# Hardware Implementation and Analysis of Temporal Interference Mitigation : A High-Level Synthesis Based Approach

by

Saquib Ahmad Siddiqui

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

ARIZONA STATE UNIVERSITY

August 2020

## ABSTRACT

The following document describes the hardware implementation and analysis of Temporal Interference Mitigation using High-Level Synthesis. As the problem of spectral congestion becomes more chronic and widespread, Electromagnetic radio frequency (RF) based systems are posing as viable solution to this problem. Among the existing RF methods Cooperation based systems have been a solution to a host of congestion problems. One of the most important elements of RF receiver is the spatially adaptive part of the receiver. Temporal Mitigation is vital technique employed at the receiver for signal recovery and future propagation along the radar chain. The computationally intensive parts of temporal mitigation are identified and hardware accelerated. The hardware implementation is based on sequential approach with optimizations applied on the individual components for better performance. An extensive analysis using a range of fixed point data types is performed to find the optimal data type necessary. Finally a hybrid combination of data types for different components of temporal mitigation is proposed based on results from the above analysis.

*To my family and friends and whoever made this possible*

## ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Daniel Bliss for his continuous support and belief in me. His constant encouragement and advice has helped me come a long way and has been critical to every achievement and progress made along the way. I am grateful to my committee members Prof. Chakrabarti, Dr. Ogras and Dr. Jayasuriya for their valueble time and efforts rendered toward my development. I am extremely grateful to all my colleagues at BLISS lab for constant help and support that was critical in course of my thesis. I would especially thank Yang Li and Matthew Kinsinger for their support and precious input throughout my thesis. I would like thank my parents for their unconditional love and support throughout all my pursuits.

TABLE OF CONTENTS

# LIST OF TABLES

## LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

# INTRODUCTION

This report describes details about the high-level synthesis based hardware implementation of interference mitigation. Interference mitigation is fundamental element and required step for the existence of an RF convergence based system [3].

The problem with spectral congestion in the existing wireless communication systems is forcing the users to explore techniques involving codesign and cooperation. One of the most upcoming and promising solution to this problem has been the RF convergence. RF convergence is simply defined as the operating point at a given bandwidth allocation is jointly used for both radar and communication to serve mutual benefits to each other [1]. The applications that can benefit from the RF convergence solution are numerous. The industries and systems that can benefit from cognitive radars and radio can also benefit from RF convergence. The primary applications are Automotive Radar and vehicle-to-vechile(V2V) communication systems which is an integral element of modern day smart cars. Further commercial flight air traffic control can benefit from joint sensing communication system. Medical sensors and monitoring devices can benefit from combined sensing and communication system which can lower the invasiveness and physical footprint of these devices. Radio-frequency identification(RFID) are currently based on integrating remote sensing and data transfer. Currently RFID require external stimulus or application of RF energy to initiate the communications link, there is joint sensing and communications aspect that becomes evident.

The joint sensing and communication system can be integrated in various ways :

## 1.1 Coexistence

In this method the radar and communication transceivers treat each other as interferers [2]. This essentially implies that any information that can be used to mitigate the other system's interference is not shared but must be rather estimated. An example of this setup is a passive radar that takes advantage of communications broadcast to perform radar operations but it must also mitigate the source of interference.

The Figure 1.1 explains this setup and the architecture reflects the presence of an adaptive canceler function on the path of both systems.

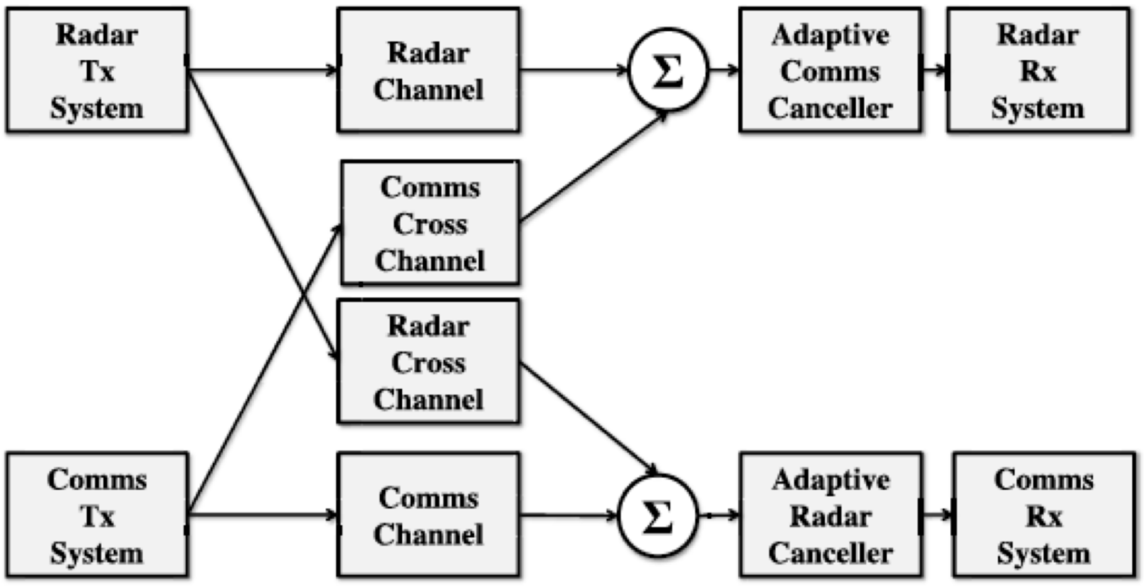


**Figure 1.1:** RF System Coexistence Block Diagram, Adapted from [3]

## 1.2 Cooperation

In this setup the users try to exploit the joint knowledge of each other to improve both their performances. This knowledge is useful in mitigating interference relative to one and other. This setup of mutually exchanging information to mitigate interference is critical step towards joint systems and seriously attempting RF convergence.

An example of this setup is when the radar system gets information from the communications system which is dynamically updated. This assists in direct path and co-channel interference mitigation. The Figure 1.2 shows this setup where the systems exchange information to benefit one and another in signal mitigation. This setup is of particular interest to us and will be discussed in more detail in the subsequent sections.

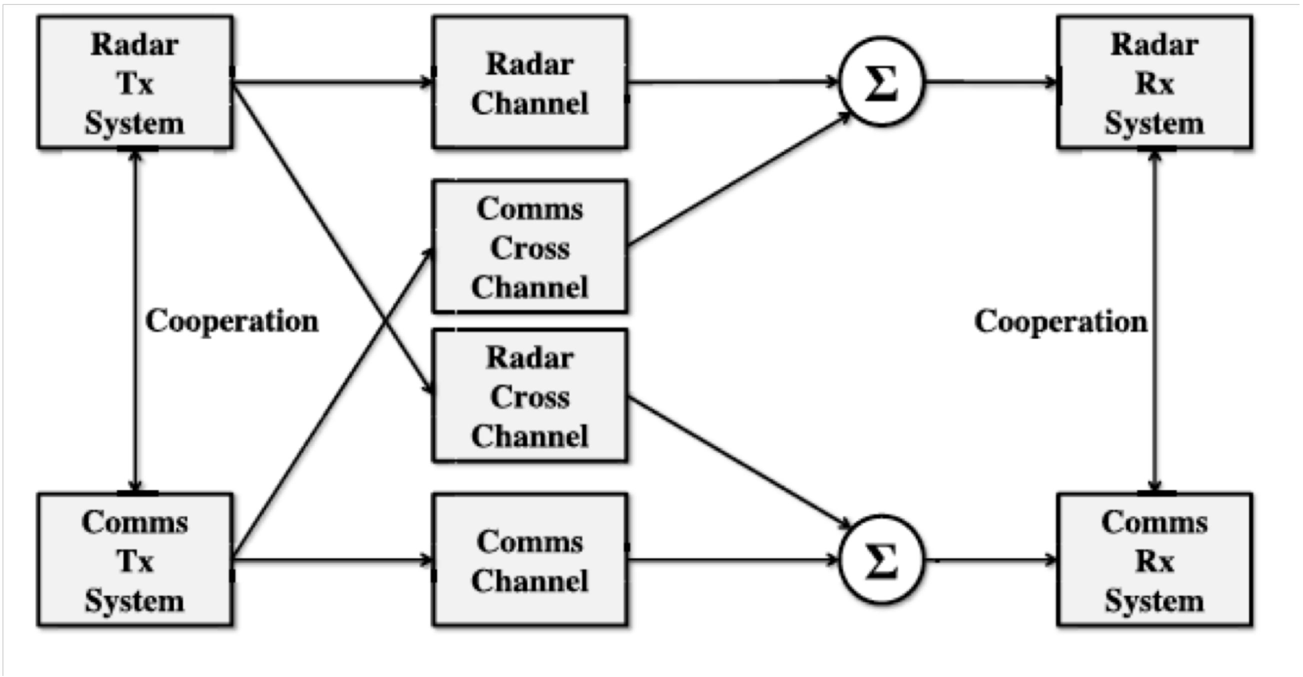


**Figure 1.2:** RF System Cooperation Block Diagram, Adapted from [3]

## 1.3 Co-Design

This method considers communications and radar jointly when designing systems to maximize the spectral efficiency. So the systems are designed jointly from the ground up and this leads to an opportunity to improve their performance over isolated operation.

The passive radar that was used in the cooperative setup can now be improved by co-design of the systems. The communications system can make design choices to use codes, modulation schemes and training sequences that can benefit the passive radar operation. The radar system can contribute by providing multi static channel estimation feedback to assist the communications system in equalization process. The Figure 1.3 reflects that where both the transmit and receive are jointly designed.

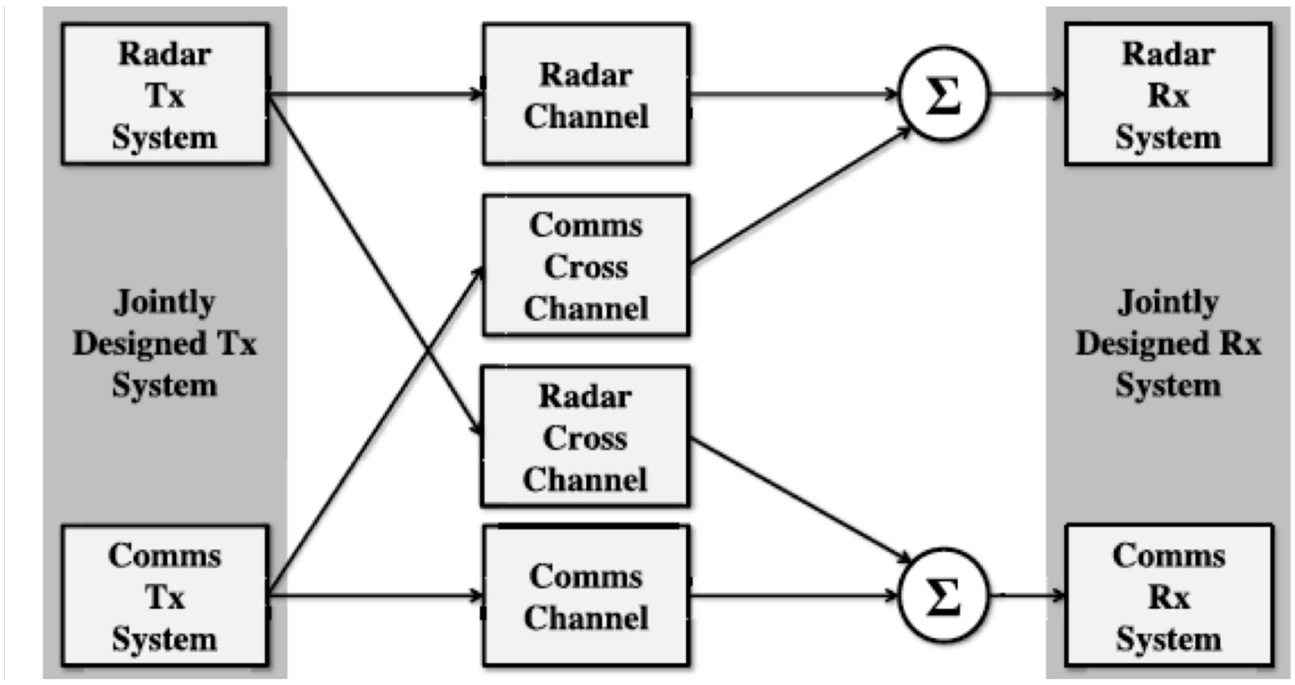


**Figure 1.3:** RF System Co-Design Block Diagram, Adapted from [3]

The following sections and chapters will try to point the focus towards RF System Cooperation setup that was described in section 1.2 as this is the most relevant concept to the further discussion of interference mitigation.

The key element in RF System Cooperation setup is the Multiple-input multiple-output (MIMO) Channel and adaptive receivers which will be described and discussed in the following sections.

## 1.4 MIMO Channel

A MIMO wireless communication link is described as a system where a single source distributes data across multiple transmit antennas as shown in Figure 1.4. By making use of the diversity offered by multiple-antennas, the communication link can be improved [4]. For instance the multiple degrees of freedom can be used to provide robustness through channel redundancy or increased data rates by exploiting multiple paths through the environment simultaneously. These benefits can also reduce the probability of interception and the knowledge of the channel can be used to generate cryptographic keys.

From the Figure 1.4 we can observe that potentially independent signals from multiple transmit antennas propagate through the environment encountering different channels. The multiple-antenna receiver then disentangles the signal from multiple transmitters. Although MIMO communication can operate in line of sight environments, the most common scenario for MIMO communication systems is to operate in an environment charecterized by complicated multipath scattering. Hence, most, if not all the energy observed at the receive array strikes upon the array from directions different from the direction to the source. The environment in which this link is operating referred to as channel.

### *1.4.1 Flat-fading channel*

A signal is considered narrow band if the channel between each transmit and receive antenna can be characterized by a single complex number. This characterization of a channel is valid when the signal bandwidth $B$ is small compared to the inverse of the charectersitic delay spread $\Delta t$,

$$B \ll 1/\Delta t \tag{1.1}$$

This is also described as flat-fading channel because the same complex attenuation can used across frequencies employed by transmission and is consequently flat. So the elements in the corresponding flat fading channel matrix $\mathbf{H} \in \mathbb{C}^{n_r \times n_t}$ contain the complex attenuation from each transmitter to each receiver. So the received signal $\mathbf{z}(t) \in \mathbb{C}^{n_r \times 1}$ as a function of time is given by

$$\mathbf{z}(t) = \mathbf{H}\ \mathbf{s}(t) + \mathbf{n}(t) \tag{1.2}$$

where the transmitted signal vector and additive noise as a function of time are denoted as $\mathbf{s}(t) \in \mathbb{C}^{n_t \times 1}$ and $\mathbf{n}(t) \in \mathbb{C}^{n_r \times 1}$ respectively.

If we consider a block of data with $n_s$ samples, the received signal for a block of data with $n_s$ samples is given by

$$\mathbf{Z} = \mathbf{H.S+N} \tag{1.3}$$

where $\mathbf{Z} \in \mathbb{C}^{n_r \times n_s}$ is the received signal, $\mathbf{S} \in \mathbb{C}^{n_t \times n_s}$ is the transmitted signal and $\mathbf{N} \in \mathbb{C}^{n_r \times n_s}$ is the noise excluding the interference signal. It is implied that the channel remains static for atleast $n_s$ samples

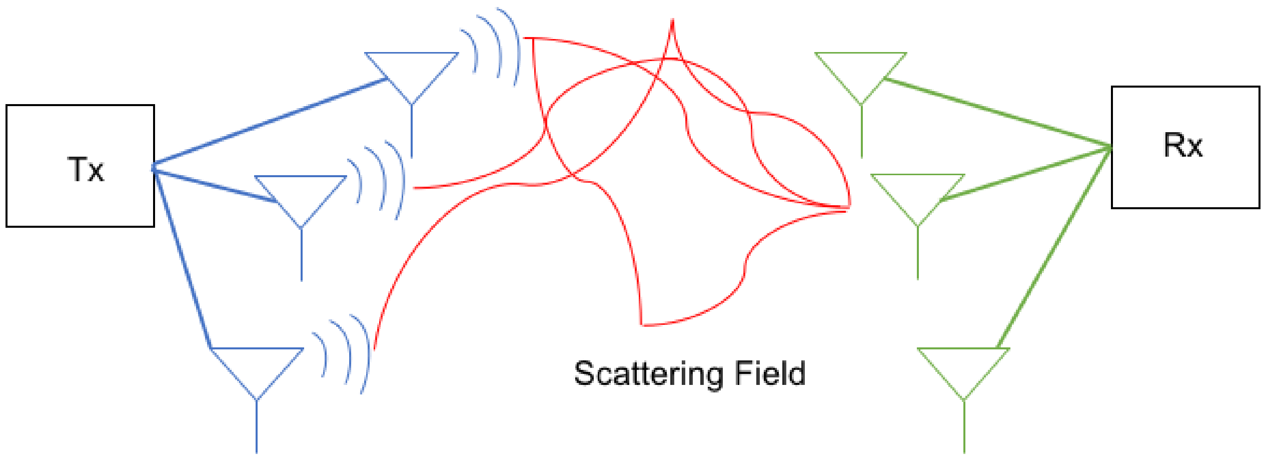


**Figure 1.4:** MIMO Wireless Communication Link, Adapted from [4]

### *1.4.2 Interference*

External Interference or the interference from other communication links was easily avoided by employing frequency-division multiple access (FDMA). But the recent surge in the use of wireless communications, external interference has become an increasing problem. Here interference is defined as external interference and interference from a communication system's own transmitters is typically defined as internal interference. In a MIMO system this is of particular interest as signals from multiple transmitters of a signal node typically interfere with each other at the receiver. In the industrial, scientific and medical(ISM) bands there no regulation and the frequency bands are loosely controlled. Hence various communication systems compete in this limited spectrum.

By decreasing sensitivity to interference in these networks, higher link densities and higher signal-to-noise ratio (SNR) links can be achieved which can lead to increase in the network throughput.

To describe the effects of interference, two essential characteristics need to be specified: the channel and knowledge of the interference waveform.

The received signal given be Equation 1.3 is further expressed as :

$$= \mathbf{H}.\mathbf{S} + \sum_{m} J_m T_m + \tilde{N} \tag{1.4}$$

where the interference contained within $\mathbf{N}$ is expressed in the sum of the terms $J_m T_m$. The term $\tilde{N}$ is the remaining thermal noise. The interference channels $J_m$ are typically statistically equivalent to those for the signal of interest of the channel $\mathbf{H}$.

Secondly the nature of the external interference can have a significant effect on a communication system's performance. For example, if the receiver is aware of the interference signal and its channel $\mathbf{JT}$ then the interference has no effect as the receiver can subtract the contributions of the interference from the received signal perfectly. From a practical standpoint the effect can known interference can be minimized at the receiver by projecting onto a basis temporally othogonal to the interfering signal,

$$\begin{aligned} \tilde{Z} &= Z.P_T^{\perp} \\ P_T^{\perp} &= I - T^{\dagger}(TT^{\dagger})^{-1}T \end{aligned} \tag{1.5}$$

When the number of samples,n$_s \gg$ $\mathbf{T}$, the loss associated with this projection approaches zero because the size of total signal space is $n_s$. As the size of transmitted signal subspace is fixed, the fraction of potential signal subspace that is subtended by the projection operator goes to zero as $n_s$ becomes large.

## 1.5 Spatially Adaptive Receiver

An adaptive receiver is defined as characteristic of being aware of the environment and adjusting the behaviour to improve the typical performance. The flexibility of an adaptive receiver is typically limited to estimating parameters used in its algorithms at the receiver. In the Figure 1.5, a typical communication chain is shown, the information is encoded and modulated. In this system a sampled approach a sampled

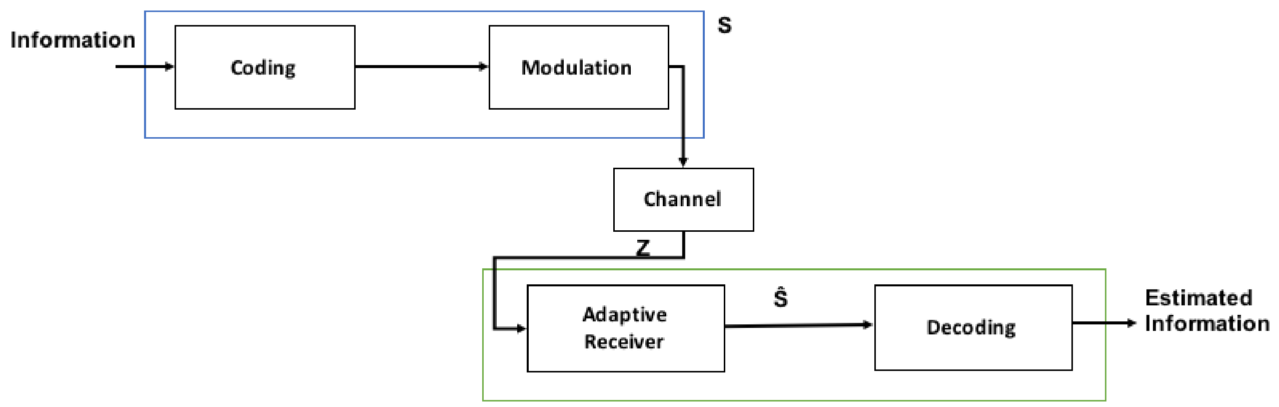


**Figure 1.5:** Communication Chain, Adapted from [4]

approach is employed,the transmitted signal $\mathbf{S} \in \mathbb{C}\mathrm{n}_t \times n_s$ where $n_t$ is the number of transmitters by the number of samples $n_s$.The signal **S** is represented at the complex base band. At the receiver the transmitted signal corrupted by the channel which is observed by $n_r$ receive antennas, its represented by $\mathbf{Z} \in \mathbb{C}\mathrm{n}_r \times n_s$.

The known signal at the receiver which is normalized version of the transmitted signal $\mathbf{X} \in \mathbb{C}\mathrm{n}_t \times n_s$. The normalization $\|X\|_F^2 = n_t \times n_s$ and the transmitted signal matrix S is given by

$\mathbf{S} = \sqrt{\mathrm{P}_o / n_t}\ \mathbf{X}$ where $\mathrm{P}_o$is the total noise-normalized power.

It is common for the communication system to transmit a predefined signal for a portion of the transmission frame. This portion transmitted signal is known as the pilot sequence which helps in the construction of the beamformer at the receiver end to extract the estimate of the transmitted signal in a region of time near the pilot sequence.

The following chapters will focus on the adaptive processing of receivers based on the knowledge of the pilot sequence which is also known as the training sequence.

## 1.6 Problem Statement and Approach

Although temporal interference mitigation is well defined algorithm that can be applied on adaptive receivers in RF systems, the existing implementation has only been limited to simulations. Having a hardware implementation will make this algorithm applicable directly in the field. As we existing RF systems can easily support embedded platform for computation there is strong motivation to have this algorithm implemented on actual hardware. An Field Programmable Gate Array (FPGA) will not only serve as proof of concept for hardware implementation but will also be much more realistic implementation that can easily be lifted to various related and similar platforms. Our goal is to have a hardware implementation that is mapped unto an embedded platform such as an FPGA with the necessary computationally intensive functions accelerated and optimized.

Our approach to implementation is three fold process. First, we identify the intermediate operations involved in the algorithm and individually map each of them to Programmable Logic so that they can hardware accelerated. Then we identify the bottlenecks in each individual operations and attempt to optimize them individually. Lastly we validate overall result of our algorithm against an existing MATLAB simulation output.

Once we performed the implementation we also perform an extensive analysis of precision range of our hardware implementation. The analysis involves implementing all of the individual operations which are defined as individual accelerators against a range of fixed point data types. The purpose of this analysis is to examine the effect of precision of data type on performance(in terms of computation time), accuracy(overall output) and resource usage. Another important element of this analysis is error calculation framework that calculates the precision output errors of all the individual operations to determine an optimal combination of accelerators with hybrid set of data types. We also test our algorithm against both realistic and idealistic input sets with series of interference levels. The ultimate objective of the above implementation and analysis is to present a qualitative and quantitative report on behaviour of individual components and their response to various tuning parameters such as data type and interference levels.

## 1.7 Contribution

This study introduces all of relevant RF convergence concepts and provides a detailed description of adaptive receivers which are an integral element of any RF system. Further we also explain the role of Multiple Input Multiple Output (MIMO) systems and how they introduce internal interference in RF systems. The report describes the mitigation strategies to remove this interference and also discusses the various estimation approaches the receiver applies to combat with interference based on the knowledge of the signal obtained from a transmitter. Temporal Interference Mitigation is introduced and described with both an analytical and mathematical formulation. Subsequently we delve in the existing hardware architectures that are suitable for our mapping purposes. We examine our target architecture FPGAs in detail and also discuss our High-Level Synthesis based hardware development. As our target device is an System on Chip (SoC), we examine the layout and structure of SoCs. The approaches used for SoC based system development are discussed in light of our temporal mitigation algorithm. Each of the individual operations are identified as compute intensive kernels and them mapped onto programmable logic. Following our hardware mapping we implement optimization strategies that benefit us on both an algorithmic and system development front. Once we completed the hardware implementation and optimization we begin with our analytical study. The analysis consists of usage of fixed point data type for each of the hardware accelerated kernels. Further an error computation model is proposed and implemented to calculate the precision error of each of the individual kernels based on fixed point data type being used. The effect of fixed point data type is studied through a series of experiments. We test the performance, accuracy and resource usage of each of various data types being used. Finally we propose a hybrid model that contains an optimal combination of kernels which is derived based on the error performance of each of the individual kernels.

Chapter 2

# ADAPTIVE RECEIVERS IN RF COOPERATION BASED SYSTEMS

This chapter and the following sections will describe the details and process used by adaptive receivers to extract known estimates out of the received signal to mitigate the interference.

## 2.1 Spatially Adaptive Receivers

By using the sampled flat-fading MIMO channel model with $n_t$ transmitter by $n_r$ receivers can be given by two forms depending on whether the power parameter is absorbed into the transmitted signal or the channel matrix. In case of transmitted signal the equation is :

$$\begin{aligned} \mathbf{Z} &= \mathbf{HS} + \mathbf{N} \\ &= \mathbf{H}\ \sqrt{P_o/n_t}\ \mathbf{X} + \mathbf{N}; \mathbf{S}{=}\sqrt{P_o/n_t}\mathbf{X} \end{aligned} \tag{2.1}$$

And in case of channel matrix :

$$= \mathbf{A}\ \mathbf{X} + \mathbf{N}\ ;\ \mathbf{A} = \sqrt{P_o/n_t}\ \mathbf{H} \tag{2.2}$$

where the received signal is indicated by $\mathbf{Z} \in \mathbb{C}^{n_r \times n_s}$, the channel matrix is indicated by $\mathbf{H} \in \mathbb{C}^{n_r \times n_t}$, the transmitted signal is $\mathbf{S} \in \mathbb{C}^{n_t \times n_s}$ and the amplitude-channel product is given by $\mathbf{A} \in \mathbb{C}^{n_r \times n_t}$ and the normalized transmitted signal is given by $\mathbf{X} \in \mathbb{C}^{n_t \times n_t}$ and the total thermal noise normalized power is indicated by $P_o$.

By using a linear operator, an estimate of the normalized transmitted signals $\hat{X}$ is given by

$$\hat{X} = W^{\dagger} Z \tag{2.3}$$

where the columns of the beam forming matrix $\mathbf{W}$ contain a beam former associated with a particular transmitter. The complex coefficients within each column are conjugated and multiplied by data sequences from each data stream corresponding to the receive antenna. These modified data streams are used to then get summed up to attempt to reconstruct the signal associated with a given transmitter. The process of adaptive spatial processing at the receiver is sometimes referred to as spatial filtering as well because of the strong connection between spatial processing and spectral processing. The spatial locations of the antennas correspond to the delay taps in the spectral filter. The spatial direction corresponds to the frequency.

There are many different approaches to the receive beamformer and a few of them will be discussed.

### *2.1.1 Spatial Matched Filter*

The matched filer concept is used to construct a beamformer that maximizes the receive power to thermal noise ratio associated with a particular transmitter. The beamformer that maximizes this power has a structure that is matched to the received array response for a particular transmitter. This is sometimes denoted a maximum ratio combiner (MRC). In the case of a line-of-sight environment, the filter corresponds to the steering vector for the direction to the particular transmitter. In this approach the beamformer for each transmitter is constructed individually. The beamformer for the mth transmitter is $\mathbf{w}_m \in \mathbb{C}^{n_r 1}$, and the channel between the mth transmitter and the receive array is given by $\mathbf{a}_m \in \mathbb{C}^{n_r X 1}$ . The transmit sequence from the mth transmitter is given by $\underline{\mathbf{x_m}} \in \mathbb{C}^{1 n_s}$ . The received signal matrix $\mathbf{Z}$ is given by

$$\begin{aligned}\mathbf{Z} &= \mathbf{A}\ \mathbf{X} + \mathbf{N} \\ &= \sum_m a_m \underline{\mathbf{x_m}} + \mathbf{N}\end{aligned} \tag{2.4}$$

When the expectation operator is applied on the Equation 2.3 using Equation 2.4 the beamformer coefficients are found to be

$$\mathbf{w}_m = \frac{a_m}{\|a_m\|} \tag{2.5}$$

and the maximum-likelihood estimate of amplitude channel vector $a_m$ under the Gaussian model is given by

$$\hat{a}_m = \frac{Z P^{\perp}_{X^*_m} \underline{x}^{\dagger}_m}{\underline{x}_m P^{\perp}_{X^*_m} \underline{x}^{\dagger}_m} \tag{2.6}$$

where the projection operator $P^{\perp}_{X^*_m}$ is orthogonal to the row space of $X^*_m$ and is defined by

$$P^{\perp}_{X^*_m} = I - X^{\dagger}_{m*}(X_{m*}X^{\dagger}_{m*})^{-1}X_{m*} \tag{2.7}$$

Here $X_{m*}$ is the transmit signal matrix without the transmitter of interest and $A_{m*}$ is the channel matrix without the transmitter of interest. The beamformer $w_m$ is given by

$$w_m = \frac{Z P^{\perp}_{X^*_m} \underline{x}^{\dagger}_m}{\|Z P^{\perp}_{X^*_m} \underline{x}^{\dagger}_m\|} \tag{2.8}$$

If the transmit sequences are orthogonal then $\hat{a}_m = \mathbf{Z}\underline{x}^{\dagger}_{m*}(\underline{x}_{m*}\underline{x}^{\dagger}_{m*})^{-1}$

### *2.1.2 Minimum Interference Spatial Beamforming*

The matched filter discussed in the previous section did not account for the presence of external interference or even the other transmit antennas. The minimum interference beamformers focus on the effects of these interferences.

The form of the minimum-interference beamformers vary based on the assumptions being used. The receiver may or may not assume that the external interference plus noise is uncorrelated. Also the number of transmit antennas may be larger than the number of receive antennas.When the number of receive antennas is equal to or larger than the transmit and interference sources and hence this beamformer has a convenient property that the signals associated with each beamformer output are uncorrelated. It is hence known as a deccorelating beamformer. The following sections will discuss some of the beamforming approaches used in minimum interference beamforming.

### *2.1.3 Channel Inversion*

This is a common beamforming approach is the channel inversion technique, which is also called zero-forcing receiver. This approach is spatial extension to the spectral zero-forcing equalizer and reconstructs the transmitted sequences exactly in the absence of noise if $n_r \geqslant n_t$ Here the beamformer is constructed $\mathbf{W}_{ZF} \in \mathbb{C}\mathbf{n}_r \times n_t$ using the psuedo inverse of the amplitude-channel product, revealing the transmit sequence corrupted by noise,

$$\begin{aligned} \mathbf{W}_{ZF} &= \mathbf{A}(\mathbf{A}^\dagger A)^{-1} \\ \hat{X} = W_{ZF}^\dagger Z &= (\mathrm{A}^\dagger A)^{-1} A^\dagger \mathrm{Z} \\ &= X + (\mathrm{A}^\dagger A)^{-1} A^\dagger \mathrm{N} \end{aligned} \tag{2.9}$$

The output of this beamformer under the assumption of perfect channel knowledge and atleast as many receive antennas as transmit antennas, has no contributions from other transmitters. The beamformer adapted for the $m$th transmitter $\mathbf{w}_m$ is expressed as

$$\begin{aligned}\mathbf{w}_m &= \mathbf{W}_{ZF}\mathbf{e}_m \\ &= A(A^\dagger A)^{-1}e_m\end{aligned} \tag{2.10}$$

where the selection vector $\{\mathbf{e}_m\}_n = \delta_{m,n}$ and is one if both m and n are equal and zero otherwise.

### *2.1.4 Orthogonal Beamformer*

In a scenario where there is a MIMO link with no external interference, the interference from other transmitters within the MIMO link [5],[6] is minimized by constructing a beamformer $\mathbf{w}_m$ for each transmit antenna that is orthogonal to the spatial subspace spanned by the spatial responses of the other transmitters,

$$\mathbf{w}_m \propto \mathbf{P}^{\perp}_{A^*_m} a_m \tag{2.11}$$

where $\mathbf{P}^{\perp}_{A^*_m} \in \mathbb{C}^{n_r \times n_r}$ is the operator that projects onto a column space orthogonal to the spatial response of the interfering transmitters. $\mathbf{P}^{\perp}_{A^*_m}$ is given by

$$\begin{aligned}P^{\perp}_{A^*_m} &= I - P_{A^*_m} \\ P_{A^*_m} &= A_{m*}(A^\dagger_{m*}A_{m*})^{-1}A^\dagger_{m*}\end{aligned} \tag{2.12}$$

By using reference signal,the beamformer $w_m$ can be estimated as

$$w_m \approx \frac{\hat{P}^{\perp}_{A_{m*}} Z P^{\perp}_{X^*_m} \underline{x}^\dagger_m}{\|\hat{P}^{\perp}_{A_{m*}} Z P^{\perp}_{X^*_m} \underline{x}^\dagger_m\|} \tag{2.13}$$

## 2.2 Minimum Mean Squared Error(MMSE) Spatial Receiver

There exists a strong similarity between the spectral Weiner filter and adaptive MMSE spatial filter. The error matrix in this case $\mathbf{E} \in \mathbb{C}^{n_t \times n_s}$ at the output of the beamformer $\mathbf{W} \in \mathbb{C}^{n_r n_t}$ is

$$\mathbf{E} = \mathbf{W}^\dagger\ \mathbf{Z}\ \text{-}\ \mathbf{X} \tag{2.14}$$

when we apply the MMSE on the above equation we obtain the following relationship

$$\begin{aligned}\langle \mathbf{Z}\ \mathbf{Z}^{\dagger}\rangle W - \langle ZX^{\dagger}\ \rangle = 0 \\ W = \langle \mathbf{Z}\mathbf{Z}^{\dagger}\rangle^{-1}\langle ZX^{\dagger}\ \rangle\end{aligned} \tag{2.15}$$

The above form has an intuitive interpretation. The first term is propotional to the receive co variance matrix $\mathbf{Q} \in \mathbb{C}^{n_r \times n_r}$ and the second term is proportional to the an array response estimator. Hence this beamformer points to the direction of signals of interest, and points away from the interference sources. With the assumption that the transmit covariance matrix is proportoinal to identity matrix (which also implies the MIMO link is operating in uninformed mode) $\langle XX^{\dagger}\ \rangle$ =n$_s$I, and the cross covariance is proportional to the channel $\langle ZX^{\dagger}\rangle$ =n$_s$A, For practical purposes the expectations can be approximated over a finite number of samples $n_s$. If $n_s \gg$n$_r$ and $n_s$ $\gg$n$_t$ then the expectations can be expressed as,

$$\begin{aligned}\langle ZZ^{\dagger}\rangle \approx ZZ^{\dagger} \\ \langle ZX^{\dagger}\rangle \approx ZX^{\dagger}\end{aligned} \tag{2.16}$$

By using the above relationships the MMSE beamformers in columns of **W** is given by

$$W \approx (ZZ^{\dagger})^{-1}ZX^{\dagger} \tag{2.17}$$

Chapter 3

# MULTIPLE ANTENNA MULTIPLE USER RECEIVER

This chapter describes the details associated with a Multiple Antenna Multiple User Receiver which chiefly employs two techniques to make a Cooperation based RF system possible:

1. Spatial beam forming
2. Temporal Projection

The previous chapter has already touched on the topic of beamforming and discussed the relavant techniques employed by the receiver. There is also temporal aspect that will be discussed and the relavant processes involved will be outlined. Both of these techniques or processes are employed with a same objective of mitigating and eliminating interference to recover the signal of interest.

We need to highlight that the multiple users above are transmitting simultaneously in the same band at the same time. The transmitters are using spreading sequences to reduce the multiple-access interference that is introduced in the above setup. This interference can be introduced by multiple antennas on the same transmit node, or from multiple nodes in a network. Another feature of the receivers in the above setup is that the separation in temporal structure between users is exploited in addition to the differences in spatial responses. There is an underlying assumption that these systems are employing a direct-sequence spread-spectrum technique.

In case of multiple-user receivers, these receivers are implemented as iterative receivers in which the receiver operates on the same block of data multiple times.

Although iterative receivers are not linear, they employ a linear operator applied to some space.

The separation between various receive states is separated by a hyper plane in some high-dimensional space.

## 3.1 Maximum-Likelihood Demodulation

As introduced in Equation 2.12, the matrix :

$$\begin{aligned} P_X^{\perp} &= I_{n_s} - P_X \\ P_X &= X^{\dagger}(XX^{\dagger})^{-1}X \end{aligned} \tag{3.1}$$

given that $\mathbf{P}_X^{\perp}$ projects onto the orthogonal complement of the row space of X. The determinant $\mathbf{Z}P_X^{\perp}Z^{\dagger}$ is minimized to demodulate the signals for all the transmitters jointly.

By maximizing the log of the probability distribution with respect to some arbitrary parameter of the channel matrix the estimate of the channel $\hat{A}$ is found,

$$\hat{A} = ZX^{\dagger}(XX^{\dagger})^{-1} \tag{3.2}$$

The probability density function is given by

$$p(\mathbf{Z}|\ X; R, \hat{A}) = \tfrac{1}{|R|^{-n_s}\pi^{n_s n_r}} e^{-tr\{(ZP_X^{\perp})^{\dagger}R^{-1}(ZP_X^{\perp})\}} \tag{3.3}$$

When we maximize the probability density with a nuisance parameter $\mathbf{A}$, for an arbitrary parameter of the interference plus noise co variance matrix $\mathbf{R}$ the estimate is given by

$$\hat{R} = \frac{1}{n_s} ZP_X^{\perp}Z^{\dagger} \tag{3.4}$$

When we substitute this result into the probability density, only the received data matrix and the transmitted signals are left

$$\log p(Z \mid X; \hat{R}, \hat{A}) \propto |Z P_X^{\perp} Z^{\dagger}|^{n_s} \tag{3.5}$$

Thus maximizing of the likelihood function is equivalent to minimizing $|Z P_X^{\perp} Z^{\dagger}|$.

For the purpose of demodulation we can make use of an iterative receiver that cycles through the rows of the transmitted signal matrix **X** Some of the details in the derivation have been omitted but the relavant results are:

$$Z P_X^{\perp} Z^{\dagger} = Z P_{\underline{X}_U}^{\perp} Z_U^{\dagger} \tag{3.6}$$

where $\mathbf{U} \in \mathbb{C}^{(n_t-1)\times n_s}$

And the corresponding beam forming terms are :

$$\begin{aligned} \mathrm{w} &= \hat{R}_U^{-1} \hat{a} \\ {}_U &= \frac{1}{n_s} Z P_{X_{\cancel{m}}}^{\perp} Z^{\dagger} \\ &= Z P_{X_{\cancel{m}}}^{\perp} \underline{x}^{\dagger} (\underline{x} P_{X_{\cancel{m}}}^{\perp} \underline{x}^{\dagger})^{-1} \end{aligned} \tag{3.7}$$

The $n_r \times 1$ vector **w** contains the receive beamforming weights, $\hat{R}_U$ is the interference mitigated signal plus noise co variance matrix estimate, and  is the channel estimate associated with x̲ with $\mathbf{X}_{\cancel{m}}$ mitigated temporally. Demodulation is performed by maximing the inner product of the beamformer output, $\mathbf{w}^{\dagger} Z_U$ and the interference-mitigated reference signal, $\underline{x}_U$

Once we have established the maximum-likelihood demodulation setup we can perform and channel estimation and interference subtraction. The following section talks about how we can make use the channel estimate that was computed in the previous section to perform interference subtraction and mitigation consequently.

Another point worth mentioning is the Multiple Antenna Multiple user receiver performs the receive process in an iterative fashion where we begin the iteration with estimates for the signals in the $\mathbf{X}_{\not{m}}$ and then the receiver processing gives us an estimate of the $m^{th}$ signal $\underline{x}$ After we have estimated $\underline{x}$ we can move on to the next signal of interest and $\underline{x}$ is now considered interference. Hence iterating over all the signals of interest will lead to improvements in all of the estimates.

The following sections talk about how the knowledge of channel estimate can be used to perform interference subtraction and estimate our signal of interest.

## 3.2 Channel Estimation

As we obtained from Equation 3.2 the channel estimate

$$\hat{A} = ZX^{\dagger}(XX^{\dagger})^{-1} \tag{3.8}$$

where Z$\in \mathbb{C}^{n_r \times n_s}$ is the matrix of the received signals and X $\in \mathbb{C}^{n_t \times n_s}$ is the matrix of the transmitted signals. At the receiver the term

$$\mathbf{Z}\mathbf{X}_{\not{m}}^{\dagger}(X_{\not{m}}X_{\not{m}}^{\dagger})^{-1} \tag{3.9}$$

is seen to the channel estimate for all the signals(interference signals) but not including the signal of interest $\underline{x}$. Hence

$$\mathbf{Z}\mathbf{X}_{\not{m}}^{\dagger}(X_{\not{m}}X_{\not{m}}^{\dagger})^{-1}X_{\not{m}} \tag{3.10}$$

is an estimate for the interference sensed at the receiver. So uptil now we have performed the channel estimation at the receiver but using the training sequence and the channel estimate.

The following section is going to talk about the topic of Self Interference Mitigation that draws concepts from channel estimation and projections to orthogonal basis. The underlying assumption here is the MIMO self-interference channel estimation is the same as discussed in section 3.10. We also assume that we can make use of the iterative estimator to estimate the non-linear coefficients.

The iterative estimator performs the following steps :

$$\begin{aligned}
X_3^{(1)} &= X^{\{3\}} \\
&\quad \textit{repeat} \\
P_{\{3\}}^{\perp,(m)} &= I - (X_3^{(m)})^\dagger[(X_3^{(m)})(X_3^{(m)})^\dagger]^{-1}(X_3^{(m)}) \\
(X^{(m)})^\dagger &= XP_{\{3\}}^{\perp,(m)} \\
\hat{A}^{(m)} &= Z(\tilde{X}^{(m)})^\dagger[(\tilde{X}^{(m)})(\tilde{X}^{(m)})^\dagger]^{-1} \\
X_3^{(m+1)} &= \begin{pmatrix} X^{\{3\}} \\ (\hat{A}^{(m)}S)^{\{3\}} \end{pmatrix}
\end{aligned} \tag{3.11}$$

Here the $m^{th}$ estimate of the non linear contributions is indicated by $X_3^{(m)}$, and the $m^{th}$ estimate of the transmitted signal projected onto the orthogonal basis is $\tilde{X}^{(m)}$, and of the $m^{th}$ self-interference channel estimate is indicated by $\hat{A}^{(m)}$. The iterative approach is convergent and converges quickly. The plot in Figure 3.1 shows this trend more explicitly and was performed for a 4 antenna system with passband nonlinearities in the transmitter with a gain of -40 dB and receiver gain -60 dB, and a 130 dB self interference.

The following section details the temporal self-interference mitigation and corresponding estimation and subtraction employed.

## 3.3 Temporal Self-Interference Mitigation

The concept used for temporal mitigation is to project on to a basis orthogonal to the interference similar to the approach used in the previous sections. This results in a form that can be interpreted as estimation and subtraction. The temporal space

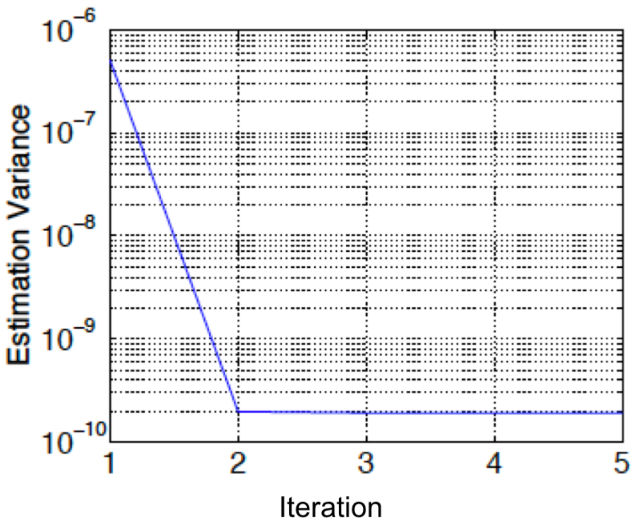


**Figure 3.1:** Iterative Convergence for Self-Interference Channel Estimation Variance

that needs to be removed occupies the row space of

$$\check{X} = \begin{pmatrix} X \\ X^{\{3\}} \\ [\check{A}X]^{\{3\}} \end{pmatrix} \in \mathbb{C}^{(3N)\times n_s} \tag{3.12}$$

under a 3rd order approximation. The self-interference mitigation version of the data $\check{Z} \in \mathbb{C}^{N\times n_s}$ is expressed as

$$\begin{aligned} \check{Z} &= ZP^{\perp}_{\check{S}} \\ P^{\perp}_{\check{S}} &= I_{n_s} - \check{X}^{\dagger}(\check{X}\check{X}^{\dagger})^{-1}\check{X} \end{aligned} \tag{3.13}$$

There are a couple of ways to evaluate $\check{Z}$. The estimation and subtraction can be used to evaluate $\check{Z}$ as follows

$$\hat{Z} = ZP^{\perp}_{\check{S}} = Z - (Z\check{X}^{\dagger})(\check{X}\check{X}^{\dagger})^{-1}\check{X} \tag{3.14}$$

### *3.3.1 Estimation and Subtraction*

From Equation 3.14 we observe that the projection operation can be interpreted as estimation and subtraction, $\hat{Z} = Z - \underbrace{(Z\check{X}^{\dagger})(\check{X}\check{X}^{\dagger})^{-1}}_{\check{A}}\check{X}$

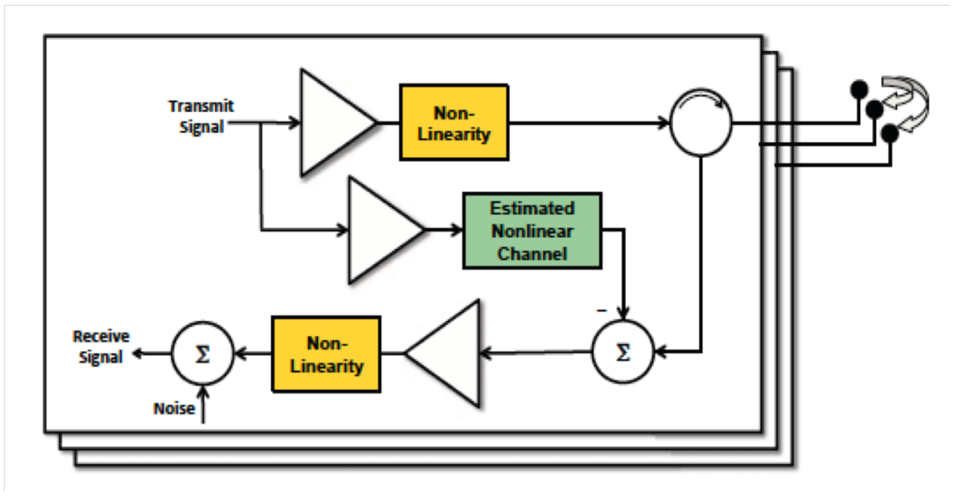


**Figure 3.2:** Channel Estimation Using Subtraction Mitigation at Passband

where

$$\check{H} \equiv (Z\check{X}^{\dagger})(\check{X}\check{X}^{\dagger})^{-1} \in \mathbb{C}^{N\times(3N)} \tag{3.15}$$

is an estimate of the nonlinear channel as seen at complex baseband. In the above form the nonlinear channel can be estimated at one point in time and then applied at a later time as shown in the Figure 3.2

### *3.3.2 Real Space Self Interference Mitigation*

The interference mitigation approach can be used to increase the dimensionality by introducing a real space of parameters. The temporal projection operation is given by

$$\begin{aligned}\begin{pmatrix}\Re\{\check{Z}\}\\ \Im\{\check{Z}\}\end{pmatrix} &= \begin{pmatrix}\Re\{Z\}\\ \Im\{Z\}\end{pmatrix}\mathbf{P}^{\perp}_{\check{S}}\\ \mathbf{P}^{\perp}_{\check{S}} &= I - \check{X}^{\dagger}(\check{X}\check{X}^{\dagger})^{-1}\check{X}\end{aligned} \tag{3.16}$$

where the temporal space is now spanned by

$$\hat{S} = \begin{pmatrix} \Re\{S\} \\ \Im\{S\} \\ (\Re\{S\})^{(3)} \\ (\Im\{S\})^{(3)} \\ (\Re\{[\hat{A}X]\})^{\{3\}} \\ (\Im\{[\hat{A}X]\})^{\{3\}} \end{pmatrix} \in \mathbb{R}^{(6N)\times n_s} \tag{3.17}$$

The following chapters will now focus on the temporal interference equation that was computed in Equation 3.14 . They will be about the methods and approaches used to implement the Equation 3.14 on hardware and the corresponding results obtained.

Chapter 4

# HARDWARE METHODS

The previous chapters discussed the concepts related to RF Convergence and how temporal mitigation stems as an important element and an enabler for most of the RF convergence based system.

This chapter will explain the hardware methodology and techniques adopted to implement temporal interference mitigation onto hardware. Before we dive into the methods being used we need to highlight a few details about the reasons why a hardware implementation is required.

As observed from the previous chapters, RF convergence can serve as very valueble solution to the problem of spectral congestion and the industries and applications that can benefit from it are numerous.

One common feature among these applications that were listed in Chapter 1 was that most of them were embedded applications which required some degree of onboard computing to execute the required application. For example Automotive Radar and vechile-to-vechile communication systems which are an integral element of modern day smart cars, they don't have the liberty of unlimited computing support and require the computing to take place on board. Also commercial flight air traffic control systems require some form of on board computing that can be done in real time with limited compute resources.

One of the viable solutions to this problem is to attempt to implement RF based systems on an embedded platform that caters to the requirement of on board computing while offering real time results.

Among existing embedded systems the most tempting choice for past few years has been Field Programmable Gate Arrays (FPGAs). There are numerous reasons why they are better alternative over their counterparts but the most relevant factors will be discussed henceforth.

## 4.1 FPGA : Architecture and Features

FPGAs are integrated circuits that consist of a matrix of configurable logic blocks(CLBs) that are connected via programmable interconnects. The most important logic blocks found in FPGAs are :

1. Flip-Flops: They are binary shift registers used to synchronize logic and save logical states between clock cycles within an FPGA circuit. On the edge of every clock cycle, the flip-flop latches the value on its input and holds it until the next clock edge.

2. Lookup Tables(LUTs) : Most of the logic operations in CLBs are implemented using a small amount of Random Access Memory and LUTs. LUTs are predefined list of combinatorial logic operations(such as ANDs,ORs,XORs,etc.,) which store the output for these combinatorial logic functions for a combination of inputs.

3. Digital Signal Processing (DSP) slices and multipliers : While LUTs provide output for combinatorial logic, the corresponding circuitry gets extremely complicated(the number of gates increases at an exponential rate) if we perform multiplicative operations. Hence FPGAs provide DSP slices and multipliers that provide multiplier circuitry to save on the usage of LUTs and Flip-flops for math and signal processing applications.

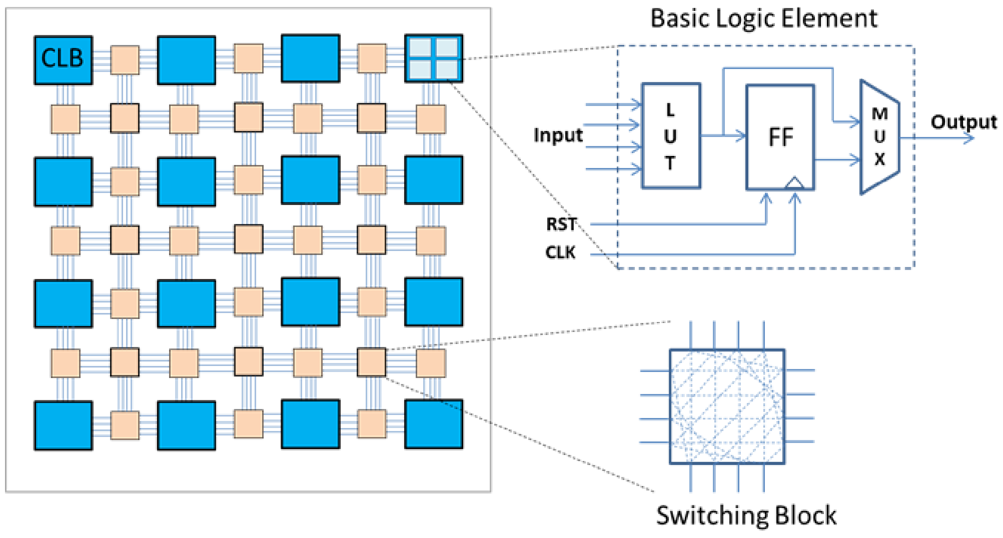


**Figure 4.1:** CLB Block of an FPGA

4. Block RAM: Block Random Access Memory(or BRAM) is a discrete processing part of FPGAs. It is useful for storing data sets or passing values between parallel tasks. Most signal processing algorithms need to keep track of an entire block of data and coefficients of complex equations. Without on board memory that is provided by BRAMs most of the processing functions would not fit within the configurable logic blocks.

The Figure 4.1 shows the CLB block[16] structure with the corresponding elements.

As we can observe from the previous list FPGAs have the following salient features:

1. Re-configurable Architecture : FPGAs provide a extremely re-configurable architecture [13] which can configured off the shelf. The re-configurable ability lets the programmer reprogram FPGA based on the user requirement. There is an extremely degree of flexibility which helps reduce the cost as compared to ASICs and other couterparts which cannot be reprogrammed.

2. Parallel Computation : FPGAs provide a multiple instruction multiple datapath (MIMD) computation capability. This is an extremely valuable aspect as there is a high degree of inherent parallelism. One of the most important benefits of MIMD is the fact that it can provide both data and instruction level parallelism. Even existing Graphic Processing Units (GPUs) can only provide Single Instruction multiple datapath(SIMD) which implies they dont have instruction level parallelism. Hence FPGAs can be exploited for their MIMD architecture while they are also much more cost efficient as compared to GPUs

3. Power Efficient : One of the most important factors that is often ignored in modern day computation devices is power efficiency. FPGAs besides having the features described above also cater to the need of power efficient computation.

The above factors have played a pivotal role in the selection of FPGAs for the hardware implementation. The following section will talk about the FGPA implementation approaches adopted and the corresponding reason for the same.

## 4.2 Hardware Methods Used

For FPGA hardware development there are multiple approaches one can take to develop and implement. There are varying level of abstractions one can use such as gate level, register-transfer level(RTL) and algorithmic level. While gate level requires a huge amount of development time for simple computations as the programmer is using very low level of design abstraction. Register-transfer level is a design abstraction that models the circuitry in terms of data transfer between registers. This design abstraction is modelled commonly using Verilog and Very High Speed Integrated Circuit Hardware Description Language,VHDL. Although this has been a viable approach for a long time there are certain aspects of Hardware Description Languages (HDLs) that need to looked at and inspected. HDLs were initially used for Application Specific Integrated Circuits (ASICs) and hence require a large degree of customization time to program even FPGAs. Further there is no optimization whatsoever offered for naive code that is being used for HDLs. Also the existing developments in FPGA architecture and design have made it extremely taxing to adapt and change the design everytime a different FPGA board is used. This requires specific optimization expertise and need high degree of fine tuning depending on the target board being used. Also in terms of code level optimization strategies, the programmer has the task of hand-coding the pipeline. This leads to ripple effects in the whole circuitry every time a small tweak is made. Further existing HDL based dependence has a heavy dependence on manufacturers Intellectual Property(IP) and a custom IP requires an extensive development period.

The above factors have led to the algorithmic level development of FPGAs. In this regard a number of approaches have emerged and High-level Synthesis(HLS) is one of the most common approaches. The following section will discuss details about HLS and its relevance to our implementation.

## 4.3 High-Level Synthesis

High-level synthesis(HLS) [15] is an automated design process that interprets an algorithmic description of a desired behaviour and creates digital hardware that implements that behavior. Synthesis begins with a high-level specification of the problem, where behavior is generally decoupled from clock-level timing. The code is analyzed, architecturally constrained and scheduled to transcompile into a register-transfer level(RTL) design in hardware description language(HDL). HDL is synthesized to gate level by the use of a logic synthesis tool. For our implementation we use Xilinx Vivado High-Level Synthesis(HLS). Vivado HLS [19] allows for the implementation of computationally intensive software algorithms by creating abstraction of algorithmic description, data type specification and interfaces. It enables accelerated IP creation by enabling C,C++ and System C specifications directly targeting FPGAs.

## 4.4 Software Defined System on Chip

The previous section introduced the concept of High level synthesis and how it can help with hardware mapping for our purposes. For the purpose of our implementation, Xilinx UltraScale+ MPSOC ZCU102 board was targeted. As we can observe from Figure 4.2 that FPGA contains two primary components Processing System (PS) and the Programmable Logic(PL). The PS contains the various processors that are available within the Application, Real-time and Processing Unit. PS also contains DDR memory and high speed connect ports. The PS contains optimized cache memory that is suitable for various complex mathematical operations. The PL contains the BRAMs, DSP, LUTs and Flip-Flops as discussed above. They are connected through high speed interconnects. The adoption HLS to implement the temporal mitigation needs to consider the architecture that the application and is targeting and the application itself. Xilinx provides a environment known as Software-Defined System-on-Chip(SDSoC) environment that provides a full-system optimizing C/C++ compiler that performs cross compilation and linking of C/C++ functions into programmable logic(PL) fabric as well PS within a Xilinx FPGA. It also provides automated system connectivity generation by providing complete boot image firmware, operating system and application executable. SDSoC environment is used for hardware implementation of temporal mitigation because of the System on Chip(SoC) architecture of the target FPGA board.

The following section will explain why a system on chip based system is suitable for our hardware implementation purposes.

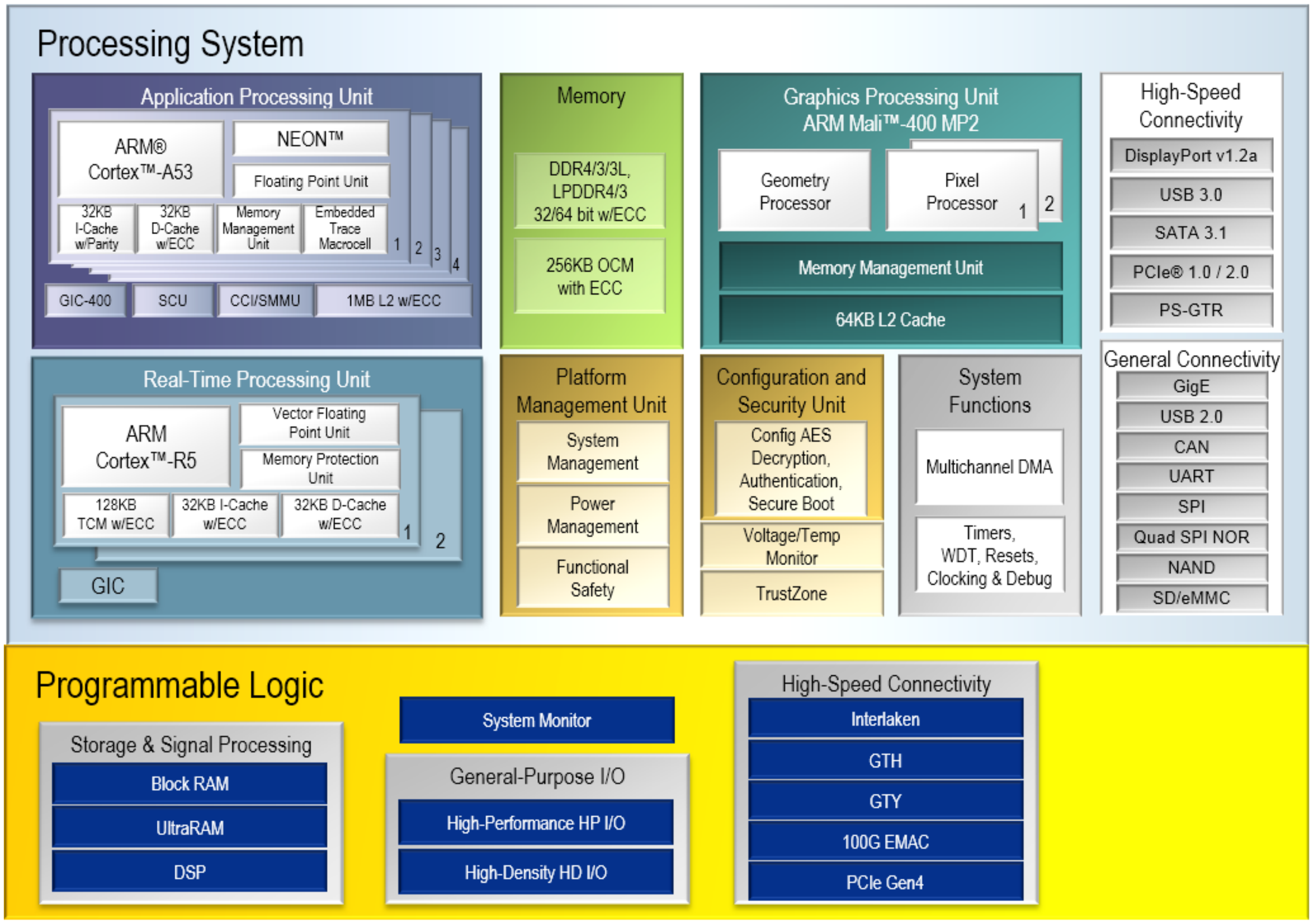


**Figure 4.2:** MPSOC Block Diagram [17]

## 4.5 System on Chip Based System Development

If we recall the temporal mitigation from Equation 3.14 it contains a series of complex linear algebra routines. It requires interfacing between processing system and the programmable logic for involved routines. This basically implies that the PS launches the routines to be executed on the PL and the data gets transferred through the memory access engines and the BRAM on the PL takes over. Figure

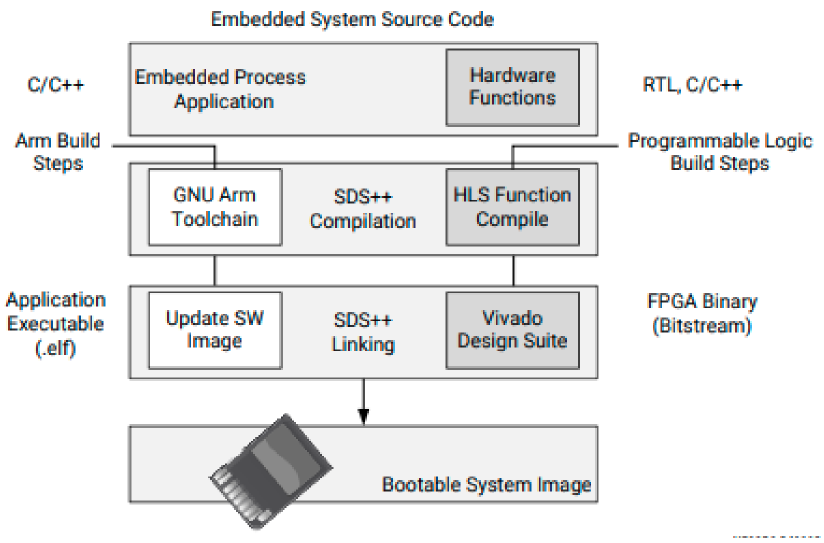


**Figure 4.3:** SDSoC Build Process [18]

4.4 shows the methodology used in SOC based architecture where the PS invokes the PL to execute various IP blocks. Here the IP blocks correspond to each intermediate step that needs to be accelerated. The Direct Memory Access engine serves as bridge to establish connection between the PS and the PL. As the operations involved in temporal mitigation involve complex linear algebra arithmetics. So the PS invokes the PL with every intermediate operation and then returns the result back to PS to proceed to next operation involved. The following section will talk in detail about every operation involved temporal mitigation and how its mapped onto hardware.

## 4.6 Hardware Implementation Initial Strategy

The equation 3.14 is defined as :

$$\hat{Z} = ZP_{\check{S}}^{\perp} = Z - (Z\check{X}^{\dagger})(\check{X}\check{X}^{\dagger})^{-1}\check{X} \tag{4.1}$$

It can be observed in the above equation that certain complex matrix algebra is being performed. The major operations that are observed are complex matrix multiplication,inversion,subtraction and conjugate transpose. So the corresponding operations

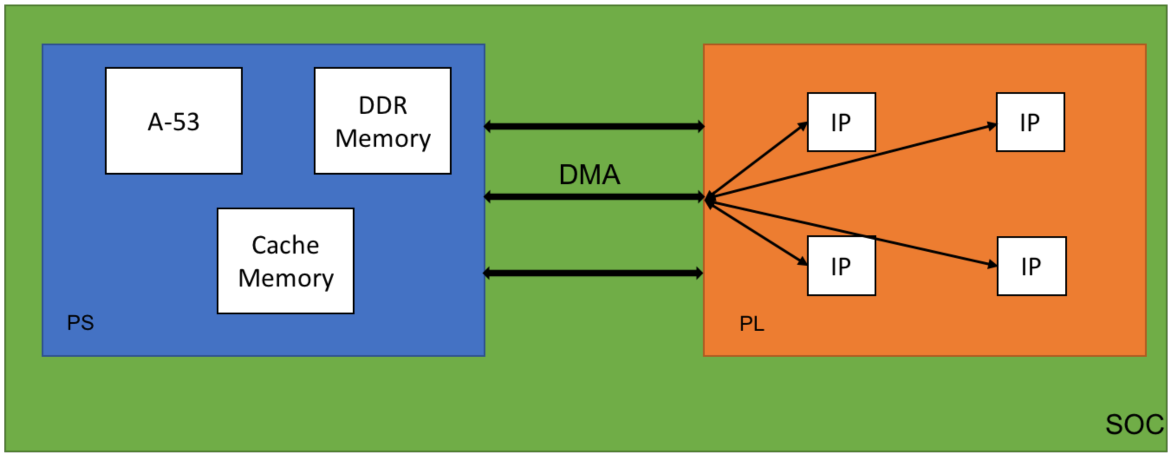


**Figure 4.4:** SOC Flow Process

are each mapped onto hardware as individual IPs which get launched from the PS onto the PL with the required input arguments. As shown in the previous sections the process by which the PS invokes the PL to execute each individual IP block that is essentially a hardware accelerated version of each function involved. For the purpose of our implementation the complex matrices $\mathbf{Z}$ and $\check{X}$ both have a dimension of $4 \times 64$. This corresponds to 4 receivers being present and the number of samples being processed is 64.

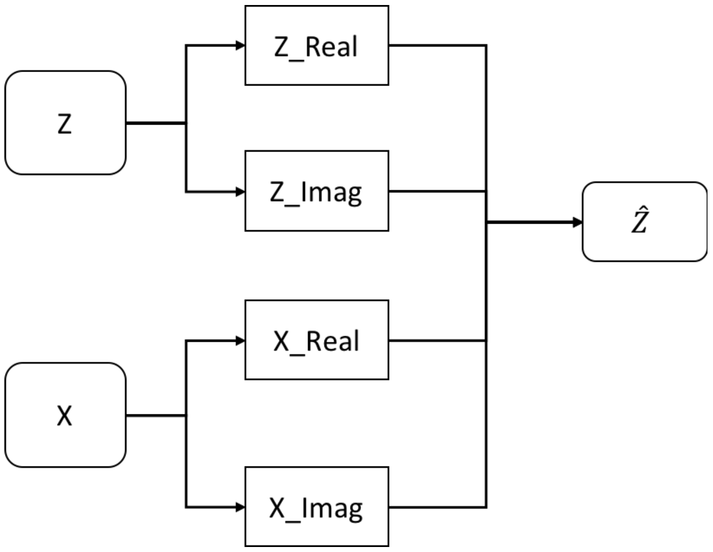


**Figure 4.5:** Implementation Strategy to Circumvent the Nature of Complex Matrices

Another important feature of the implementation was the requirement to accommodate the presence of complex matrices in the temporal mitigation equation. In terms of implementation, it would be very difficult to assign complex matrices unto hardware because of valid limitations. Hence we attempt to abstract this layer of detail(complex matrices) by treating the complex matrices as combination of two real matrices, while keeping track of the corresponding matrix algebra involved. The figure 4.5 exhibits this strategy where process the real and imaginary parts of the matrices separately and conveniently recombine them to complex matrices down the processing chain. Besides providing abstraction for hardware this also allows for parallel processing that we employ in the implementation chain.

The above sections have given a dive into the target architecture and how its awareness allows us to make valuable implementation decisions and design choices. The following chapters will explain in further detail the actual system level implementation with results.

# Chapter 5

# IMPLEMENTATION APPROACH

As discussed in the previous chapter the input signal for both the received signal $\mathbf{Z}$ and the matrix of known training sequence $\check{X}$ both have a dimension of $4 \times 64$ are both complex matrices. As discussed previously the major operations involved are complex matrix multiplication, inversion, subtraction and conjugate transpose.

Each of the above operations are implemented as independent IP block the details for which will be discussed henceforth.

## 5.1 Individual Operations

1. As we recall from linear algebra complex multiplication for two numbers is defined as follows :

   $$(a + ib).(c + id) = (a.c - b.d) + i(a.d + b.c) \tag{5.1}$$

   From the Equation 5.1 we can express a complex number as its real and imaginary part. The same formulation is scaled and used for matrices in our case.

2. As for matrix subtraction or addition, it follows the same rules as used for real addition and subraction

3. For conjugate transpose the following rule is used :

   $$(a + ib)^{\dagger} = (a)^{\dagger} + i(-b)^{\dagger} \tag{5.2}$$

   which is also scaled to complex matrices.

4. Inversion : This is a complicated task and gets even more complicated when it involves complex matrices. For implementation there were a couple of strategies adopted . In case of the prelimanary approach the following approach was used,

$$(A + iB)^{-1} = (A + B \times A^{-1} \times B)^{-1} - i(B + A \times B^{-1} \times A)^{-1} \tag{5.3}$$

The above equation can easily be mapped to hardware using the approach adopted in Figure 4.5 where we treat the real and imaginary parts independently.

The trend that is observed in all of the above operations is notion of treating the real and imaginary part of a complex matrix separately to provide abstraction for hardware as implementation of complex matrices is extremely taxing on hardware. This also makes parallel processing possible as the matrices can each be accelerated independently and recombined when required.

The previous section has detailed the mapping strategy used for each operation involved in temporal mitigation.

The following sections will discuss the details of each of the IP with their corresponding features.

## 5.2 List of Accelerators Required

Upon analyzing the operations in temporal mitigation equation :

$$\hat{Z} = Z - (Z\check{X}^{\dagger})(\check{X}\check{X}^{\dagger})^{-1}\check{X} \tag{5.4}$$

Its observed that there is sequence of operations that needs to applied to compute the final projected output $\hat{Z}$. The following list provides details about each of the operations with the corresponding of data required for them,

1. Conjugate Transpose: This operation requires the inversion of $4 \times 64$ matrix X which is performed using the method listed in above section. This operation can be implemented by just shifting the index of elements to obtain the transpose and the appropriately negating the imaginary part. It takes as input a two $4 \times 64$ matrices containing real and imaginary part respectively and returns two $64 \times 4$ matrices.

2. Matrix Multiplication: As matrix multiplication is not commutative and we are dealing with different dimensions of matrices, the accelerators have to be appropriately designed for required dimensions. The various orders of multiplication are $4 \times 64$ multiplied with $64 \times 4$, $4 \times 4$ multiplied with $4 \times 4$ and finally $4 \times 4$ multiplied with $4 \times 64$. The accelerators are designed and adjusted with above dimensions and the corresponding output dimensions.

3. Matrix Inversion: Inversion of complex matrix of size $4 \times 4$ is performed using the technique described in previous section. The inversion of real matrix is performed based on the adjugate formula.

4. Matrix Addition and Subtraction:This is performed using separate accelerators that satisfy the input and output dimensions of the matrices. The design has one accelerator that performs the final subtraction operation of two $4 \times 64$ matrices and one accelerator that is used for matrix addition (of two $4 \times 4$ matrices)as an intermediate step in inversion.

## 5.3 Preliminary Results

The initial implementation was based on the methods described in the previous section. Table 5.1 shows the performance and the speed up obtained on most expensive kernels(kernel refers to the operations which are accelerated). To obtain

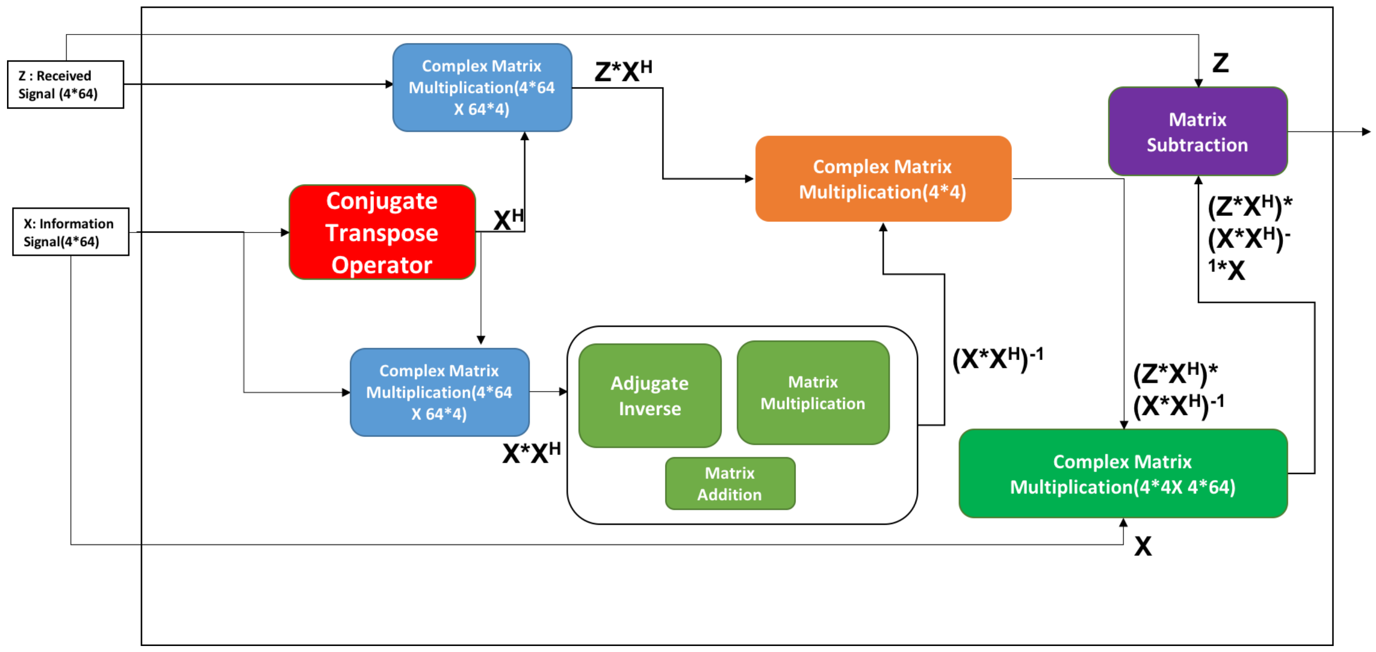

Z : Received Signal (4*64)
X: Information Signal(4*64)
Complex Matrix Multiplication(4*64 X 64*4)
Conjugate Transpose Operator
$X^H$
$Z*X^H$
Complex Matrix Multiplication(4*4)
Z
Matrix Subtraction
$(Z*X^H)*(X*X^H)^{-1}*X$
Complex Matrix Multiplication(4*64 X 64*4)
$X*X^H$
Adjugate Inverse
Matrix Multiplication
Matrix Addition
$(X*X^H)^{-1}$
$(Z*X^H)*(X*X^H)^{-1}$
Complex Matrix Multiplication(4*4X 4*64)
X

**Figure 5.1:** Flow Graph

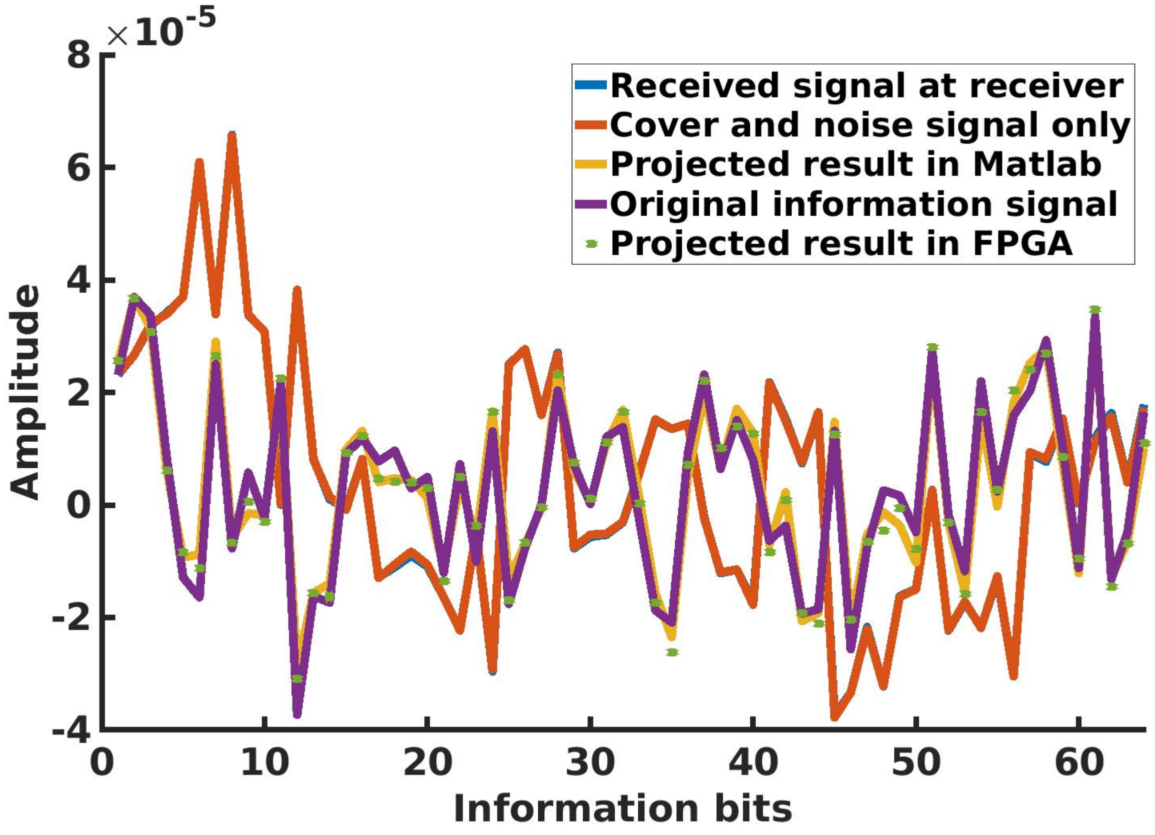


**Figure 5.2:** HLS Result Comparison with Matlab

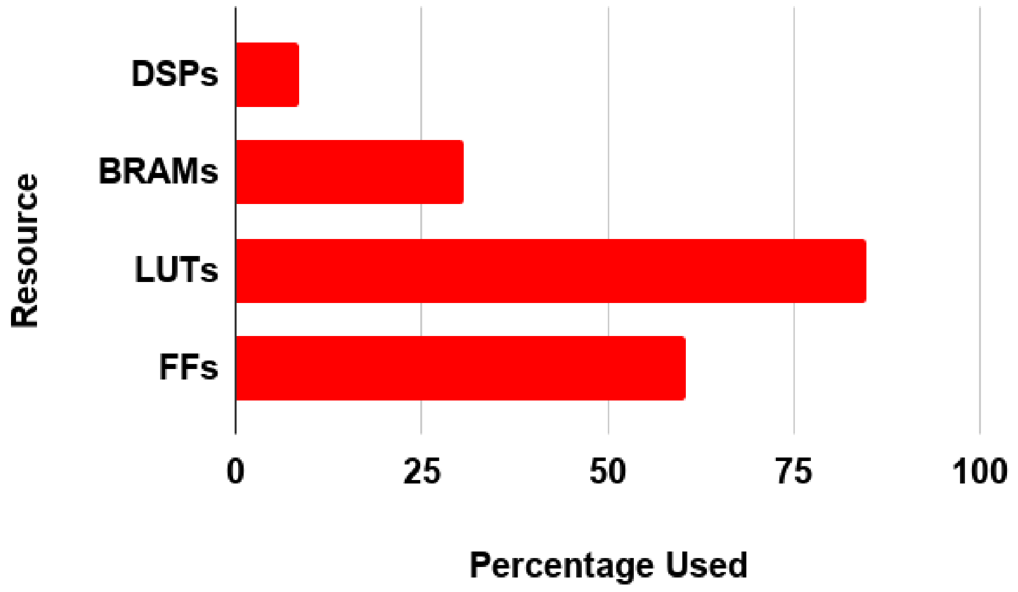


**Figure 5.3:** Preliminary Resource Utilization

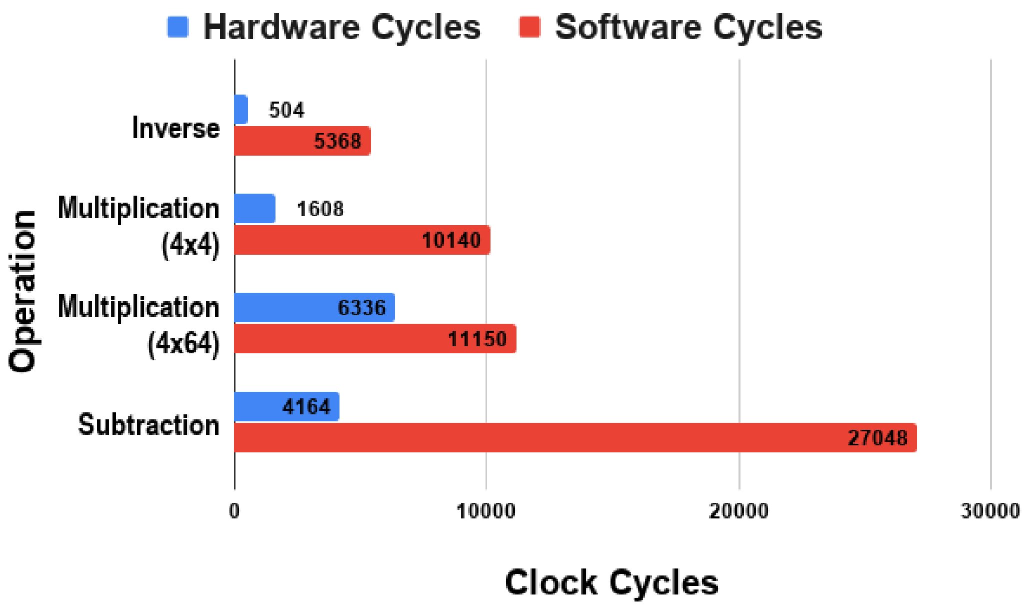


**Figure 5.4:** Preliminary Performance Analysis

performance analysis, the temporal mitigation equation was implemented in MatLab and the corresponding output for 64 samples from FPGA was compared. The Figure 5.4 shows the performance and its observed that the FPGA output closely follows

**Table 5.1:** Preliminary Results

| | Time taken(in clock cycles) | | | Latency |
|---|---|---|---|---|
| Function | Hardware | Software | Speedup | Clock cyles |
| Matrix Inverse | 504 | 5368 | 10.65 | 406 |
| Matrix Multiplication(4*4) | 1608 | 10140 | 6.3059 | 135 |
| Matrix Multiplication(4*64) | 6336 | 11150 | 1.7597 | 617 |
| Matrix Subtraction | 4164 | 27048 | 6.495 | 535 |

the MatLab output. The preliminary FPGA implementation is done using a single floating point data type and there is -80 dB precision(Mean Absolute Error) when comapared with MatLab output. The frequency of the PL is set to 599.99 MHz with a worst case negative slack time of $2\exp^{-9}$seconds.

Figure 5.4 shows the speedup obtained in terms hardware and software cycles. We can observe a speedup of 10X for matrix inverse which is a significant performance improvement. Both multiplication and subtraction have almost a 6X speedup.

Figure 5.3 shows the resource utilization which shows judicious usage of resources. The subsequent implementation attempts to improve this utilization.

## 5.4 Optimized Method of Matrix Inversion

As the preliminary method of matrix inversion in Equation 5.3 involved a very expensive list of operations, for instance the inversion of a complex matrix required four real inversion and four matrix multiplications and two additions. All of these task required a huge amount of resource and expensive in terms of overall computation time as well. Hence alternate approaches for implementation was explored. The following was then selected for implementation based on various factors discussed:

[8] proposes a two NXN(where N is the dimension of the matrix to be inverted) Matrix Inversion solution which is described. Let us consider the following equations :

$$\begin{aligned}
&(A + iC).(x + iy) = (b + id) \\
&A.x - C.y = b \\
&C.x + A.y = d \\
&\begin{pmatrix} A & -C \\ C & A \end{pmatrix} . \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ d \end{pmatrix}
\end{aligned} \tag{5.5}$$

where A and B are both real matrices and A contains the real part and B contains the imaginary part of the matrix to be inverted,b and d are also real matrices representing the real and imaginary part of the resulting inverted matrix. The goal is to find x and y which is done by solving system of equations [9]. Then the following sub matrices are obtained:

$$\begin{pmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{pmatrix} = \begin{pmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{pmatrix}^{-1} \tag{5.6}$$

Then recursive implementation of matrix inversion as demonstrated in [10] and [11] which leads to the following sets of equations,

$$\begin{aligned}
y_{01} &= (C + A.C^{-1}.A)^{-1} \\
y_{10} &= -(C + A.C^{-1}.A)^{-1} \\
y_{00} &= A^{-1} + A^{-1}.C.y_{10} \\
y_{11} &= (C.A^{-1}.C + A)^{-1}
\end{aligned} \tag{5.7}$$

Let $r_0 = A^{-1}.C$ then the following sets of equations are obtained:

$$\begin{aligned} y_{11} &= (C.r_0 + A)^{-1} \\ y_{01} &= r_0.y_{11} \\ y_{10} &= -y_{01} \\ y_{00} &= y_{11} \end{aligned} \tag{5.8}$$

Hence the resulting matrix inverse for $(A + iC)$ is $(y_{00} + iy_{10})$ which can calculated through two real inversions and three real matrix multiplications.

# Chapter 6

# PRECISION RANGE ANALYSIS

With widespread growth of re configurable computing platforms, more developers are getting exposed to hardware development [12]. One of the paradigm shifts is to grasp the notion of bit-level operations. On a typical FPGA fabric logical and arithmetic operators can work at bit level instead of the word level. With careful optimizations of the precision of the datapath, overall size and relative speed [14] of the resulting circuit can be dramatically improved.

Most re configurable logic devices such as FPGAs operate at the bit level. This allows the developer to tune data paths to any word size desired. The following section will discuss this tunability that is available in the light of HLS which is used for implementation. Choosing a wide datapath in FPGAs,usually results in an implementation that is larger than necessary. This consumes valueble resources and potentially reduces the performance of design. On the other hand, if hardware implementation uses too little precision, errors can be introduced at runtime through quantization effects,typically via roundoff or truncation.

## 6.1 Least-Significant Bit Problem

In the process of determining the fixed-point representation of a floating point datapath, the most-significant and least-significant ends must both be considered.

Reducing the relative bit position of the most significant bits reduces the maximum range that the datapath may represent. On the other end, increasing the relative bit position of the least-significant bit(toward the most-significant end) reduces the maximum precision that the datapath can attain.

Having a fixed-point datapath means that results or operations will exhibit some quantity of error compared to their infinite precision counter parts. This quatization error can be introduced on both the most-significant and least-significant sides of the datapath.

If the value of an operation is larger than the maximum value that can be represented by a datapath, the quantization error is a result of truncation or saturation depending on the implementation of the operation. Likewise error gets accumulated at the least significant end of the datapath if the value requires greater precision than the datapath can represent resulting in truncation or round-off error.

## 6.2 Precision Analysis and Error Computation Model

As we can observe from the Figure 6.1 the precision is performed as illustrated. There a two data paths that are propagated through the IPs. The first data path contains the data in the fixed data type (this will be discussed in further detail in the following section) and the second data path contains the data in floating point data type.

After the execution of the IP at every stage the output is compared between the fixed and floating point data type. The error metric use is the mean squared error (MSE) which compares the output of float and fixed output. After every stage we typecast the output of the floating stage and use it as an input for IPs on both the data paths. This allows us to reduce the loss of precision in the fixed data path at the previous stage. Besides serving this purpose it also serves as fair comparison of IP outputs for both data path as the input is essentially the same but with different levels of precision. This serves the purpose of our analysis and also prevents the saturation or overflow of values from cascading through the various stages.

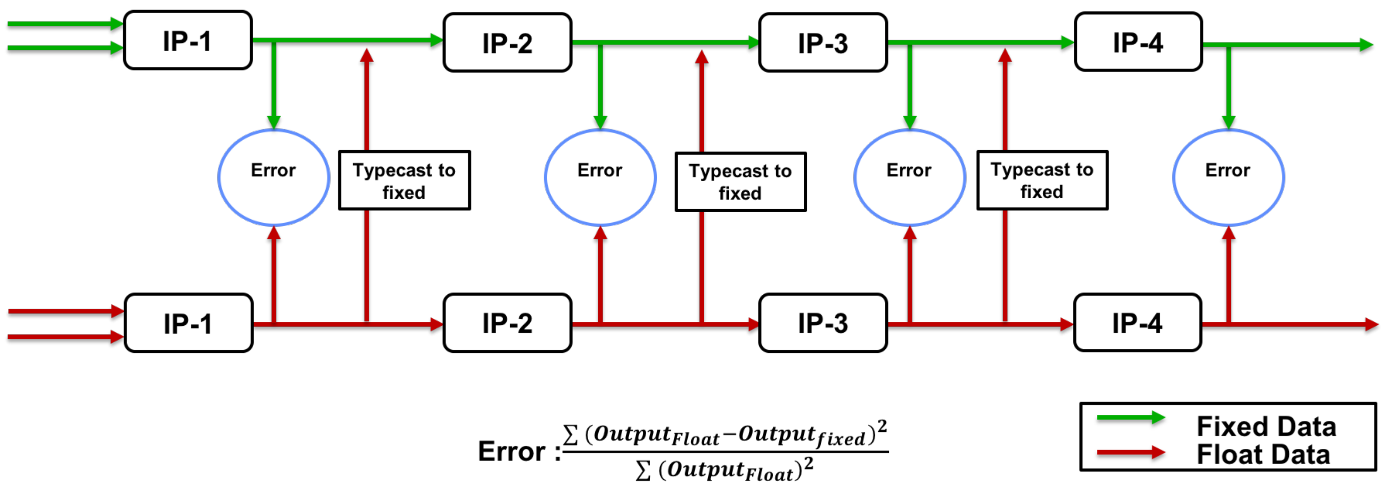


**Figure 6.1:** Error Computation and Implementation of Precision Analysis

The following section discuss the details related to the fixed point data type and how its implemented in hardware by using HLS.

## 6.3 Fixed Data Point Implementation

For our implementation we make use of the Vivado HLS Arbitary Precision Fixed point data type library ”ap_fixed.h”. This allows us to vary the precision bits based on our requirements. Before detailing the identifiers used, the fixed data type needs to be discussed briefly. The Figure 6.2 shows the structure in fixed point. As we can see the Most significant bit (MSB) corresponds to the highest bit of the integer part of the number and the Least Significant bit(LSB) corresponds to the lowest bit of the fractional part.

Further the data type has predefined signature as follows :

$$ap\_[u]fixed < W, I, Q, O, N > \tag{6.1}$$

where, [u] represents the fact that the data type supports both signed and unsigned numbers. W is word length in bits, I is the number of bits used to represent the integer

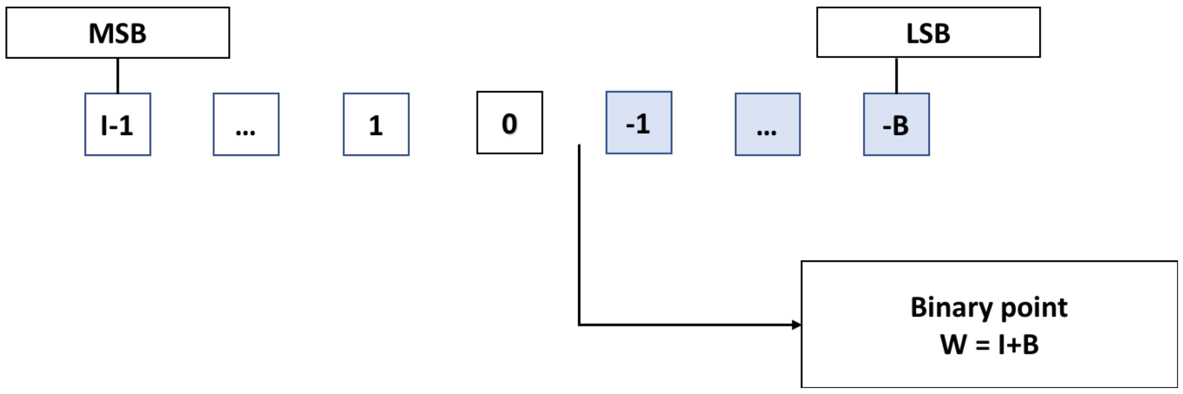


**Figure 6.2:** Fixed Point Data type

part and B in Figure 6.2 is number of bits used to represent the fractional part. Also,

$$W = I + B \tag{6.2}$$

Parameter Q corresponds to the quantization mode being used it dictates the behaviour when greater precision is generated than can be defined by the smallest fractional bit(LSB) in the variable used to store the result. Here the programmer can select from various rounding and truncation schemes.

Parameter O corresponds to the overflow mode where the result of the operation exceeds the maximum possible value that can be stored in the variable used to store the result. The programmer can select various types of saturations and wrapping schemes.

N is the number of saturation bits in the overflow wrap mode.

# Chapter 7

# RESULTS

The chapters 5 and 6 introduced and described the hardware mapping methods and approaches used for analysis. This chapter is contains all of the corresponding results and their interpretations.

## 7.1 Experimental Setup

To make our analysis comprehensive and exhaustive the following sets of experiments were performed :

1. The input to the interference mitigation routine consists of two sets of signals, received signal and training signal(information obtained from transmitter). For the purpose of our experiment we have looked at two kinds of inputs. The signal can be received at Line of Sight(LOS) or it can be received through a multi-path. The channel and the corresponding parameters in simulation are modelled to mimic this effect. We have run our experiments on both LOS and Multi-path inputs. As LOS inputs are much simpler there characteristics are much more relaxed and don't necessarily portray a realistic scenario. Hence the following results are plotted for Multi-path but the LOS experiments were also performed.

2. The interference level between the signal is another factor that is relevance. This gives us a picture of when our algorithm breaks and what is an acceptable range of interference. The term delta(which is used henceforth) is the difference between the interference levels of the received signal(which is also referred

to as Comms Signal) and training signal(referred to as Radar Signal).For our experiments we have swept through a range of 0 to 70 dB interference levels with discrete jumps of 10 dB. So our analysis contains the following interference levels : 0, 10, 20, 30, 40, 50, 60, 70 dB respectively .

3. The precision of the data type discussed in Chapter 6 is implemented as follows: As discussed in 6.3 the data type is adjusted such that a certain number of bits are assigned to integer and fraction part. We run implementation for 8, 16 and 32 bits with the following configurations. For 8 bits, one and three bits are assigned to the integer part : 8 _ 1 and 8 _ 3

   For 16 bits, one, three, five and seven bits were assigned to the integer part : 16_ 1, 16_ 3, 16_ 5 and 16_ 7

   For 32 bits, one, three, five, seven, nine, eleven and thirteen bits were assigned to the integer part : 32_ 1, 32_ 3, 32_ 5, 32_ 7, 32_ 9, 32_ 11, 32_ 13.

4. The MSE error 6.1 being propagated is also recorded for each of the interference level and data type combination. The most relevant plots for the above are shown in the following section

The acceleration achieved in the preliminary implementation was discussed in Section 5.3. Upon further optimization the inversion was improved as discussed in Section 5.4. This was algorithmic as well as hardware optimization. Complex Inversion was simplified by orders of operation and reduced to two inversion and three multiplications respectively.

## 7.2 Results Obtained

### *7.2.1 Computation Time as a Function of Data Type*

The Figure 7.1 shows the relationship between computation time of each IP as function of data type. The IPs are represented by their respective colors in the legend and the horizontal axis represents the data type and the vertical axis represents the time taken for each of the IPs in micro seconds on a log scale.

The plot shows as increasing trend as we move to higher precision data type as there more CLBs involved in each computation and the precision of the data type plays a pivotal role in this trend. The two matrix multiplication represented in red and light blue color both show prominent peaks across all the data types. This is an expected trend as the both these multiplications contain approximately ($4 \times 64 * 64 \times 4$ and $4 \times 4 * 4 \times 64$ elements respectively) which is an expensive calculation. From the plot it evident that a lower precision data type has lower computation time and compute time increases as we progress towards higher precision data type.

### *7.2.2 Output Error as Function of Data Type with Varying Interference Levels*

The most important performance metric for our analysis is the final output error. For this purpose we treat our golden output as the output from MATLAB. The error is calculated as the log of the absolute difference between the values given by :

$$NormalizedError = 10 \log_{10} |\frac{X - Y}{X}| \tag{7.1}$$

where X is the MATLAB value which is assumed to have infinite precision and Y is the output of the FPGA. This analysis is performed for varying levels of interference and corresponding variation in data type.

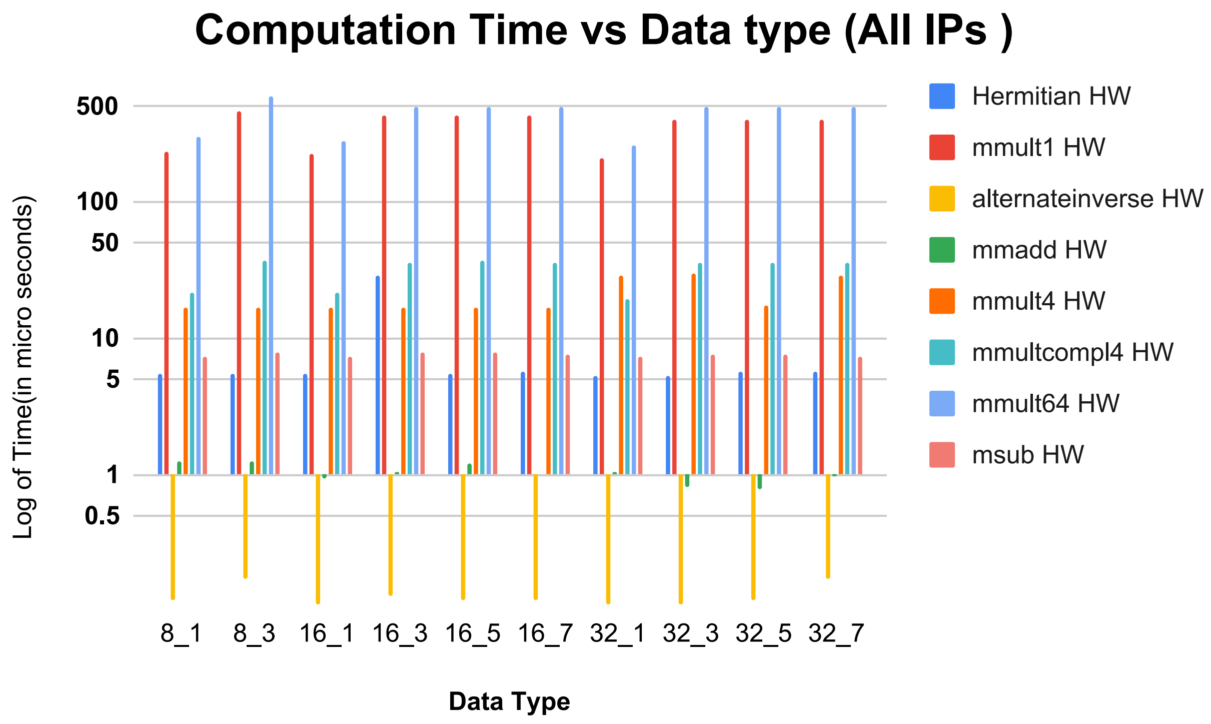


**Figure 7.1:** Computation Time as a Function of Data Type for Various IPs Involved

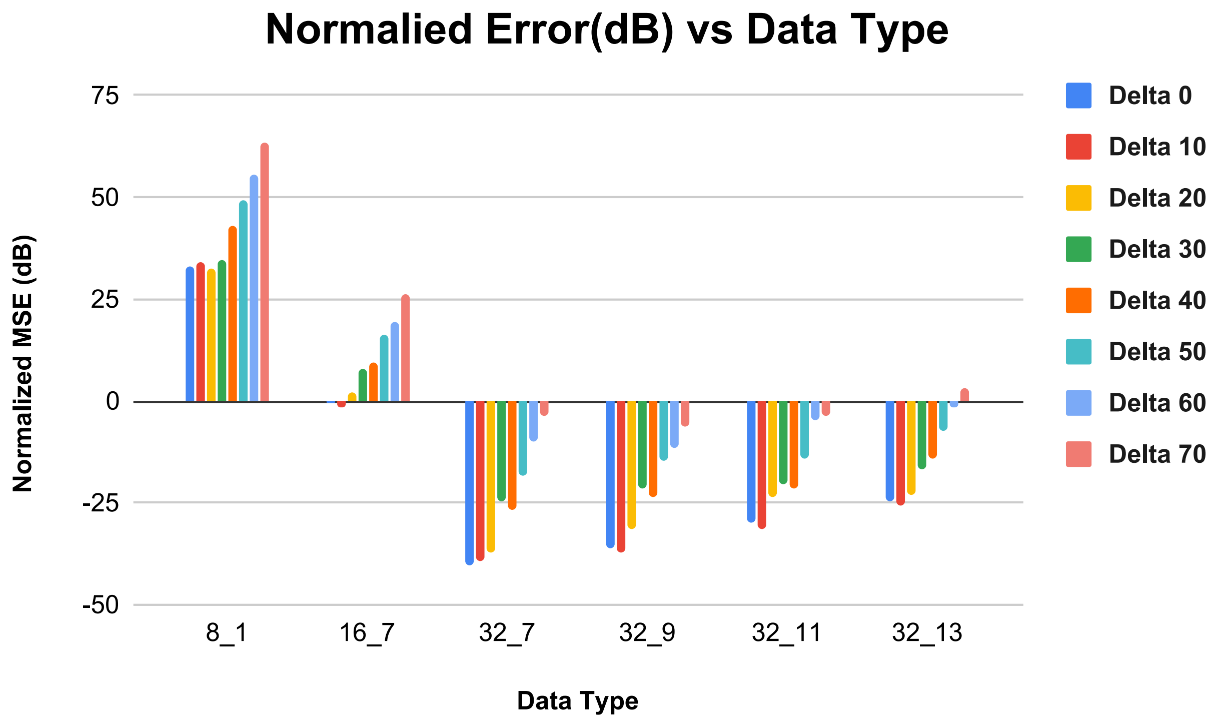


**Figure 7.2:** Final Output Error as Function of Data Type for Various Values of Delta

Upon plotting the Normalized Error it was discovered that the 8 bit data type is just not enough for our computation purposes and this is represented by the first set of column in the plot and we can see an error of above 25 dB for even a 0 dB difference in the interference levels. Further it was discovered that we require atleast 7 bits of integer for the overall computation to combat with conditions of overflow which can occur if the integer bits are less than 7. Hence even with a 16 bit data type with 7 bits for the integer part we can see the error is a very small negative value and it spikes up rapidly as we increase the interference levels. The reason for this trend is attributed to the lack of fraction bits which leads to quantization and rounding errors in intermediate computations. If we observe the trend for 32 bits we can see an increase in the error as we increase the integer bits(i.e., as we move from 32_ to 32_3). This is explained by the same reason as discussed above, the decrease in the fraction bits degrades the performance. This decrease is not as drastic because of sufficient integer bits which avoids any overflow and rounding errors which are more prominent in the lower bit data types.

### *7.2.3 Resource Utilization*

The Figure 7.3 shows the utilization of resources as function of data type. The trend is as predicted, the resources used increases as we increase the precision and more CLBs and BRAMSs are used. Also the DSP utilization shows an increasing trend as we move toward a higher precision data type.

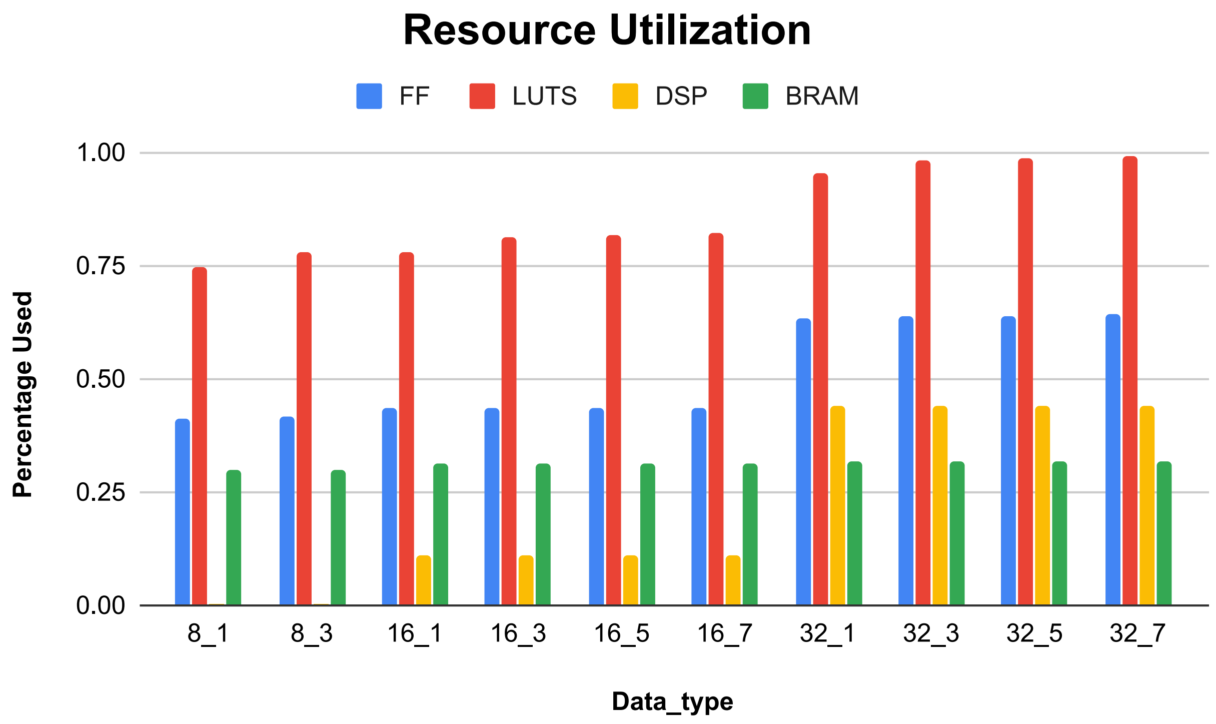


**Figure 7.3:** Resource Utilization as Function of Data Type

## 7.3 MSE Loss Propagation

We also perform analysis of the MSE propogation across the various stages of the temporal mitigation chain and each IPs MSE loss after every subsequent stage is recorded.

The Figures 7.4 and 7.5 shows this MSE propogation trend for 10dB and 70dB interference level differences for the hermitian operator.

The Figures 7.6 and 7.7 shows this MSE propogation trend for 10dB and 70dB interference level differences for the Matrix Multiplication(4X64 * 64X4) operator.

The Figures 7.8 and 7.9 shows this MSE propogation trend for 10dB and 70dB interference level differences for the inverse operator.

The Figures 7.10 and 7.11 shows this MSE propogation trend for 10dB and 70dB interference level differences for the Matrix Multiplication(4X4 * 4X4)operator.

The Figures 7.12 and 7.13 shows this MSE propogation trend for 10dB and 70dB interference level differences for the Matrix Multiplication(4X4 * 4X64) operator.

The Figures 7.14 and 7.15 shows this MSE propogation trend for 10dB and 70dB interference level differences for the Matrix Subtraction operator.

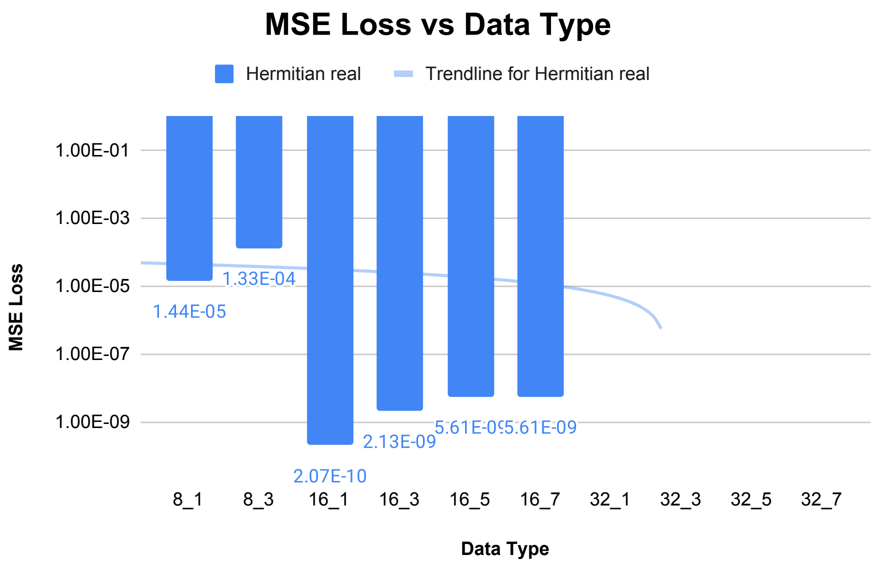


**Figure 7.4:** MSE Loss as Function of Datatype for 10dB Delta in Hermitian Accelerator

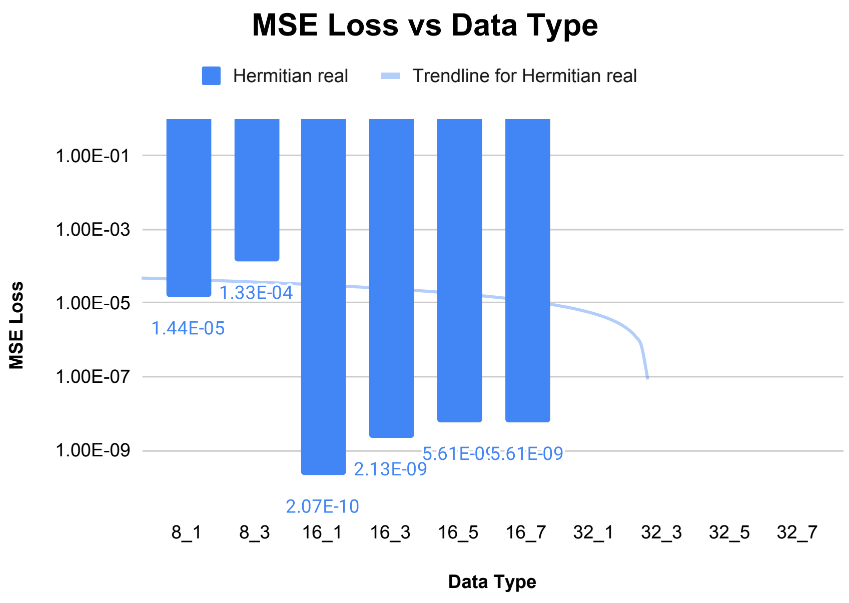


**Figure 7.5:** MSE Loss as Function of Datatype for 70dB Delta in Hermitian Accelerator

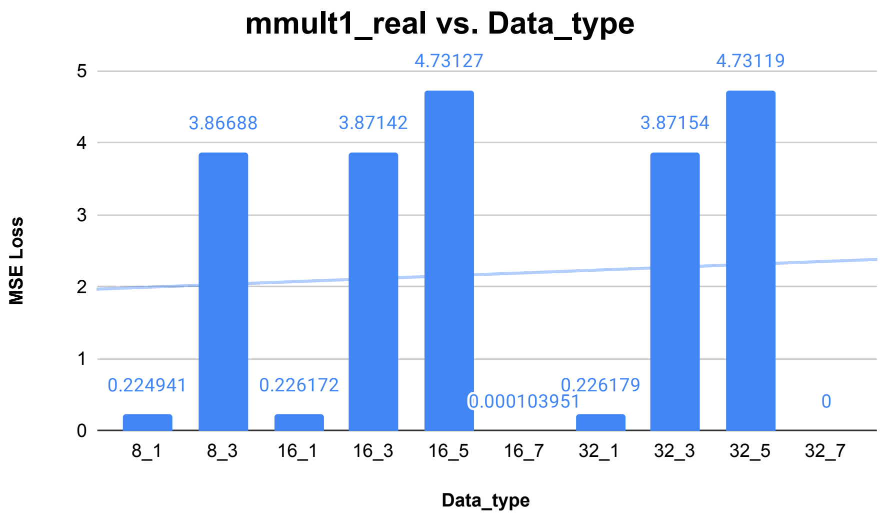


**Figure 7.6:** MSE Loss as Function of Datatype for 10dB Delta in Matrix Multiply Accelerator

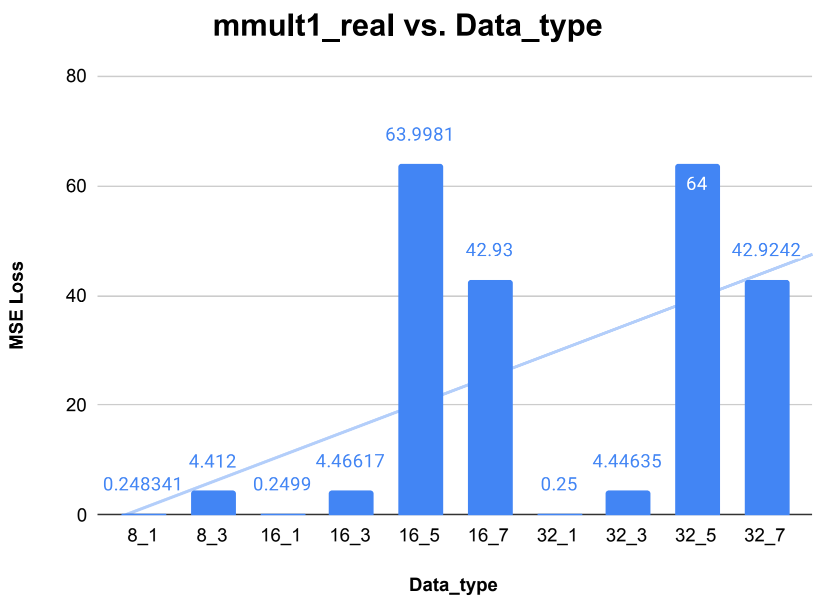


**Figure 7.7:** MSE Loss as Function of Datatype for 70dB Delta in Matrix Multiply Accelerator

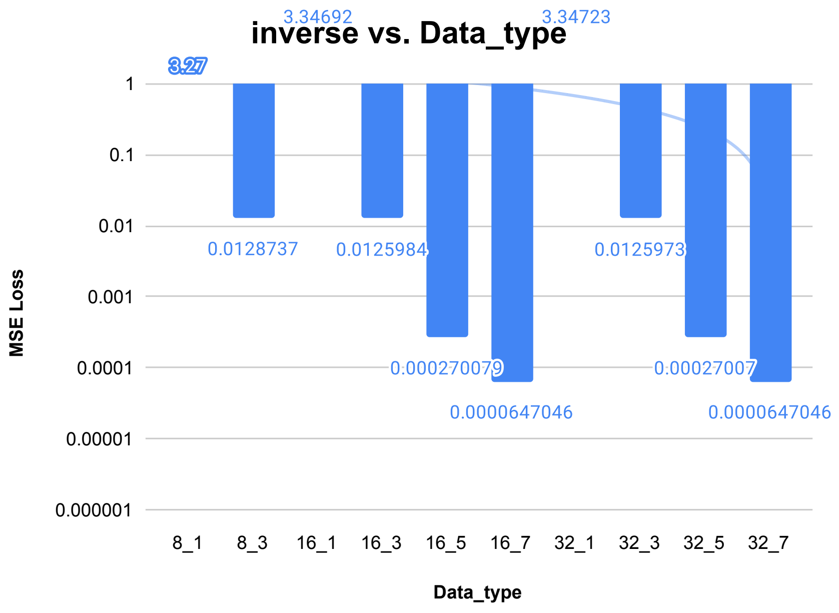


**Figure 7.8:** MSE Loss as Function of Datatype for 10dB Delta in Inverse Accelerator

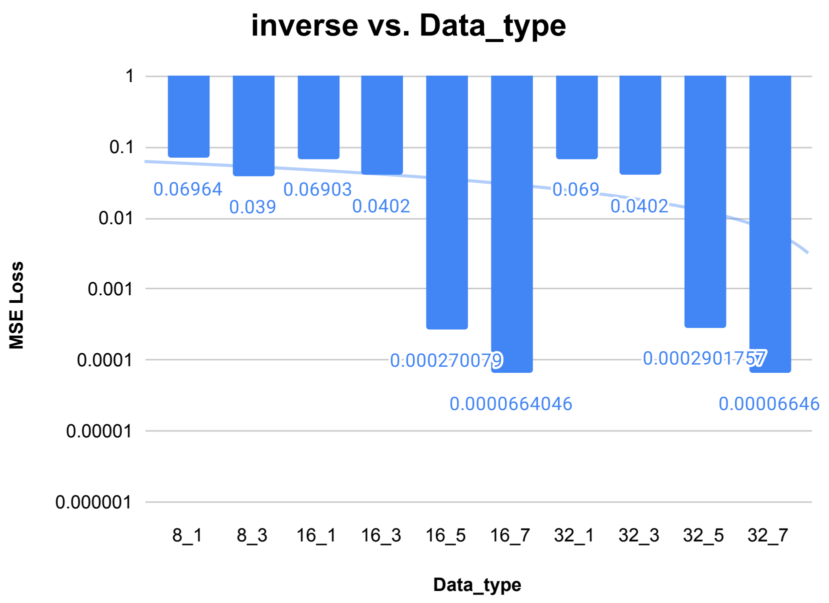


**Figure 7.9:** MSE Loss as Function of Datatype for 70dB Delta in Inverse Accelerator

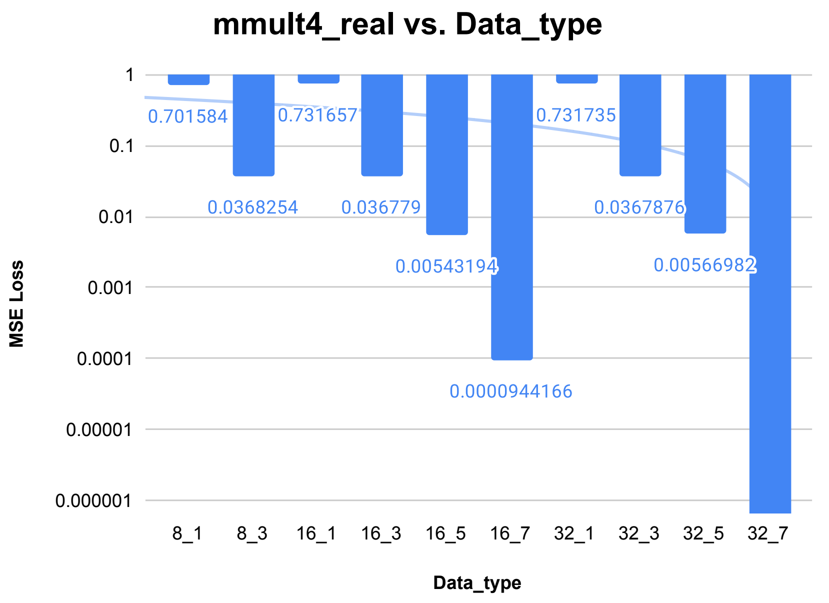


**Figure 7.10:** MSE Loss as Function of Datatype for 10dB Delta in Matrix Multiply (4X4) Accelerator

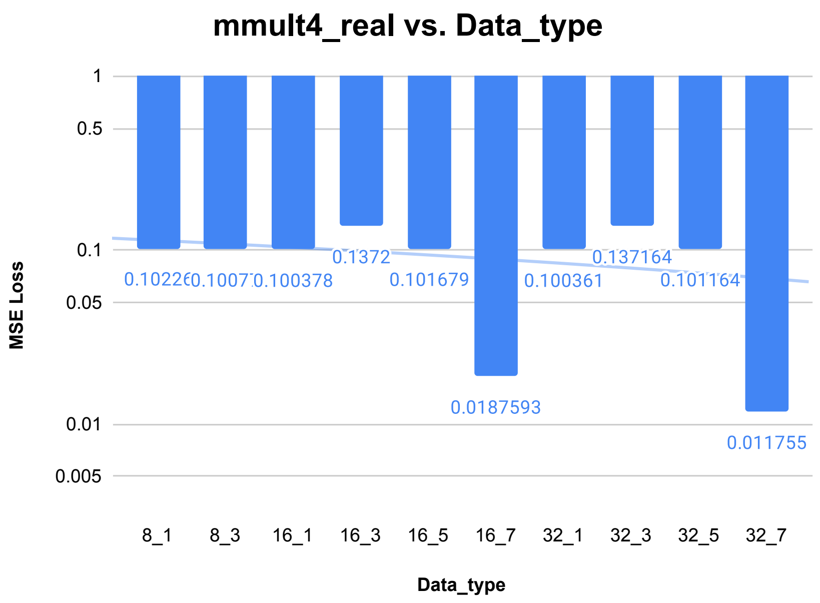


**Figure 7.11:** MSE Loss as Function of Datatype for 70dB Delta in Matrix Multiply(4X4) Accelerator

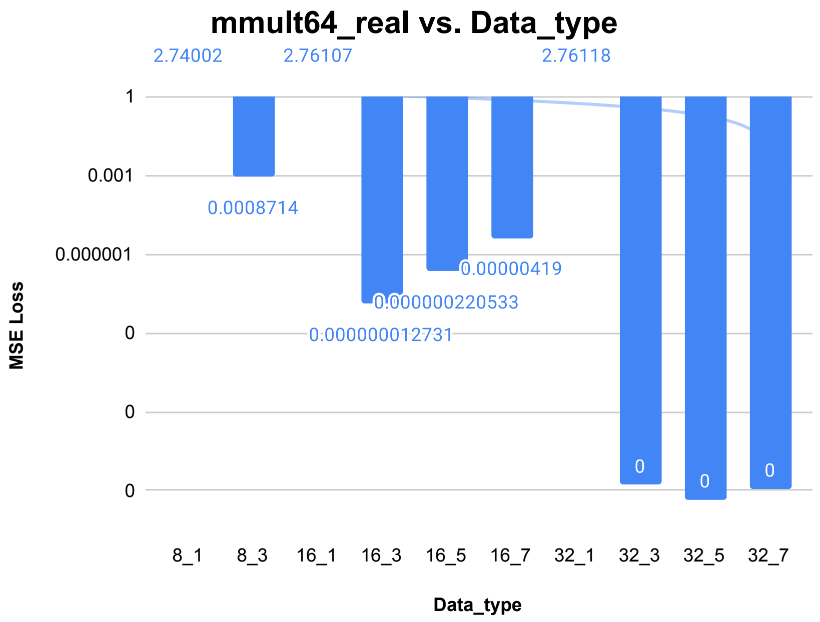


**Figure 7.12:** MSE Loss as Function of Datatype for 10dB Delta in Matrix Multiply(4X64) Accelerator

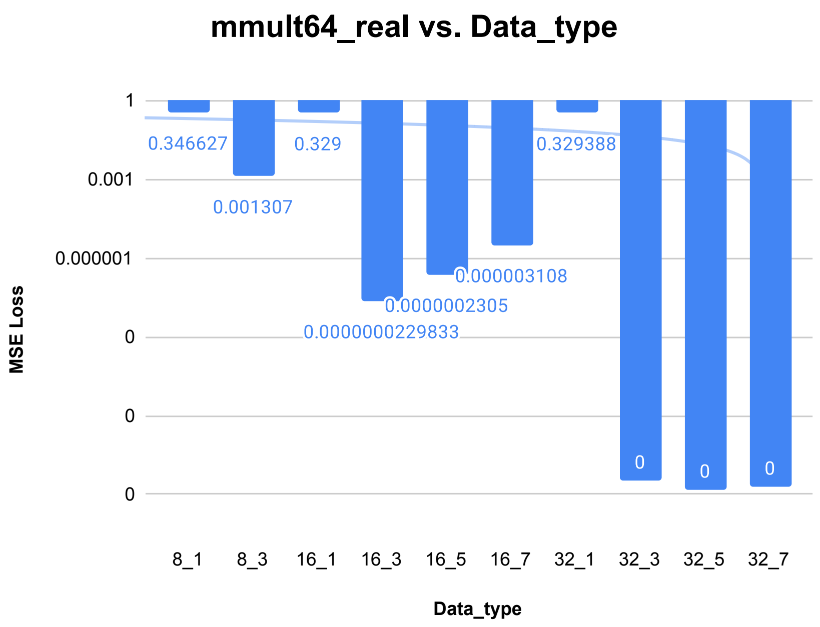


**Figure 7.13:** MSE Loss as Function of Datatype for 70dB Delta in Matrix Multiply(4X64) Accelerator

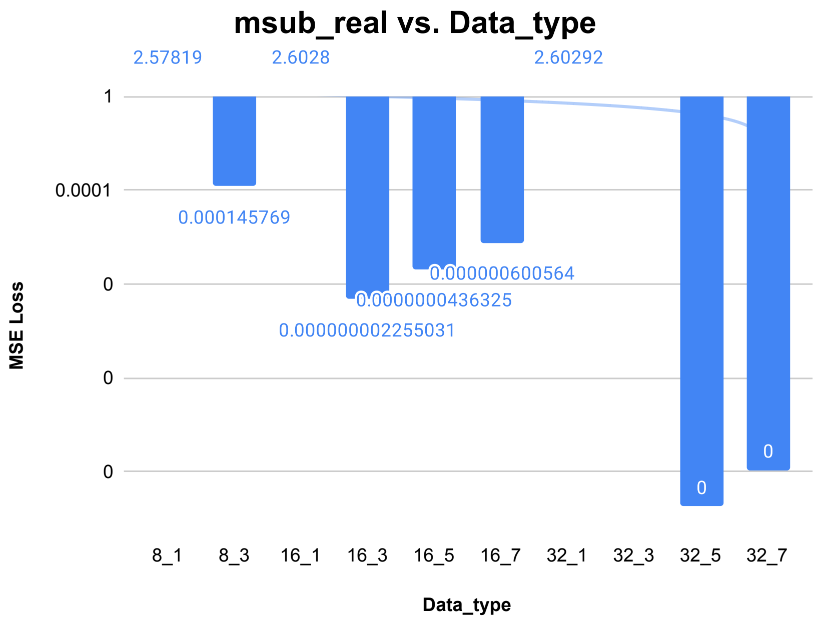


**Figure 7.14:** MSE Loss as Function of Datatype for 10dB Delta in Matrix Subtraction Accelerator

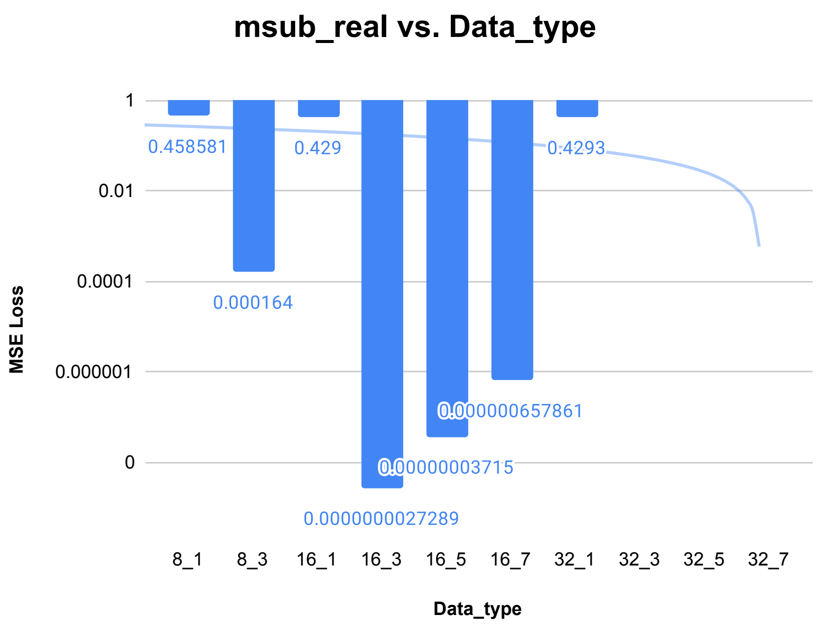


**Figure 7.15:** MSE Loss as Function of Datatype for 70dB Delta in Matrix Subtraction Accelerator

The trend is consistent in both 10 dB and 70 dB interference levels which shows us that IP performs in consistent manner and error is actually attributed to the increase in interference levels as go from 10 dB to 70 dB. The trends also follow the observations discussed in 7.2.2 as see the MSE spikes up as we increase or decrease the integer bits which consequently decreases or increases the fraction bits respectively.

## 7.4 Overall Result

Based on the MSE errors calculated in the above section, the following hybrid combination of data type IPs is proposed. The Figure 7.16 shows the most optimal data type for each of individual IPs selected based on their MSE error performance. This model is an attempt to have a structure where we can judiciously use hardware resources while remaining within acceptable limits of overall performance and accuracy.

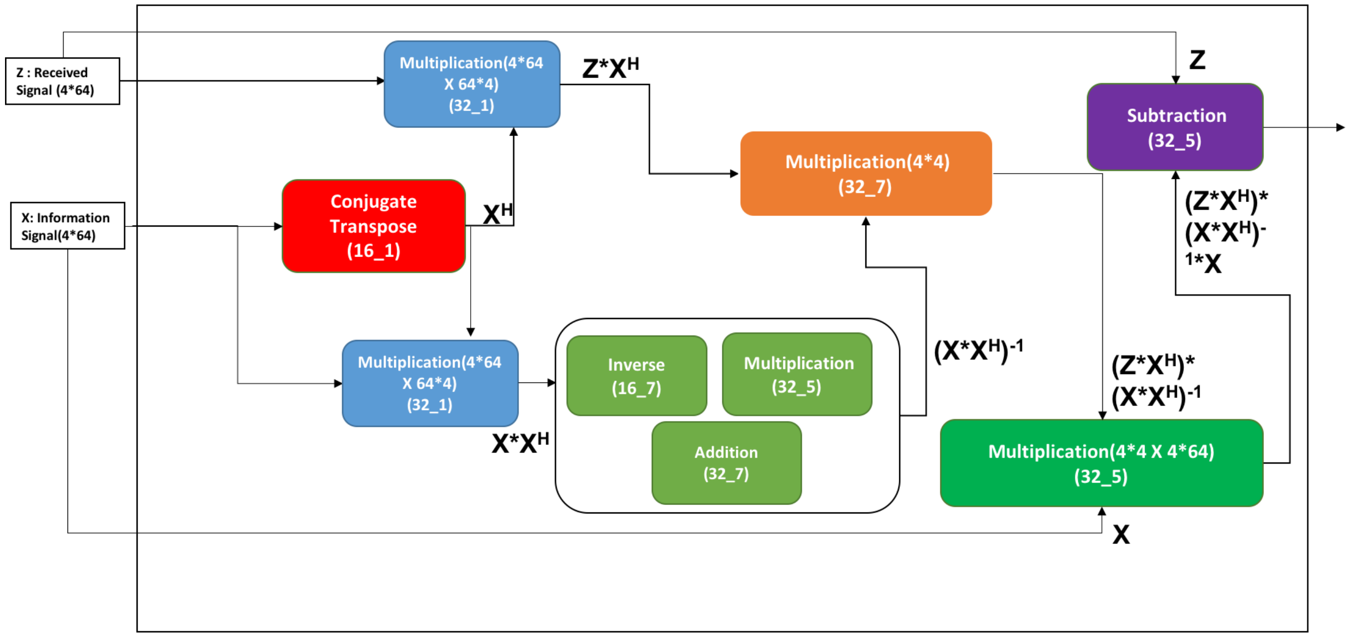

Z : Received Signal (4*64)
X: Information Signal(4*64)
Multiplication(4*64 X 64*4) (32_1)
Z*X^H
Conjugate Transpose (16_1)
X^H
Multiplication(4*4) (32_7)
Subtraction (32_5)
Z
(Z*X^H)* (X*X^H)^-1*X
Multiplication(4*64 X 64*4) (32_1)
X*X^H
Inverse (16_7)
Multiplication (32_5)
Addition (32_7)
(X*X^H)^-1
(Z*X^H)* (X*X^H)^-1
Multiplication(4*4 X 4*64) (32_5)
X

**Figure 7.16:** Final Hybrid Data Type Flow Graph

The most optimal data type for the normalized final output error is 32 bits word length with seven bits for integer part, as we can observe from Fig. 7.2 that Normalized MSE is below zero for a wide range of interference levels.

In terms of resource utilization eight bits has significantly low resource utilization but corresponding accuracy is very low. Although 32 bits has much higher resource utilization the corresponding accuracy is very high as well. So the accuracy and resource usage have trade off where high accuracy comes at the cost of expensive(or high) resource usage.

The computation time has dependence on number of integer bits being used irrespective of the word length. As we can observe from Fig. 7.1 that as we increase the number of integer bits involved the computation time increases. For example, for both word lengths of 16 or 32 bits as the number of integer bits increase the computation time for both the multiplication IPs in red and light blue increases. Although some operations such as conjugate transpose(or hermitian) and subtraction are not affected by the change in data type. This trend is explained by the nature of the operations themselves.

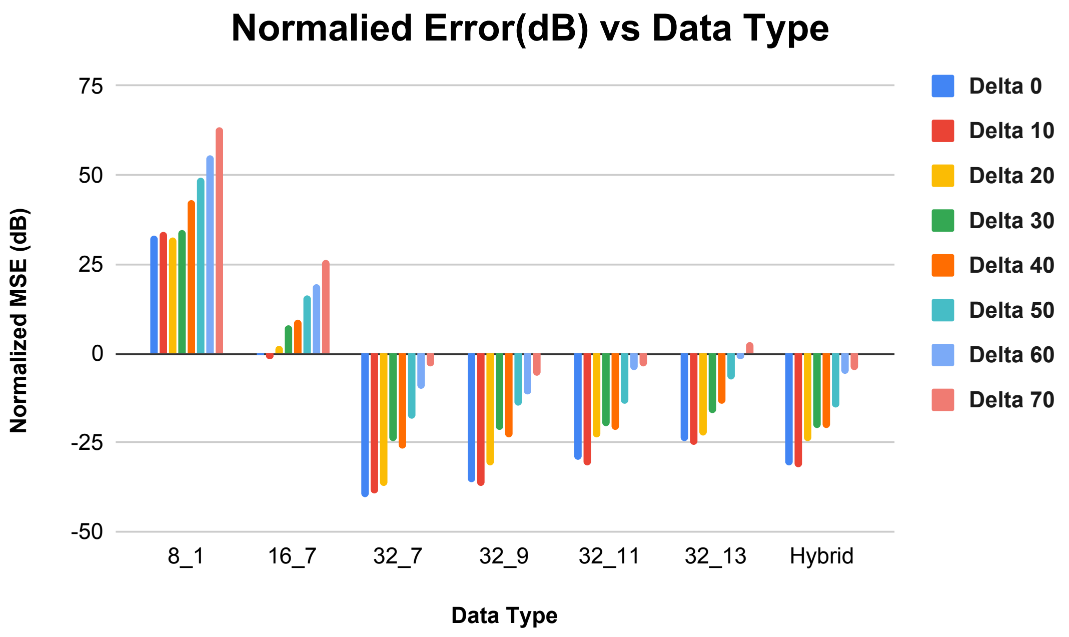


**Figure 7.17:** Final Output Error as Function of Data Type and Hybrid Data Type for Various Values of Delta

From Fig.7.17 we can observe the overall error for hybrid data type (which is based on Fig.7.16). The overall error performance is appreciable and comparable to the best output error. Another conclusion is that the overall output error for the hybrid data type is comparable to the best result although some of the operations are now performed with a lower precision data type. For example, the transpose and inverse are performed using only 16 bits of word length. Also, the remaining IPs use much lower integer bits for computation.

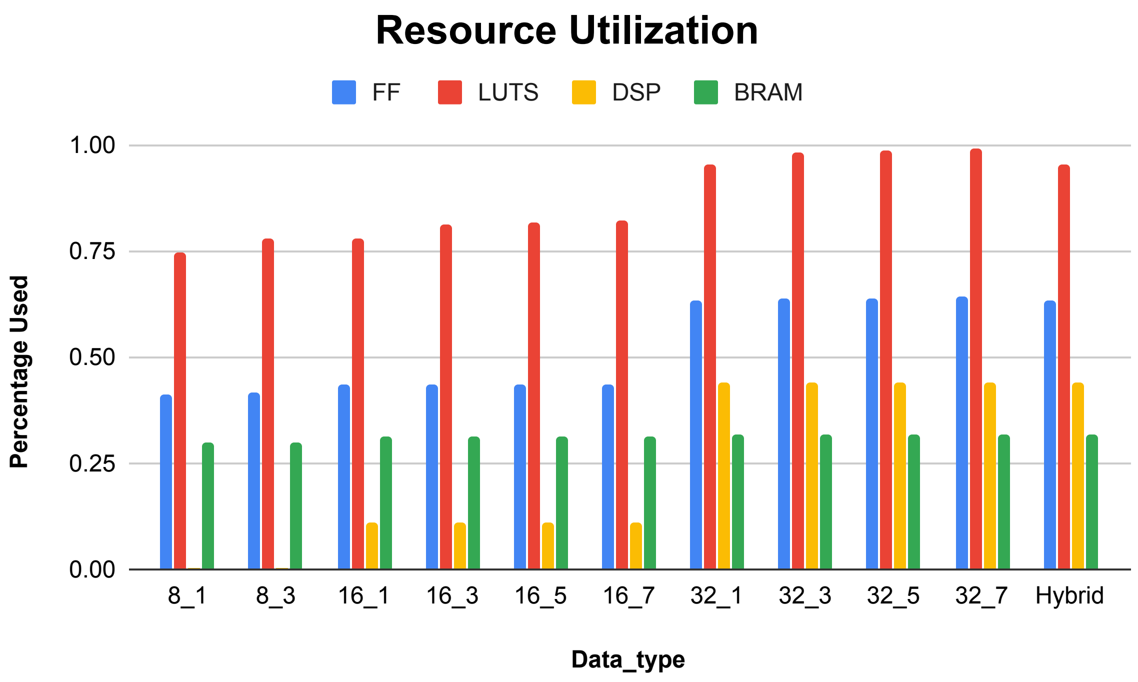


**Figure 7.18:** Resource Utilization as Function of Data Type and Hybrid Data Type

Fig.7.18 shows the resource utilization for hybrid data type compared to other data type combinations. The resource percentage usage is comparable to most 32 bits and usage of CLBs(FFs and LUTs) is better than best performance data type. The data type 32_7 is best in terms of overall results but its also expensive in terms of the resource usage. When comparing this with our hybrid data type we can observe a reduction is usage of CLBs and the overall percentage for both is comparable to lower precision data type like 32_1. This is significant gain but the reason there not a drastic improvement with hybrid data type is because even with hybrid data type we still a majority of IPs with 32 bit word length. Further the usage of resources by each IP is not evenly distributed and hence the IPs which have higher word length are also expensive calculations(i.e, multiplications). They steer the resource utilization trend and hence the corresponding results are observed. Another observation is the overall error performance improves or comes at the cost of corresponding resource usage.

Chapter 8

# CONCLUSION AND FUTURE WORK

The previous chapters shows us the results for the various types of analyses. The precision analysis give us a deep insight into the performance of the IP and the various parameters that play a crucial role in the characteristics discussed. As we can observe the effect of precision of data type is extremely important parameter and it can control various aspects of performance. The contribution of the above analyses is that it performs hardware implementation of temporal interference mitigation. It looks at the internal processes involved whereby it also performs algorithmic improvements which benefit the overall computation. It also performs comprehensive precision analysis that accounts for various factor in action and proposes a framework for the same. The framework is beneficial as it isolates and examines many factors that contribute to the performance in terms of accuracy, time and resource usage.

The conclusion is that hardware mapping of temporal interference mitigation was performed along with a performance evaluation analysis that provided valuable insights. The future work will attempt to focus on using this analysis as base when developing any hardware acceleration implementation as this leads to very exhaustive search on the various factors involved and gives the designer and better picture when making implementation and design decision based on given requirements and various acceptance thresholds.

# REFERENCES

[1] D. W. Bliss, "Full-duplex self-interference mitigation analysis for direct conversion RF nonlinear MIMO channel models with IQ mismatch," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 6543-6547.

[2] B. Paul, A. R. Chiriyath and D. W. Bliss, "Joint communications and radar performance bounds under continuous waveform optimization: The waveform awakens," 2016 IEEE Radar Conference (RadarConf), Philadelphia, PA, 2016, pp. 1-6.

[3] B. Paul, A. R. Chiriyath and D. W. Bliss, "Survey of RF Communications and Sensing Convergence Research," in IEEE Access, vol. 5, pp. 252-270, 2017.

[4] Bliss, D., Govindasamy, S. (2013). Adaptive Wireless Communications: MIMO Channels and Networks. Cambridge: Cambridge University Press.

[5] N. I. Miridakis and D. D. Vergados, "A Survey on the Successive Interference Cancellation Performance for Single-Antenna and Multiple-Antenna OFDM Systems," in IEEE Communications Surveys  Tutorials, vol. 15, no. 1, pp. 312-335, First Quarter 2013.

[6] G. S. Deepthy and R. J. Susan, "Analysis of Successive Interference Cancellation in CDMA Systems," 2012 Second International Conference on Advanced Computing Communication Technologies, Rohtak, Haryana, 2012, pp. 481-485.

[7] A. Al Housseini and S. Saoudi, "Nonlinear Successive Interference Cancellation for Turbo-CDMA," 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications, Damascus, 2008, pp. 1-3.

[8] Andreas Falkenberg,"Using a Real Matrix Inversion Algorithm to Implement the Inversion for Complex Matrices,"The 7th International Conference on Optimization: Techniques and Applications (ICOTA7),2007.

[9] Visual DSP++ 4.0 C/C++ Compiler and Library Manual for TigerSHARC Processors; Analog Devices; 2005.

[10] W. Press, S.A. Teukolsky, W.T. Vetterling, B.R. Flannery; Numerical Recipes in C++, The art of scientific computing, Second Edition; p52 : "Complex Systems of Equations";Cambridge University Press 2002.

[11] W. Press, S.A. Teukolsky, W.T. Vetterling, B.R. Flannery; Numerical Recipes in C++, The art of scientific computing, Second Edition; p106 : "Is Matrix Inversion an N3 process";Cambridge University Press 2002.

[12] Chang, Mark  Hauck, Scott. (2003). Variable Precision Analysis for FPGA Synthesis.

[13] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk and P. Y. K. Cheung, ”Reconfigurable computing: architectures and design methods,” in IEE Proceedings - Computers and Digital Techniques, vol. 152, no. 2, pp. 193-207, 4 March 2005.

[14] A. Nayak, M. Haldar, A. Choudhary and P. Banerjee, ”Precision and error analysis of MATLAB applications during automated hardware synthesis for FPGAs,” Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001, Munich, Germany, 2001.

[15] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers and Z. Zhang, ”High-Level Synthesis for FPGAs: From Prototyping to Deployment,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 4, pp. 473-491, April 2011.

[16] `https://rb.gy/vuhexn`

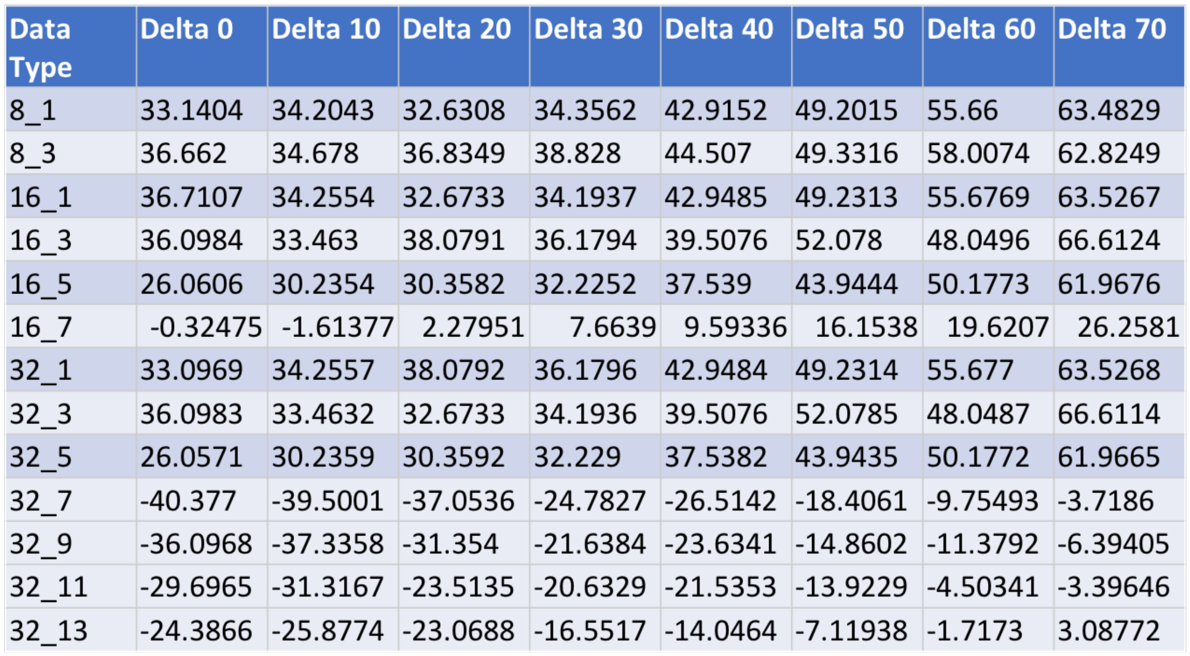
[17] `https://rb.gy/9s4uge`

[18] `https://rb.gy/bxwidu`

[19] `https://rb.gy/l4qenj`

# APPENDIX A

# RAW DATA



| Data Type | Delta 0 | Delta 10 | Delta 20 | Delta 30 | Delta 40 | Delta 50 | Delta 60 | Delta 70 |
|---|---|---|---|---|---|---|---|---|
| 8_1 | 33.1404 | 34.2043 | 32.6308 | 34.3562 | 42.9152 | 49.2015 | 55.66 | 63.4829 |
| 8_3 | 36.662 | 34.678 | 36.8349 | 38.828 | 44.507 | 49.3316 | 58.0074 | 62.8249 |
| 16_1 | 36.7107 | 34.2554 | 32.6733 | 34.1937 | 42.9485 | 49.2313 | 55.6769 | 63.5267 |
| 16_3 | 36.0984 | 33.463 | 38.0791 | 36.1794 | 39.5076 | 52.078 | 48.0496 | 66.6124 |
| 16_5 | 26.0606 | 30.2354 | 30.3582 | 32.2252 | 37.539 | 43.9444 | 50.1773 | 61.9676 |
| 16_7 | -0.32475 | -1.61377 | 2.27951 | 7.6639 | 9.59336 | 16.1538 | 19.6207 | 26.2581 |
| 32_1 | 33.0969 | 34.2557 | 38.0792 | 36.1796 | 42.9484 | 49.2314 | 55.677 | 63.5268 |
| 32_3 | 36.0983 | 33.4632 | 32.6733 | 34.1936 | 39.5076 | 52.0785 | 48.0487 | 66.6114 |
| 32_5 | 26.0571 | 30.2359 | 30.3592 | 32.229 | 37.5382 | 43.9435 | 50.1772 | 61.9665 |
| 32_7 | -40.377 | -39.5001 | -37.0536 | -24.7827 | -26.5142 | -18.4061 | -9.75493 | -3.7186 |
| 32_9 | -36.0968 | -37.3358 | -31.354 | -21.6384 | -23.6341 | -14.8602 | -11.3792 | -6.39405 |
| 32_11 | -29.6965 | -31.3167 | -23.5135 | -20.6329 | -21.5353 | -13.9229 | -4.50341 | -3.39646 |
| 32_13 | -24.3866 | -25.8774 | -23.0688 | -16.5517 | -14.0464 | -7.11938 | -1.7173 | 3.08772 |

**Figure A.1:** Normalized Error as Function of Data type