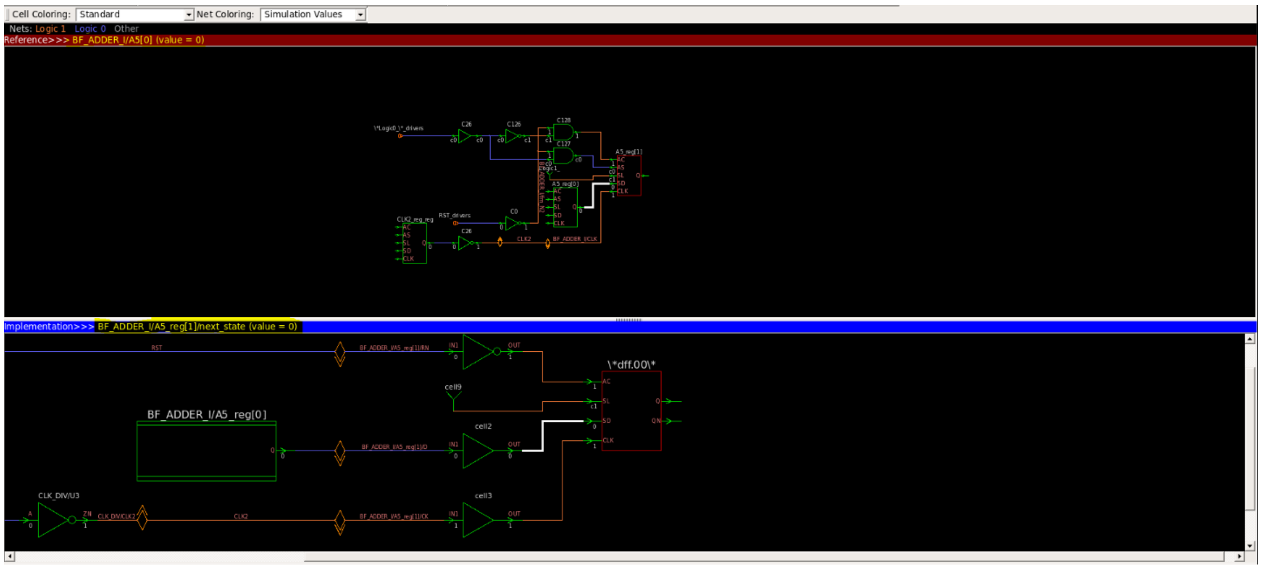
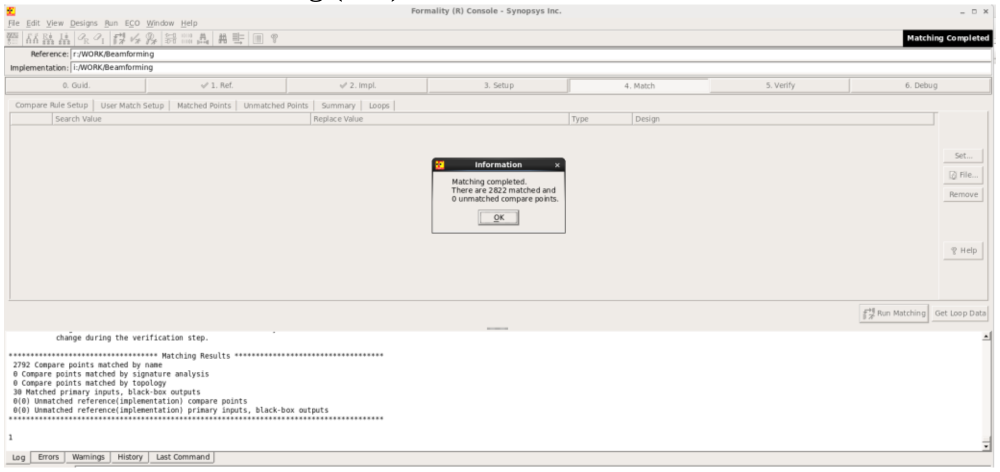

Cell Coloring: Standard
Net Coloring: Simulation Values
Nets: Logic 1 Logic 0 Other
Reference>>> BF_ADDER_I/A5[0] (value = 0)
Implementation>>> BF_ADDER_I/A5_reg[1]/next_state (value = 0)

Figure 132 Example for a passing point

7.5 Formal verification after physical synthesis

In this part we apply the steps. This verification is netlist to netlist verification. The process here is much easier than the previous one.

1- Start formality as stated before.
2- Guidance: In this step we do nothing. No need for svf files here.
3- Reference: Loading the output netlist from logical synthesis and loading the .db files and setting the beamforming as the top design.
4- Implementation: Loading the output netlist (Beamforming_icc_nopg.v) from physical synthesis and loading the .db files and setting the beamforming as the top design.
5- Setup: In this step we do nothing.
6- Matching: Its result is shown in fig (128).

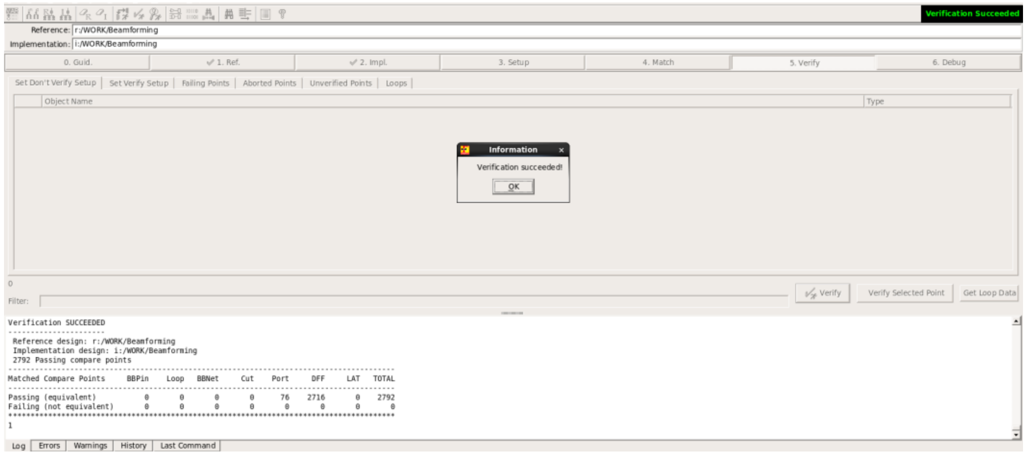


Matching completed.
There are 2822 matched and 0 unmatched compare points.

```
          change during the verification step.

********************************** Matching Results **********************************
 2792 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 30 Matched primary inputs, black-box outputs
 0(0) Unmatched reference(implementation) compare points
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
**************************************************************************************

1
```

Figure 133 Matching results

7- Verification: Its result is shown in fig (129)



Verification succeeded!

```
Verification SUCCEEDED
----------------------
 Reference design: r:/WORK/Beamforming
 Implementation design: i:/WORK/Beamforming
 2792 Passing compare points
------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
------------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      76    2716       0    2792
Failing (not equivalent)       0       0       0       0       0       0       0       0
************************************************************************************
1
```

Figure 134 The result of verification

8- Interpret results: From the previous figure we have all the points are passing points.
9- Debug: No need for this step.

Conclusion

In this thesis, we presented the idea of beamforming and how it is needed in the 5G systems. Then we listed the different beamforming receiver architectures with the advantage and disadvantage of each of them. By comparing between the architectures, we decided to use in our system the digital receiver that is based on combination of CTBPDSMs and BSP. The reason behind that is that this architecture avoids the high-power consumption and the use of large area. Our system performs the digital down conversion and the phase shifting by using only multiplexers. It occupies 29343.79 μm^2 and consumes power by 9744 μW. Also, we make sure after each stage during the development of the design that the function is still done correct, and expected results are seen.

Future Work:

For our system, there is some improvements that we can consider getting better performance and more flexibility:

- Adding LNA to increase the system SNR and Dynamic Range.
- Increase number of linear antenna elements as we are using DBF.
- Increasing number of beams in the system.
- Adding correction filter after decimation to avoid the decrease of SNR in bandwidth.
- We can use advanced DSP algorithms as they can be implemented with DBF.
- Use commercial libraries instead of educational libraries.
- Use Cadence tool instead of Synopsys tool
- Implementing the system on an IC chip.
- Verifying the system using UVM.

References

[1] J. Jeong, "IF-Sampling Digital Beamforming with Bit-Stream Processing," University of Michigan, 2015.

[2] J. Jeong, N. Collins and M. P. Flynn, "A 260 MHz IF Sampling Bit-Stream Processing Digital Beamformer With an Integrated Array of Continuous-Time Band-Pass ΔΣ Modulators," *IEEE JOURNAL OF SOLID-STATE CIRCUITS,* pp. 1-9, 2015.

[3] R. Lu, C. Weston, D. Weyer, F. Buhler and D. Lambalot, "A 16-Element Fully Integrated 28-GHz Digital RX Beamforming Receiver," *IEEE JOURNAL OF SOLID-STATE CIRCUITS,* Vols. VOL. 56, NO. 5,, pp. 1374-1386, MAY 2021.

[4] M. Flynn, S. Jang, J. Jeong and R. Lu, "16 Element 1GHz Input 4 Simultaneous Beam 100MHz Bandwidth Digital Beamformer in 40nm CMOS," pp. 25-28, 2021.

[5] S. Jang, R. Lu, J. Jeong and M. P. Flynn, "A 1-GHz 16-Element Four-Beam True-Time-Delay Digital Beamformer," *IEEE JOURNAL OF SOLID-STATE CIRCUITS,* 2019.

[6] S. Jang, R. Lu, J. Jeong and M. P. Flynn, "A True Time Delay 16-Element 4-Beam Digital Beamformer," *IEEE Radio Frequency Integrated Circuits Symposium,* pp. 12-15, 2018.

[7] S. Jang, "A CMOS Digital Beamforming Receiver," Michigan, 2018.

[8] S. Jang, J. Jeong, R. Lu and M. P. Flynn , "A 16-Element 4-Beam 1 GHz IF 100 MHz Bandwidth Interleaved Bit Stream Digital Beamformer in 40 nm CMOS," *IEEE JOURNAL OF SOLID-STATE CIRCUITS,* 2018.

[9] P. Zahradnik, B. Sim´ak and M. Kopp, "Design of Half-Band FIR Filters for Signal Compression," *International Journal on Advances in Telecommunications,* vol. 4 no 3 & 4, pp. 240-248, 2011.

[10] F. Harris, Multirate Signal Processing for Communication Systems, Second Edition, California San Diego, USA: River Publishers, 2021.

[11] Ljiljana Milić, Multirate Filtering for Digital Signal Processing: MATLAB Applications, Information Science Reference, 2009.

[12] H. Aboushady, Y. Dumonteix and M. M. Louerat, "EFFICIENT POLYPHASE DECOMPOSITION OF COMB DECIMATION FILTERS IN ΣΔ ANALOG-TO-DIGITAL CONVERTERS".

[13] H. Abushady, "LE DECIMATION DES CONVERTISSEURS ΣΔ," Paris, 1997.

[14] R. Shankar, S. Eswaran, S. Bhat and L. Balasubramanian, "Pulse Width Insensitive Design and Verification Methods," *EAI.EU,* vol. 6, no. 17, pp. 1-6, 2019 - 5/2020.

[15] H. Xin, H. Xu, Y. Wu and L. Yujing, "Clock Tree Synthesis and Optimization of SoCs under Low Voltage," *Atlantis Press,* vol. 148, p. 214, 2017.

[16] C. . J. Alpert, S. K. Karandikar, Z. Li, G.-J. Nam, S. T. Quay, H. Ren, C. N. Sze, P. G. Villarrubia and M. C. Yildiz, "Techniques for Fast Physical Synthesis Fast," *PROCEEDING OF THE IEEE,* Vols. 95, No 3, pp. 573-599, March 2007.

[17] Formailty User Guid Version D, Synopsys, 03-March-2010.

[18] H. Bhatnager, ADVANCED ASIC CHIP SYNTHESIS Using Synopsys Design Complier and Prime Time, Springer Science+Business Media, LLC, 1999.