5-2024

Digital Beamforming Implemented in Hardware

Nicole Dulieu

DIGITAL BEAMFORMING IMPLEMENTED IN HARDWARE

by

NICOLE DULIEU

GRADUATE PAPER

Submitted in partial fulfillment
of the requirements for the degree of
MASTER OF SCIENCE
in Electrical Engineering

Approved by:

Mr. Mark A. Indovina, Senior Lecturer
Graduate Research Advisor, Department of Electrical and Microelectronic Engineering

---

Dr. Ferat Sahin, Professor
Department Head, Department of Electrical and Microelectronic Engineering

DEPARTMENT OF ELECTRICAL AND MICROELECTRONIC ENGINEERING
KATE GLEASON COLLEGE OF ENGINEERING
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

MAY, 2024

Dedication

I dedicate this research paper to my mom Stephanie Dulieu and my dad Paul Dulieu. Thank you for always believing in me and pushing me to become a better engineer and person.

Nicole Dulieu

Declaration

I declare that except where specific reference is made to the work done by others, that the entirety of this graduate paper is my work. The submission of the graduate paper is not considered for any other degree in this University or any other University. The Graduate Project is the work of my own and does not include any work done in collaboration, except where work is referenced to others.

Nicole Dulieu

May, 2024

Acknowledgements

I would like to thank my parents Stephanie Dulieu and Paul Dulieu for their unwavering support throughout my life. Thank you for giving me the privilege of a undergraduate and graduate education, I am so grateful for you both. To my advisor Mark Indovina for sharing the knowledge and experience that provided me an exceptional education of digital design. Your continuous support of my education from my freshman to senior year has had such a positive effect on my academic career. Thank you for advising my graduate paper and allowing me to explore a topic outside of your expertise and still provided me with knowledgeable insight.

Nicole Dulieu

Abstract

Digital beamforming is a popular method used in modern communication systems. The ability to track and locate a transmitting signal adaptively is necessary in communication systems. Beamforming is one solution to this problem. Beamforming uses an array or matrix of isotropic antenna elements. This eliminates the need to create a physically larger antennas to achieve the same radiation pattern and gain of a phased array of antenna elements. Additionally, the antennas are electronically controlled allowing the radiation pattern and gain to adapt quickly. It is necessary to use a digital platform for beamforming because hardware can digitize analog signals efficiently. The research done in this paper starts with a system created in Matlab's Simulink environment. This system has both software and hardware beamforming algorithms. The results from the two algorithms is verified in waveforms. The hardware beamforming system is converted to hardware description language (HDL) using Simulink's HDL Coder application. The HDL files are used in a simulation environment using Cadence Incisive simulator to verify the beamforminf results. The digital beamforming HDL project is implemented on different application specific integrated circuit (ASIC) technologies. Using Synopsys compiler suites the project is synthesized, placed and routed on the ASIC technologies. This paper will analyze the results from implementing a beamforming algorithm on digital hardware.

Contents

Contents v

List of Figures vii

List of Tables ix

1 Introduction 1

1.1 Research Goals 2

1.2 Contributions 2

1.3 Organization 3

2 Bibliographical Research 5

2.1 A brief overview of beamforming and phased arrays 5

2.1.1 Beamforming Algorithms 6

2.1.1.1 Direction of Arrival and power method 7

2.1.1.2 Null-steering techniques 7

2.1.1.3 Adaptive Beamforming 7

2.1.1.4 Phase-shift beamforming 8

2.2 Digital Hardware Implementation 9

2.2.1 Types of hardware platforms 10

2.2.1.1 Field Programmable Gate Arrays 10
2.2.1.2 Digital Signal Processor 10
2.2.1.3 Application Specific Integrated Circuits 11

3 MATLAB and Simulink Environment 12
3.1 Behavioral Algorithm 12
3.2 Hardware Description Language Algorithm 13
3.2.1 Digital Conversion of the signal information 13
3.2.2 Generating the Steering Vector 14
3.3 Generating HDL code 16
3.3.1 Target Platform 16
3.3.2 HDL and Testbench Generation 17

4 Simulation Results 18
4.1 Simulation Results 18
4.2 Co-simulation Results 22
4.3 Testbench files 25

5 Hardware Results 27
5.1 Synthesis 28
5.2 Pre and Post Scan Insertion Results 29
5.3 Integrated Custom Compiler Results 36

6 Conclusion 41
6.1 Future Work 42

References 43

List of Figures

2.1 Delay and Sum Beamformer 9

3.1 Simulink Beamforming System 13
3.2 HDL Logic to Find Steering Vector 15
3.3 System with the Co-simulation Platform 16

4.1 Behavioral Beamformed Signal Result 19
4.2 Output Signals from the HDL and Behavioral Models 20
4.3 Error between Behavioral and HDL Beamformed Signals 21
4.4 Co-simulation Environment 23
4.5 Co-simulation Error Console 24
4.6 Waveform of Generating Steering Vectors 25
4.7 Waveform of the Beamformed Signal 26

5.1 Pre-Scan Insertion Technology vs Area Graph 29
5.2 Pre-Scan Insertion Technology vs Internal Power Graph 30
5.3 Pre-Scan Insertion Technology vs Switching Power Graph 30
5.4 Pre-Scan Insertion Technology vs Leakage Power Graph 31
5.5 Pre-Scan Insertion Technology vs Total Power Graph 31

5.6 Pre-Scan Technology vs Slack 32
5.7 Post-Scan Insertion Technology vs Area Graph 34
5.8 Post-Scan Insertion Technology vs Internal Power 34
5.9 Post-Scan Insertion Technology vs Switching Power 35
5.10 Post-Scan Insertion Technology vs Leakage Power 35
5.11 Post-Scan Insertion Technology vs Total Power 36
5.12 Post-Scan Insertion Technology vs Slack 37
5.13 Place and Route of the Digital Beamforming Design 38
5.14 Detailed View of the Place and Route Result 39

List of Tables

5.1 Pre-Scan Insertion Area Results 28
5.2 Pre-Scan Insertion Power Results 29
5.3 Pre-Scan Timing Results 32
5.4 Post-Scan Insertion Area Results 33
5.5 Post-Scan Insertion Power Results 33
5.6 Post-Scan Insertion Timing Results 37
5.7 Results From the IC Compiler 39

Chapter 1

Introduction

Digital beamforming is a concept first proposed in 1980 [1]. The advancement of beamforming allowed designers to configure the antenna's radiation pattern while not altering the physical dimensions of the antenna. The power of an antenna's signal is measured by its gain. The antenna will emit a radiation pattern that encompasses phase and amplitude. The phase and amplitude are parameters the designer can set. For an array of antennas their radiation patterns can constructively add or subtract, this enables a precise and stronger radiation pattern compared to a single antenna element. A widely used implementation of beamforming is null-steering beam. Radar technology and other communication applications can use null-steering to block a signal from being transmitted or received [2].

Analog beamforming uses a similar configuration as digital beamforming with elements spaced at half wavelengths of each other. The disadvantages of analog beamforming include the amount of space needed to package the hardware. Analog beamforming uses power dividers and phase shifters for each element of the array and for every beam created which is cumbersome. Digital beamforming performs the calculations after the signal has been digitized through and Analog to Digital Converter (ADC). This allows for a compact and efficient system. The

operations needed to perform the computations of beamforming require either application specific integrated circuits (ASIC) or a field programmable gate array (FPGA) [1, 3]. The success of digital beamforming is constrained by the sample rate and power needed by the ADC. Complementary metal oxide semiconductors (CMOS) allow higher speed and less power consumption for ADCs. Most digital beamforming configurations have an ADC set behind the RF front end of all the elements in a phased array of antennas.

1.1 Research Goals

This research is a proof of concept that digital beamforming can be applied to digital hardware. The goals set for this digital beamforming implementation on hardware involved generating a behavioral system of the algorithm in software using MATLAB, simulate beamforming of an array of elements receiving a generated input signal and calculating the steering vectors. After verifying the functionality of the software algorithm the hardware description language (HDL) equivalent algorithm is implemented in Simulink and the results are compared to the behavioral algorithm. The HDL algorithm is validated and the HDL code is generated using an app in Simulink called HDL Coder. The HDL code is transferred to a logic synthesis environment. Using Synopsys Design Compiler HDL code is synthesized for hardware implementation. Scan ports are added to the design to for testing. The scan insertion will provide power consumption, timing analysis and hardware area used. Lastly, the HDL beamforming model is placed and routed on 4 different ASIC technology sizes 32 nm, 65 nm, 90 nm and 180 nm.

1.2 Contributions

The significant contributions to the projected are listed below:

1. Generated HDL files from MATLAB using the application HDL Coder.
2. Simulated the beamforming output from Simulink and Cadence Incisive simulation environment.
3. Synthesized the HDL project with and without scan insertion for data analysis.
4. Routed the design using Custom Compiler.
5. Synthesized the HDL project on different ASIC hardware technology for data analysis
6. The obtained information is analyzed and is presented using graphs and charts.

1.3 Organization

The structure of the thesis is as follows:

- Chapter 2 Bibliographic Research: This section will explain the theory behind beamforming and use of phased arrays. Different beamforming techniques will be described that have been implemented on digital hardware. Discussion of how digital hardware is implemented for beamforming algorithms and the different hardware platforms that are most commonly used.
- Chapter 3 Description of the Project: Chapter 3 describes the Simulink model generated to model phase-shift beamforming and the corresponding hardware algorithm. This section will explain how the HDL files are generated using Simulink.
- Chapter 4 Simulation Results: The results presented in this section show the beamforming output from the Simulink behavioral model as well as the HDL equivalent algorithm

output. Next, the results from creating a co-simulation test bench environment within Simulink and using Cadence Incisive simulation environment.

- Chapter 5 Results from Hardware Implementation: This section will describe data analyzed from synthesizing the HDL code with and without scan insertion. Results from routing the design on an ASIC. Lastly, data analysis from implementing the design on different ASIC technology.

- Chapter 6 Conclusion: This section will summarize the results and analysis of the beamforming model implemented in hardware. Additionally, this section will include future work on advancements that can be implemented with digital beamforming.

Chapter 2

Bibliographical Research

2.1 A brief overview of beamforming and phased arrays

Antenna theory is broad and extensive. It is crucial to comprehend the fundamentals of antenna knowledge and the capabilities of an array of antennas for the research done in this work. A single antenna provides a wide radiation pattern with a low gain. For signal processing it is imperative to have a high gain when transmitting information. The gain of an antenna characterizes the antennas directivity and radiation efficiency. If the gain of an antenna is high that signifies the input power applied to the antenna is almost completely converted to the electromagnetic wave produced. A lower gain implies power inefficiency this results in a weak electromagnetic wave since power has been lost [4, 5].

By increasing the physical dimensions of the antenna the gain can increase if the antenna can output more power. This leads to a higher directivity but impractical for some implementations. Instead of changing the physical dimensions of an antenna, using an array of multiple antenna elements together can create high gain and directivity. The elements are used in a specific electrical and geometrical configuration to allow maximum efficiency. To create directive

patterns with an array of elements the radiation field of each element should constructively interfere with one another. When designing the directivity of a phased array of antenna elements the engineer has parameters to optimize. These variables include the configuration of the array, the research done in this paper uses a linear configuration. The distance between each element is set by the designer and consistent between all elements. The amplitude and phase of each antenna can be independent of one another and affect the directivity of the resulting radiation pattern. Lastly, the relative pattern of radiation can be manipulated for each element [6] [7].

The digital beamforming studied in this paper uses MATLAB's Simulink narrowband receive element array design block. The phased array creates a beam of radiation resulting from each antenna elements phase and amplitude directed towards the incoming signal's location. For this experiment the direction of the incident angle is known and used in the beamforming algorithm. An overview of different beamforming algorithms and hardware platforms are explained below.

2.1.1 Beamforming Algorithms

The received signals from each antenna in a phased array are multiplied by complex weights that can be individually created for each antenna element. This allows the gain of the whole phased array to be manipulated electronically [8]. The conventional beamforming techniques are based on digital signal processing theory. Convolution is used to find the response of the phased array of elements from some input signal received from the phased array. The system is linear time invariant (LTI). The theory of superposition is used to find the individual responses from each input signal as a weighted sum of the signals applied to each individual signal. Moreover, the theory of linearity deals with complex exponentials that are realized in space-time signals [4].

2.1.1.1 Direction of Arrival and power method

In a receiving phased array the direction of arrival of the input signal is used to calculate the steering vectors to change the direction of the radiation pattern. If the direction of arrival (DOA) is not known there is an additional algorithm to estimate the DOA. The research done in [9] developed an algorithm to estimate the DOA. A projection matrix is created from the vectorized phase and magnitude of the input signals which are found using an autocorrelation matrix. Once finding the largest projection values of this matrix these will be used as estimates for the DOA.

A uniform linear array (ULA) of antenna elements are the most common configuration for beamforming. The number of antenna elements, the incident angle of the signal intercepting the antenna, and the white noise that the phased array also receives are all part of a mathematical relationship. To filter the signal spatially amongst the array of antennas, weights are applied to each output signal [10–12].

2.1.1.2 Null-steering techniques

Many applications of beamforming include technology for the military. When communicating it is important to maintain a safe and reliable signal that can withstand adversarial attacks. Using nulls at certain angles causes no radiation towards that direction. Null-steering deflects any incoming signals that are deemed undesirable to the system. The research done in [13] uses a population-based, evolutionary, and stochastic method to assign certain coefficients to each antenna to create a desired radiation pattern [14].

2.1.1.3 Adaptive Beamforming

There are blind beamforming algorithms that do not need a reference signal to train the algorithm where the direction of arrival is. With no reference there is less computation needed. For non-

blind algorithms the reference signal is required and will slow down the overall computation however, the non-blind algorithm converges faster than the blind algorithm [15, 16]. Research conducted in [17] acknowledges the advantages of both algorithms and implements hybrid beamforming. This hybrid beamforming research combines blind and non-blind algorithms with a modified normalized least mean square normalized constant modulus algorithm. The system will start at convergence then accept signals from the phased array and generate an output through a normalized least mean square algorithm. This research done is adaptive since the system will iterate this process and update the weights assigned to each antenna based off the output found.

2.1.1.4 Phase-shift beamforming

There have been many different variations of beamforming applied to hardware systems that were considered before applying a specific beamforming technique [18–23]. The algorithm used in the research conducted in this paper uses Simulink's narrowband receive antenna array design clock. A linear time invariant filter is applied to all the antenna elements to obtain an output array. After applying the LTI filter the outputs are added together as shown in Figure 2.1.

When using a narrowband array of antennas the signals are all approximated by a phase shift in the frequency domain. This is equivalent to a delay in the time domain. Equation 2.1 shows the input field $f(t, p_n)$ as the input signal. Individual antenna elements are multiplied by a time delay $e^{jw_c t}$. There are N elements. Next, the travel time of a plane wave between two elements is calculated as τ_n. Equation 2.2 shows the narrowband antenna elements multiplied by the time delay. Multiplying by a time delay in the time domain is equivalent to a phase shift in the frequency domain [4].

$$f(t, p_n) = \sqrt{2} Re\left\{ f(t, p_n) e^{jw_c t} \right\}, n = 0, ..., N-1 \tag{2.1}$$

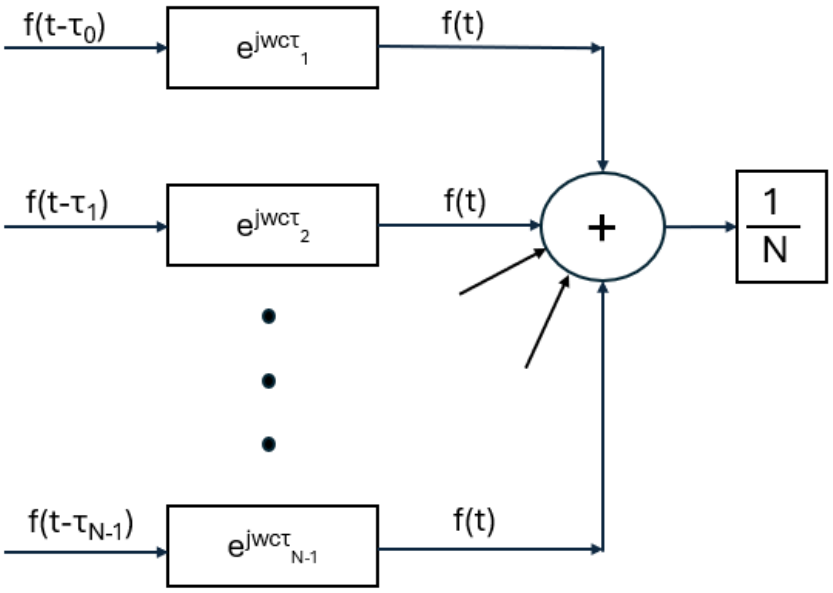

Figure 2.1: Delay and Sum Beamformer

$$f(t,p_n) = \sqrt{2}Re\left\{f(t,p_n)e^{-jw_c\tau_n}e^{jw_ct}\right\} \tag{2.2}$$

The delay and sum method which uses time delays is displayed in Figure 2.1. The functions of time shifts are put into vector form and weighted individually to form the steering vectors needed for the resulting beam.

2.2 Digital Hardware Implementation

The research conducted in this paper involves targeting a field programmable gate array (FPGA) through MATLAB. The hardware description language (HDL) is generated by MATLAB and then implemented on different application specific integrated circuit (ASIC) technology. The main difference between ASIC and FPGA platforms, is ASIC platforms cannot be repro-

grammed and draw less power than FPGAs. FPGA contains the word "field", meaning the programming of the device is done outside of a factory and in the field. Within FPGAs there are simple logic gates such as AND, OR, XOR or more complicated logic such as encoders and decoders. Incorporating a microprocessor embedded in the FPGA has become quite popular. These processors connected to the FPGA are called system on chip (SoC) boards. Whereas, if a processor core is implemented within the FPGA such as Xilinx's Microblaze this is considered a "soft" processor core alternative [8, 24] .

2.2.1 Types of hardware platforms

2.2.1.1 Field Programmable Gate Arrays

Beamforming algorithms require complex mathematical computations since the electromagnetic wave has phase and amplitude components, which requires the hardware to deal with real and imaginary numbers. Additionally, when receiving information from the analog world it is necessary to use an analog to digital converter (ADC) that can digitize the received signal without losing any precision. The research conducted in [25] relied on a MATLAB program to generate the weights applied to each antenna element then convert these values to floating point and assign these to registers in an FPGA. The digital signal processor is described in the next section however, it is important to note that the FPGA can outperform a digital signal processor (DSP) by as much as 1000:1 [8].

2.2.1.2 Digital Signal Processor

A Digital signal processor (DSP) is a processor or microprocessor able to provide fast sequences of instructions with emphasis on operations related to digital signal processing. DSPs are able to represent mathematical operations such as shift or multiply digitally. The DSPs used in

designs today are able to multiply and accumulate (MAC) in a single instruction cycle. DSP architecture incorporated the multiply-accumulate processors within the data path. Additionally, some processors have numerous multiply-accumulate processors within the data path to perform these calculations in parallel. DSPs are able to complete many tasks concurrently. For example fetching an instruction, accessing memory to obtain an operand, or storing a result in memory can all be done in the same machine cycle.

2.2.1.3 Application Specific Integrated Circuits

Depending on the system ASICs are more appealing than FPGAs. The digital beamforming within an FPGA involves many signal channels that are put through analog to digital converters and there corresponding interfaces. Other implementations include multiple analog beamformers fed to a digital beamformer. These implementations are notable however the FPGA image quality differs slightly compared to an ASIC platform. The power consumption of a digital beamformer in an FPGA is high and must be mitigated. The research conducted in [26] decided the digital beamformer power consumption did not outweigh the quality of the resulting beamformed signal and decided to use ASIC instead. The beamforming algorithm involved adaptive-resolution on an energy-scalable ASIC. This method uses a sliding window that will minimize the memory storage of input data. The resolution of the sliding window can be adjusted to decrease the memory storage constraints needed.

Chapter 3

MATLAB and Simulink Environment

A system that generates a transmitting signal to a phase shift beamforming algorithm is created in Simulink. This model contains another algorithm that is equivalent to the Simulink phase shift beamformer block except that it uses hardware description language functional blocks that are synthesizable for a digital hardware platform. This section will describe the behavioral and HDL algorithms created in Simulink. Both algorithms implement a phase shift beamforming method to calculate a simulated received signal. Simulink's phase-shift beamformer is applied to an array of ten narrowband receiver antenna elements. The output of this algorithm is sent to a waveform window and compared to the HDL equivalent phase-shift beamformer. The full system is shown in Figure 3.1.

3.1 Behavioral Algorithm

The behavioral or software based beamforming algorithm involves generating input signals at a known angle to a ULA of 10 elements. MATLAB created a phase shift beamformer function. This function will estimate a delay in time to perform beamforming. This function has many

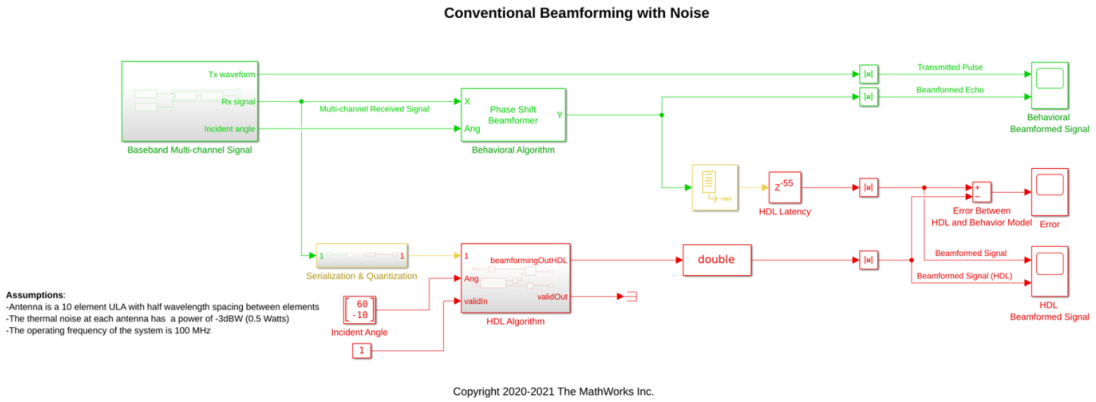

Figure 3.1: Simulink Beamforming System

parameters such as the as the number of elements, distance between elements, and type of antenna element used. The user must also specify the propagation of the signal, the frequency that the system will operate at. For the beamforming algorithm the direction of arrival must be known, this implies that this method is conventional beamforming since it must start with having the direction of arrival. The output of this function will provide the dimension of the input signal in matrix form along with their corresponding beamforming directions, and the coefficient weight that is applied to each vector. This output is then sent to a waveform window along with the output of the HDL algorithm for comparison. The full system in Simulink is shown in Figure 3.1 [27–29].

3.2 Hardware Description Language Algorithm

3.2.1 Digital Conversion of the signal information

The behavioral algorithm mentioned above is a floating point model whereas, the equivalent HDL model uses fixed point arithmetic. From Figure 3.1 there is a waveform design block that generates a multi-channel signal. After the multi-channel signal is generated there are

antenna elements that capture the signal and the received target echo that is generated from the incident angle. The signal received is then sent through the receiver pre-amplifier this will aid in reducing the noise captured.

After capturing the analog floating point information the data must be converted to fixed-point which is done through a quantize signal block from Simulink. The 10 different channels that each have 300 samples are converted to 12-bit words with 9-bits for fractional precision. The conversion from floating to fixed point is constrained by the target FPGA platform. The target FPGA is a Xilinx Virtex-7. The Virtex-7 contains a 12-bit analog to digital converter. Converting the data to 12-bit word length was needed to meet the ADC constraints. The next part of the HDL algorithm involves the phase-shift beamforming algorithm. The multi-channel input signal is converted to a 12 bit word and processed serially by the HDL algorithm. The computations required of this algorithm require delays to enable synchronization. If there are any processes that are implemented in parallel the algorithm will compensate for these delays by adding the equivalent delay to the parallel function.

3.2.2 Generating the Steering Vector

From the input signals direction of arrival the HDL algorithm will create a steering vector. The steering vector is obtained by sampling the angle at each antenna element and aligning the data in a matrix. Using matrix multiplication the information from the multi-channel input signal can be found at each element by using the position of each element within the computation. The data obtained by the array of elements will be complex and require computational delays added to the system. For this specific application the CORDIC algorithm is used. The steering vector implementation in hardware is shown in Figure 3.2.

The CORDIC algorithm or coordinate rotation digital computer, is used to calculate trigonometric functions as well as other complex mathematical operations with an arbitrary base. In

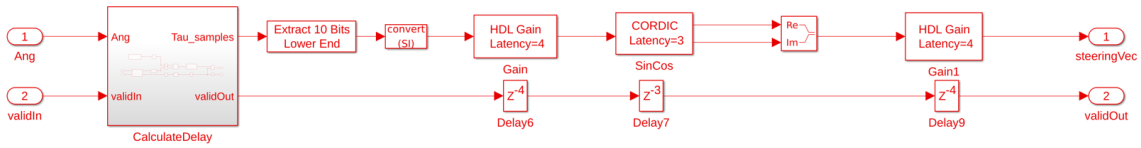

Figure 3.2: HDL Logic to Find Steering Vector

most cases this algorithm is used in lieu of a hardware multiplier. The CORDIC system is necessary for many beamforming algorithms and implemented in hardware to minimize the number of gates needed.

The spacing between elements is used to find the steering vector. The position is based off the center array and measured outward. Spacing between elements is half a wavelength which is roughly 1.5 meters or half the propagation speed divided by the operating frequency. The operating frequency is 100 MHz and used to sample the incoming signal. The pulse generated by the phase shift beamformer is 1 kHz. The spacing is represented by an 8-bit word and 4-bit fractional length. The result of the HDL algorithm is pulses of the beamformed signal this is shown in the results section.

MATLAB's co-simulation environment is used to compare the result from the behavioral algorithm and the HDL algorithm. This platform implemented in Simulink will support co-simulation between different servers. The co-simulation platform can link Simulink and another simulation environment such as Cadence Incisive. The implementation of the co-simulation environment is shown in Figure 3.3.

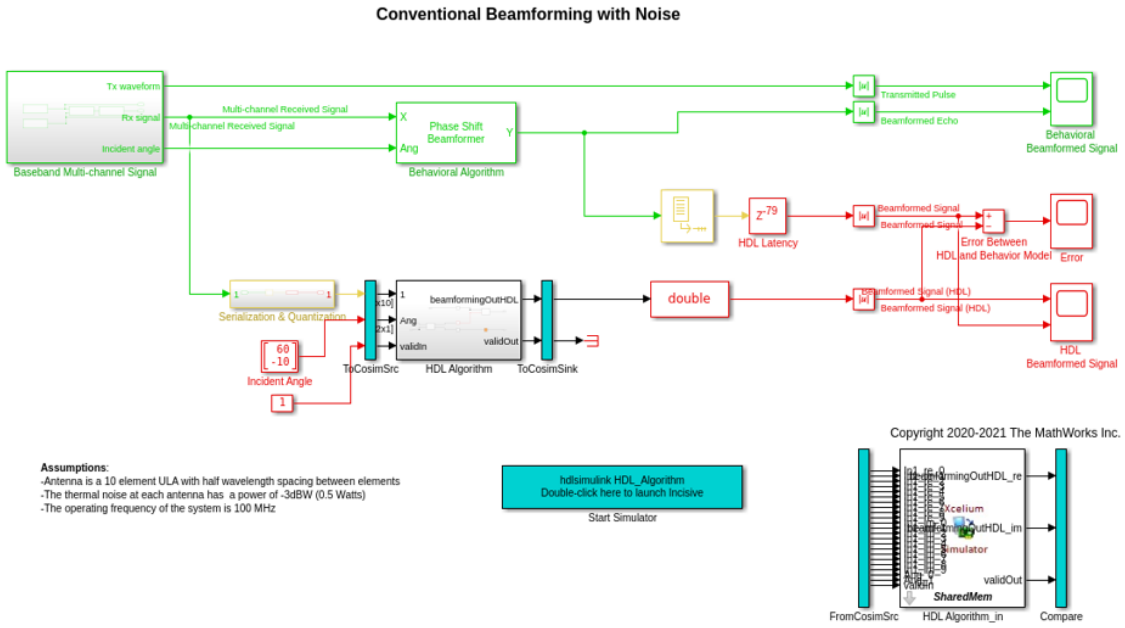

Figure 3.3: System with the Co-simulation Platform

3.3 Generating HDL code

3.3.1 Target Platform

The Simulink model targets specific FPGA boards. The target used is a Xilinx Virtex-7 FPGA with xc7vx485t microprocessor. The package is ffg1761 with a target frequency of 300 MHz. The Virtex-7 is part of the 7 series family of boards. It has the largest number of logic cells in the family at 1955K cells. It has 68 Mb of RAM, 2784 Gb/s serial bandwidth and 1200 input/output pins. This board is referenced earlier as a soft core FPGA since it has a microprocessor embedded within the FPGA. The Virtex-7 has the MicroBlaze CPU. The device xc7vx485t includes 485,760 logic cells and 2,800 DSP slices this information is found in the data sheet [30]. The device-package ffg1761 used with the xc7vx485t has 700 inputs and outputs.

3.3.2 HDL and Testbench Generation

The HDL Coder is an application created by MATLAB and used in the Simulink environment. This application has parameters set by the designer. Within the HDL Coder properties the HDL Algorithm system is selected for HDL code generation, the language chosen is Verilog. After implementing all necessary parameters for the target platform explained in 3.3.1 the designer can choose specific parameters to increase optimization of the design. The choices for optimization include choosing the size of RAM for register mapping, removing redundant registers from the design and timing constraints for the model.

A testbench is automatically generated from HDL Coder. Cadence Incisive simulator is used for the HDL testbech as well as for the co-simulation model. The co-simulation will link the Simulink simulation with the HDL compatible simulator in this case Cadence Incisive. The HDL Algorithm will have two interfaces that communicate to the third party simulation environment this is shown in Figure 3.3 [28, 29].

Chapter 4

Simulation Results

4.1 Simulation Results

The behavioral beamforming algorithm using Simulink blocks creates a vectorized beamformed output. The result of the behavioral beamforming algorithm is shown in Figure 4.1. The generated signal is in blue and shows distinct pulses at 300 and 600 milliseconds. There is noise that is received from the antennas and carried through the system to the output however, this can be mitigated with a high pass filter for higher quality results. It is important to note that the signal to noise ratio (SNR) of the beamformed pulse is 1.2:0.5 or 2.4. For many systems this SNR is adequate.

The result from the HDL algorithm will be compared to the output from the behavioral algorithm to verify that the HDL implementation of beamforming is equivalent. The result from the HDL and behavioral models are shown in Figure 4.2.

From looking at Figure 4.2 it is not easy to discern the difference between the behavioral and HDL beamformed signal. The error between the two signals is shown in Figure 4.3

Figure 4.3 shows the error between the behavioral and HDL beamformed signals are in the

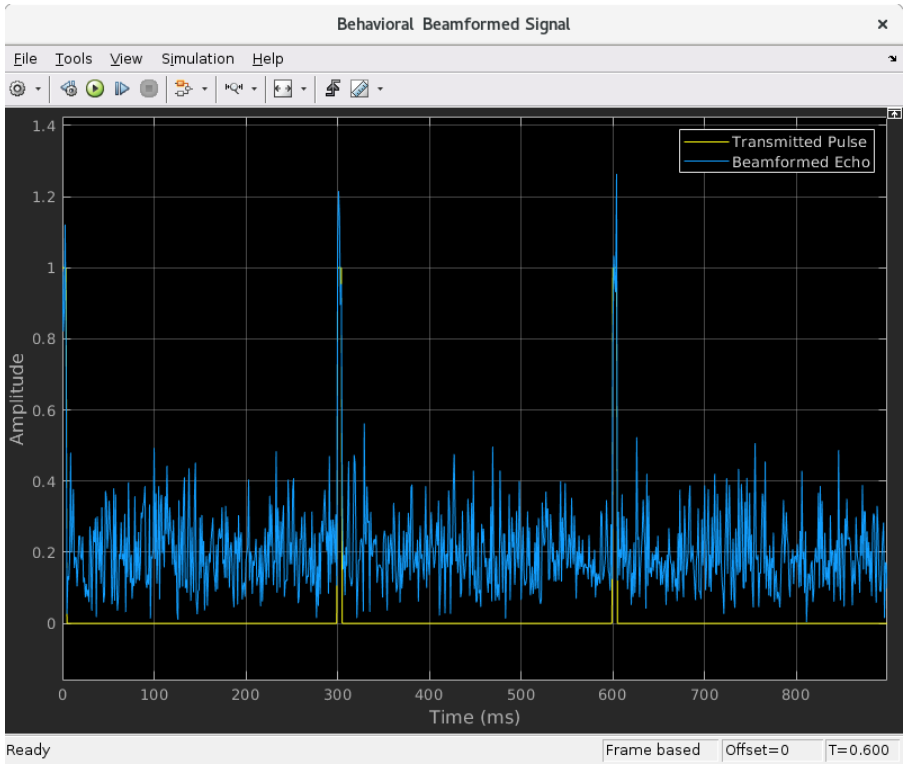

Figure 4.1: Behavioral Beamformed Signal Result

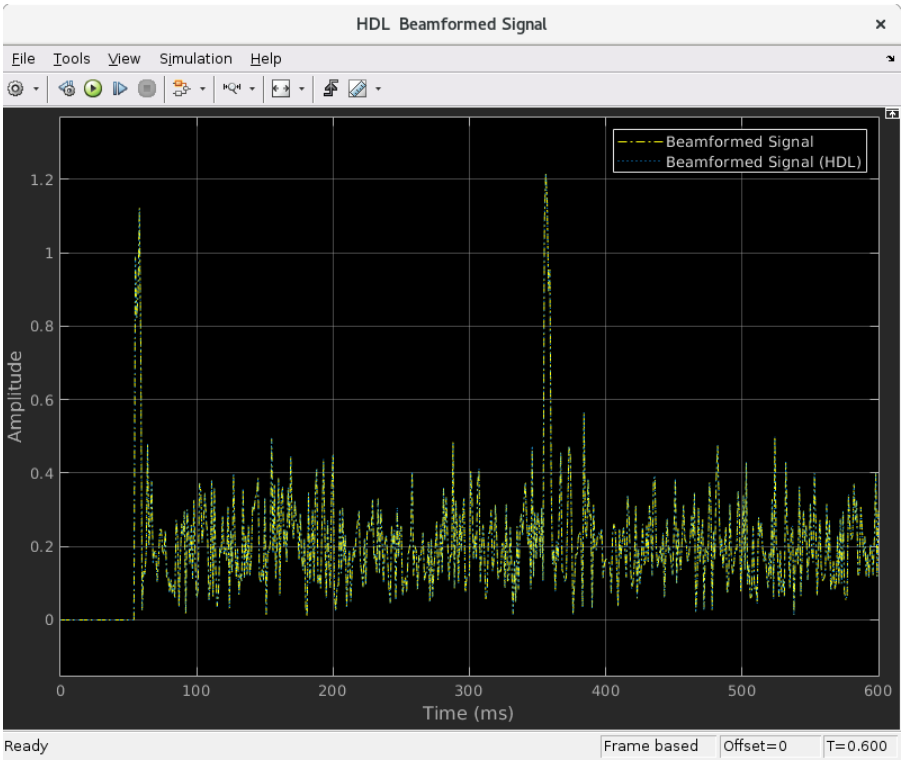

Figure 4.2: Output Signals from the HDL and Behavioral Models

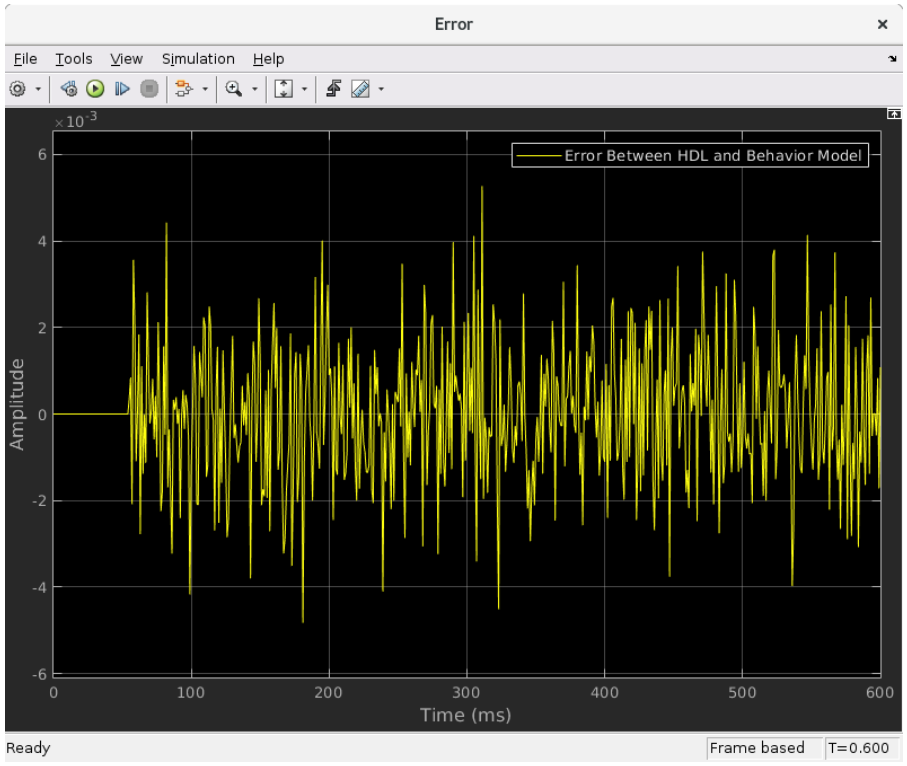

Figure 4.3: Error between Behavioral and HDL Beamformed Signals

order of 10^{-3} for this research this is statistically insignificant. Since the error is so small the HDL algorithm's functionality has been verified and can now be generated into HDL code.

4.2 Co-simulation Results

A co-simulation testbench is created to run the behavioral simulation in Simulink while running the HDL simulation in Cadence Incisive. Unfortunately, this was not able to work successfully. There is an issue connecting Simulink to the Cadence Incisive simulator. There are .sh files that are generated that compile the HDL files and the testbench file. After compiling there is a separate .sh file to simulate this project. Running these scripts on their own work and the results are shown in section 4.3. The error when running the co-simulation environment in shown in Figure 4.4. The error states that the simulator library is not using shared memory. To share memory between the simulator library and MATLAB a server using inter-processing communication called HDL Daemon is used. Unfortunately, even with this server enabled the co-simulation environment does not run. In Figure 4.5 a MATLAB function nclaunch() is used to launch the Cadence Incisive simulator environment from the MATLAB terminal. Within this function the socket used for the HDL Daemon server is set at 4449. After running this function, the new error states the version of Incisive cannot be determined. After checking the simulator is on the path of the machine the nclaunch() function has the same error. Even though the co-simulation environment was not achievable the HDL Algorithm can be verified using a separate environment with Cadence Incisive to simulate the HDL files.

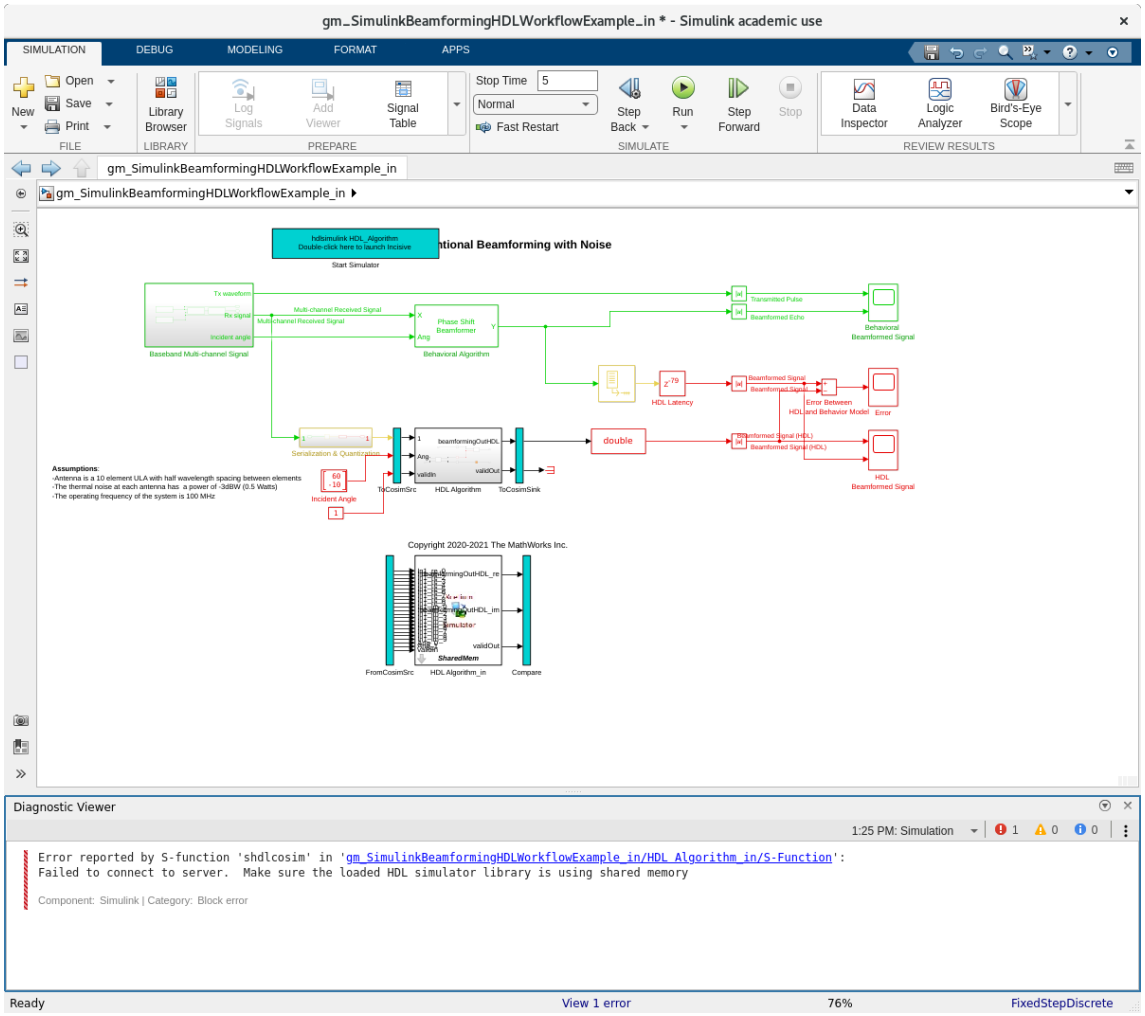

Figure 4.4: Co-simulation Environment

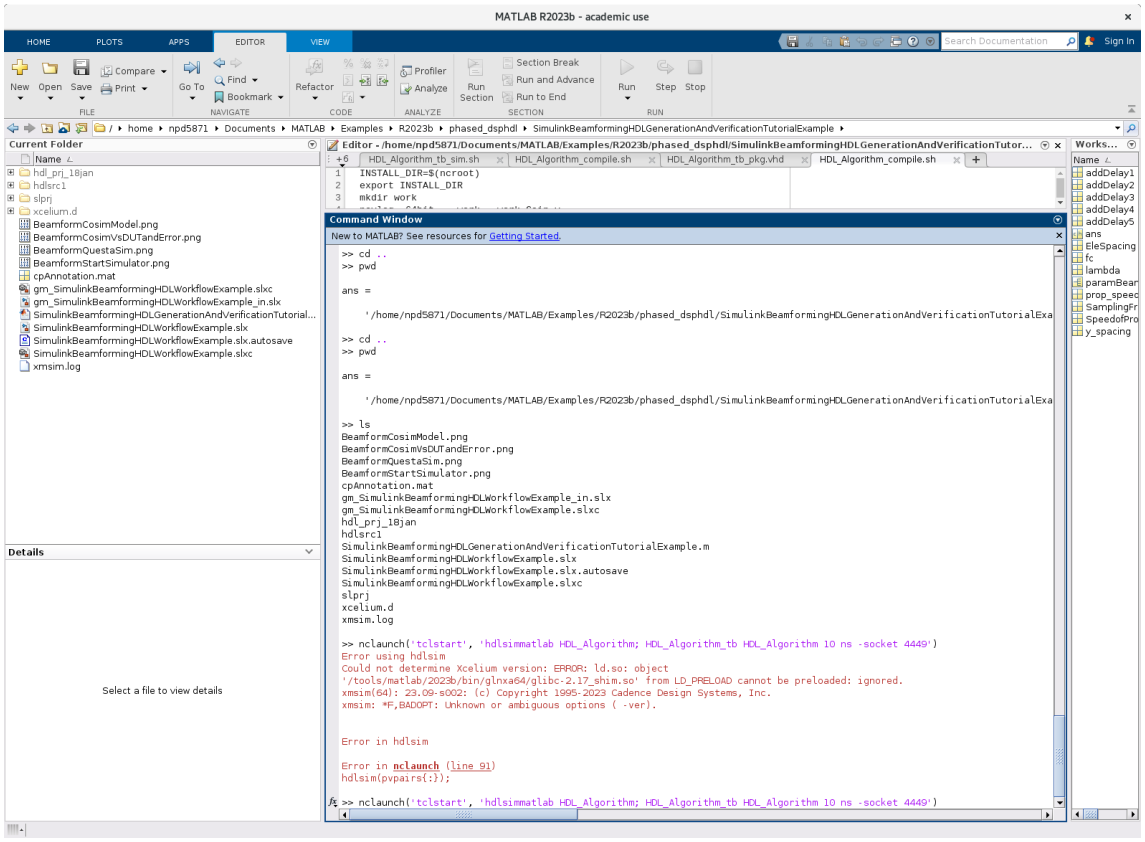

Figure 4.5: Co-simulation Error Console

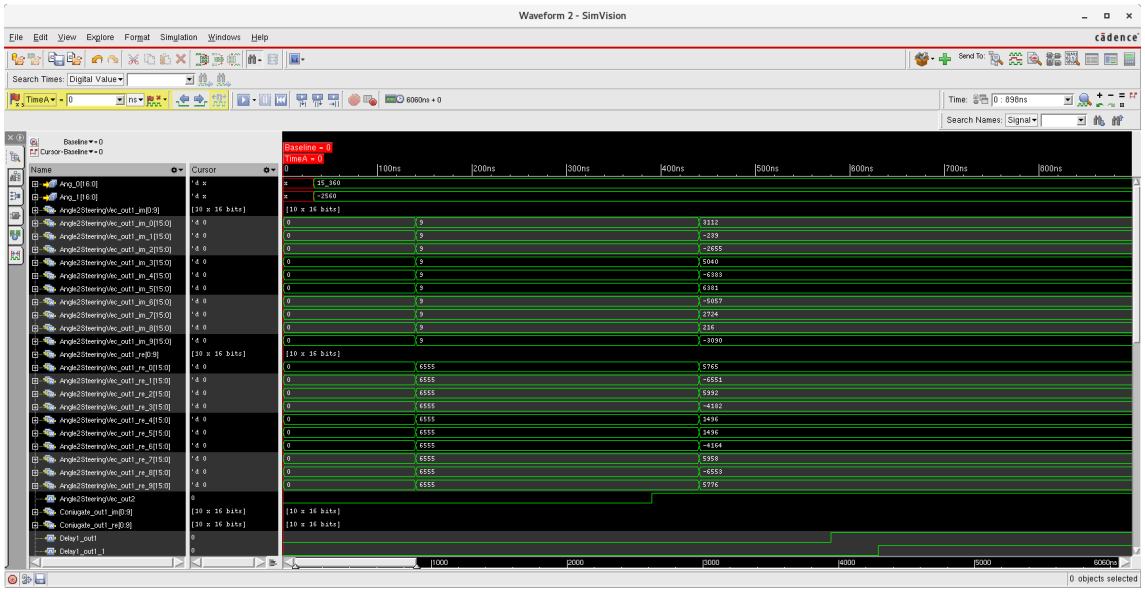

Figure 4.6: Waveform of Generating Steering Vectors

4.3 Testbench files

The generated Verilog files are implemented in a test environment that can communicate with Cadence Incisive simulator environment. The figures below Figure 4.6 and Figure 4.7 show the operation of the HDL Algorithm model when test signals are applied. The testbench used for this model generates a 16 bit word as the angle to the 10 antenna elements. This angle is then used in modules within HDL Algorithm. The testbench will send 12 bit input signals as the input pulse. The testbench will generate a 16 bit angle value that will be used in the phase shift algorithm. The pulse signals are sent to the Angle2SteeringVec.v module that will output the steering vector results this is shown in 4.6. This waveform shows the resulting steering vectors for each antenna element 0-9 with corresponding real and imaginary parts.

The resulting beamformed signals are beamformingOutHDL_im and beamformingOutHDL_re which are each 32 bit signals with values for the imaginary and real components of the beamformed output signal. The values of the resulting signals change rapidly to show the

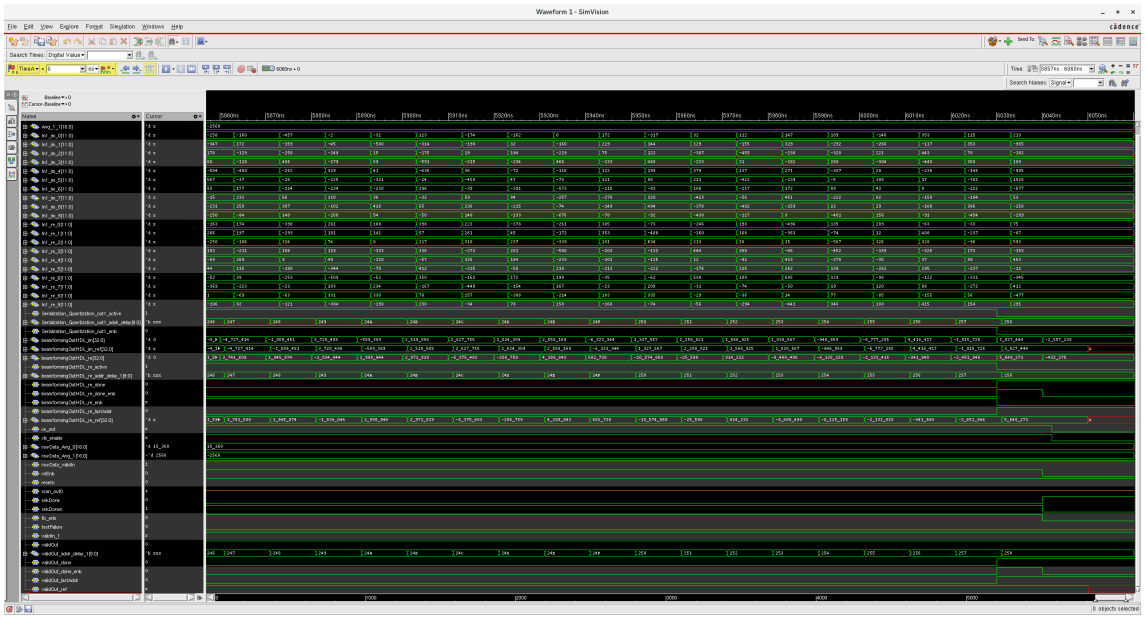

Figure 4.7: Waveform of the Beamformed Signal

computations in progress. Once the stimuli is done inputting information the done signal is set high once the algorithm calculates the result as shown in Figure 4.7.

Chapter 5

Hardware Results

The synthesis tool from Synopsys called Design Compiler (DC) is used for this design. This will generate report files to analyze the performance of the circuit. The synthesis tool is used to optimize the design for timing, area and power. Instead of implementing the design on an FPGA board the design is implemented on different ASIC technologies. The Virtex 7 which is the target platform described in Section 3.3.1 is not available for use in this project. The synthesis tool is applied to the design for each technology available. The 32 nm, 65 nm, 90 nm and 180 nm ASIC libraries are used for implementation.

After analyzing the data obtained from the DC tool the design is placed and routed using the Synopsys' tool Integrated Circuit Compiler (ICC). This tool will use the synthesized design to route signals and place components on up to 11 different metal layers. This tool will optimize the design to efficiently route all signals while optimizing timing, power and area.

Table 5.1: Pre-Scan Insertion Area Results

Technology (nm)	Total Area (nm^2)	Total Cell Area (nm^2)
32	538153.198	350733.4628
65	undefined	384922.8007
90	1932694.71	1665971.733
180	undefined	3026954.154

5.1 Synthesis

Synthesis is used for digital designs to convert Hardware Description Language into a netlist. The netlist will describe the hardware as gates and wires to connect signals to each other. The DC compiler tool from Synopsys provides optimization of timing, area and power. After this optimization the tool is able to correlate the results to within 10% of what the physical implementation would be. This tool is comprehensive and includes PrimeTime, DesignWare IP, DFTMAX and Power Compiler [31]. The PrimeTime Suite offers static timing analysis. For designs 90 nm and below PrimeTime will improve signal to noise ratio and delay caused by cross talk [32]. The Power Compiler is another Synopsys tool used to analyze consumption of static power of designs less than 90 nm. This tool will also analyze leakage, dynamic, multi-voltage and threshold voltage power use. The compiler will optimize the design around power consumption based off of the activity from the nets generated in the design. The results from using Design Compiler, PrimeTime, and Power Compiler are presented and analyzed in Section 5.2.

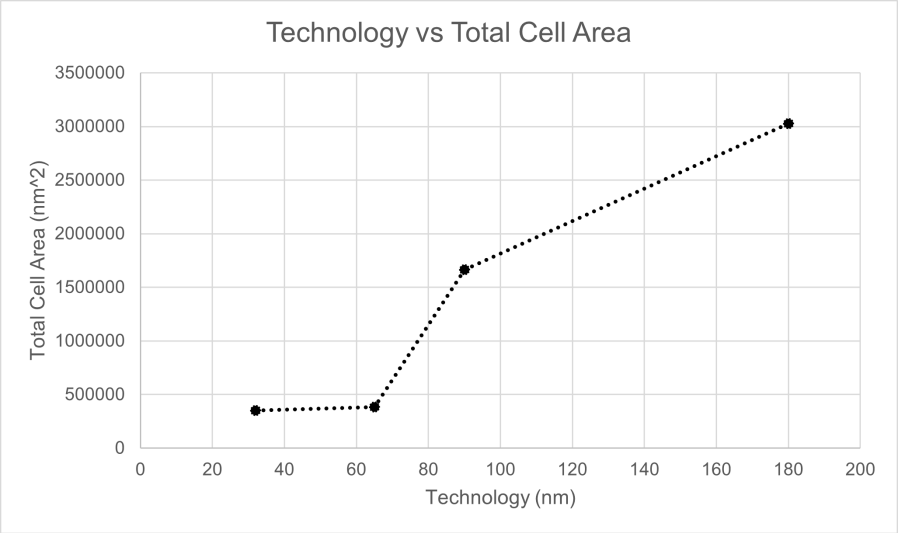

Figure 5.1: Pre-Scan Insertion Technology vs Area Graph

Table 5.2: Pre-Scan Insertion Power Results

Technology (nm)	Internal Power (mW)	Switching Power (mW)	Leakage Power (mW)	Total Power (uW)
32	16.4453	0.4035675	1.8743	18.72
65	22.2757	0.617233	0.0158717	22.9088
90	1.5168	0.706054	1.7255	3.9483
180	141.8984	11.5947	0.013207	153.5083

5.2 Pre and Post Scan Insertion Results

The results of the area analysis shows a steep increase from 65 to 90 nm technology. This could be the result of the Design Compiler not being able to accommadate designs larger than 90 nm.

The relationship between the different technology and power analysis shows a common trend of technology under 90 nm having a much smaller power consumption than power consumption over 90 nm. The results from the PrimeTime tool are shown in Table 5.3[33].

The PrimeTime synthesis tool within Design Compiler shows a very high discrepancy between the 32 nm and 90 nm technology. A possible explanation for the 65 nm technology

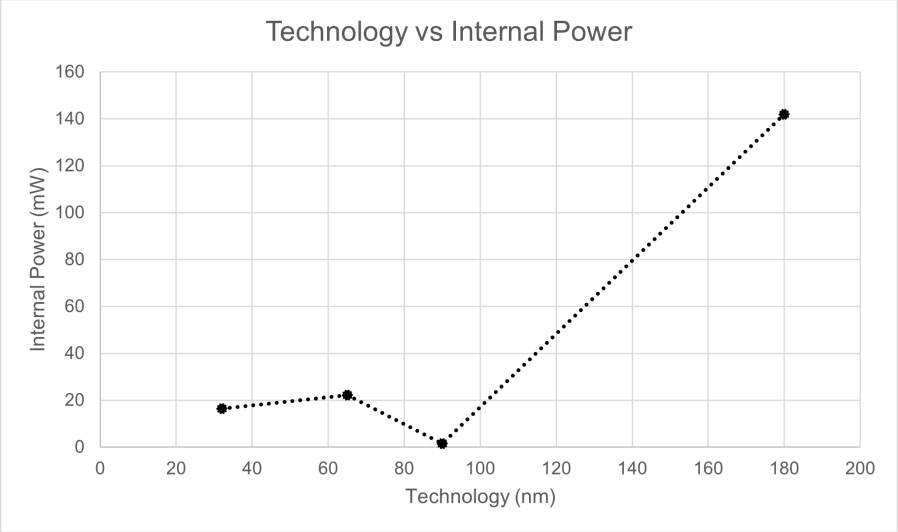

Figure 5.2: Pre-Scan Insertion Technology vs Internal Power Graph

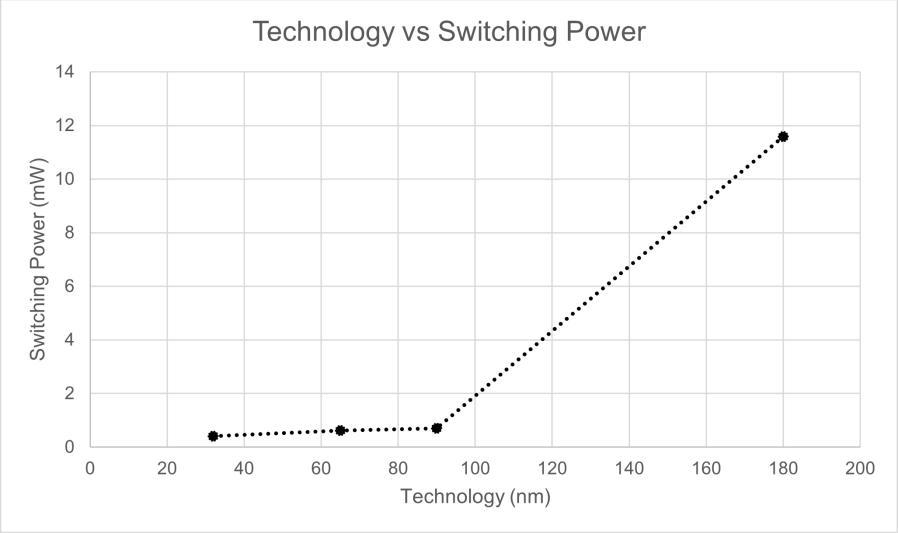

Figure 5.3: Pre-Scan Insertion Technology vs Switching Power Graph

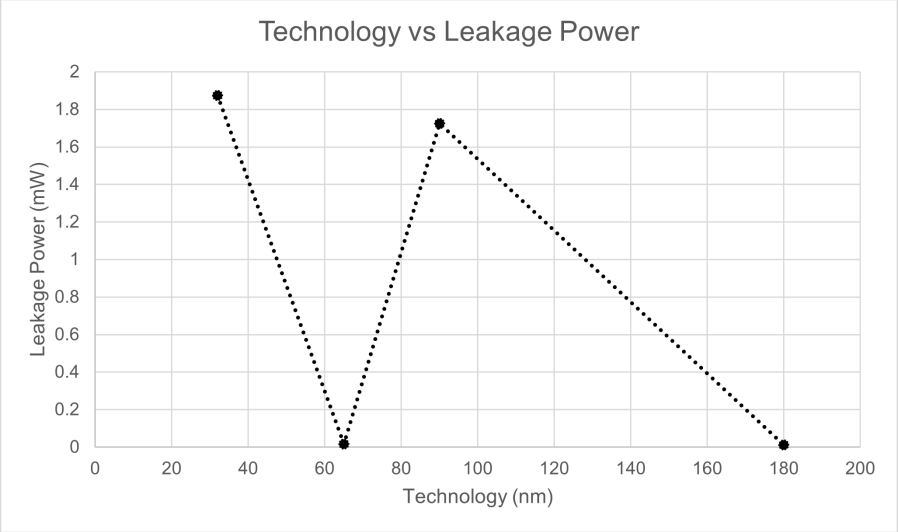

Figure 5.4: Pre-Scan Insertion Technology vs Leakage Power Graph

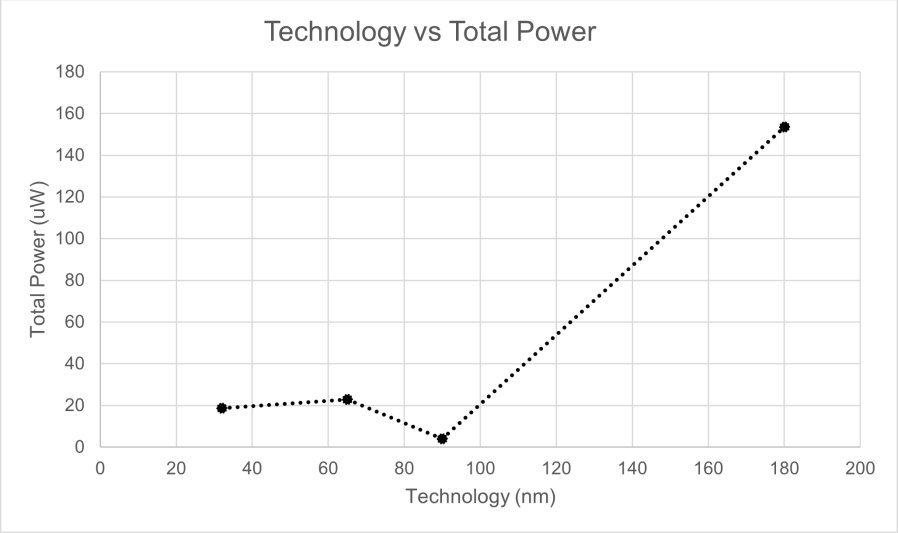

Figure 5.5: Pre-Scan Insertion Technology vs Total Power Graph

Table 5.3: Pre-Scan Timing Results

Technology (nm)	Slack
32	0.1855
65	6.493
90	0.0028
180	0.1352

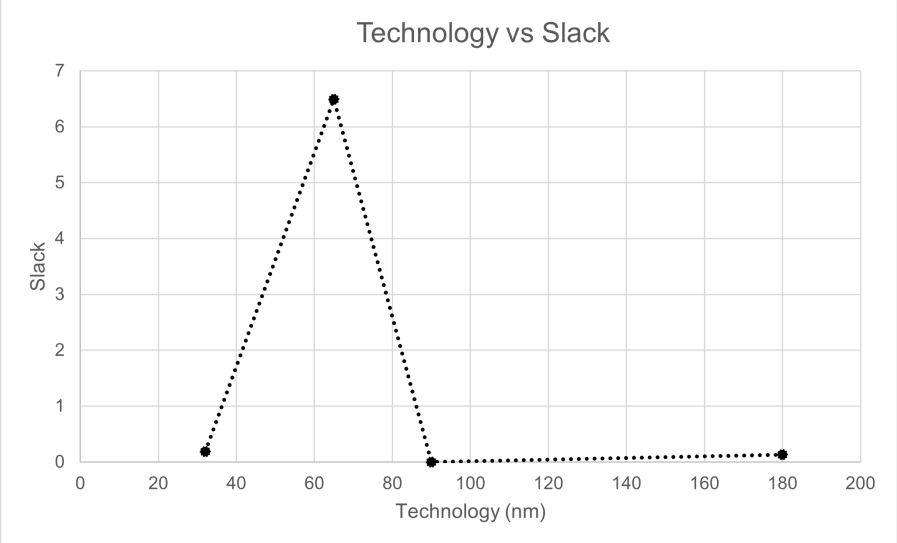

Figure 5.6: Pre-Scan Technology vs Slack

Table 5.4: Post-Scan Insertion Area Results

Technology (nm)	Total Area (nm^2)	Total Cell Area (nm^2)
32	662503.868	400531.5
65	undefined	447415.6
90	2224070.67	1802814
180	undefined	3404158

Table 5.5: Post-Scan Insertion Power Results

Technology (nm)	Internal Power (mW)	Switching Power (mW)	Leakage Power (mW)	Total Power (uW)
32	22.16	2.0554	28.216	5.2432
65	24.6949	0.6667155	0.015358	25.376
90	7.2836	3.7192	4.572	15.53
180	246.2298	39.003	0.015577	285.2479

having a much higher slack time could be the result of a less efficient technology library used compared to the other technology libraries.

The Design Compiler is also applied to the scan inserted design and the results from this tool are shown in Table 5.4 and Figure 5.7.

In Table 5.4 this shows the different area results from each technology which shows a similar pattern to the pre-scan insertion results from Figure 5.1. For all technologies used, the total cell area increased after scan-insertion. Implementing test insertion ports adds combinational logic such as flip flops and multiplexors which add to the total cell area. The results from post-scan insertion power analysis are displayed in Table 5.5 and Figure 5.8 through Figure 5.11.

The power analysis from post-scan insertion differ from the pre-scan insertion power results

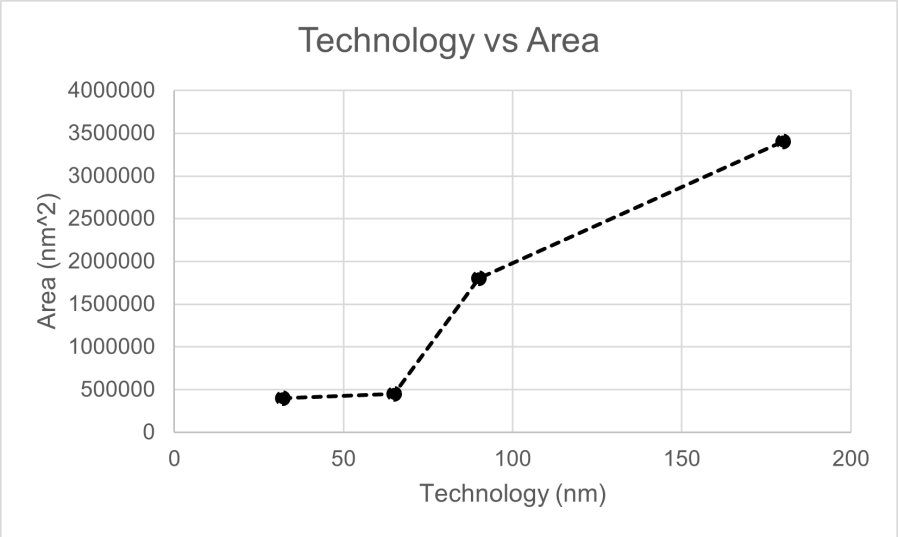

Figure 5.7: Post-Scan Insertion Technology vs Area Graph

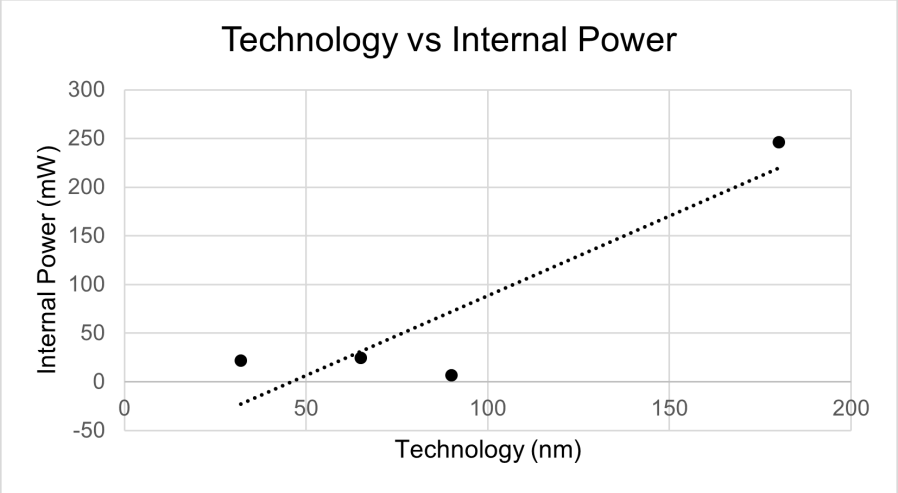

Figure 5.8: Post-Scan Insertion Technology vs Internal Power

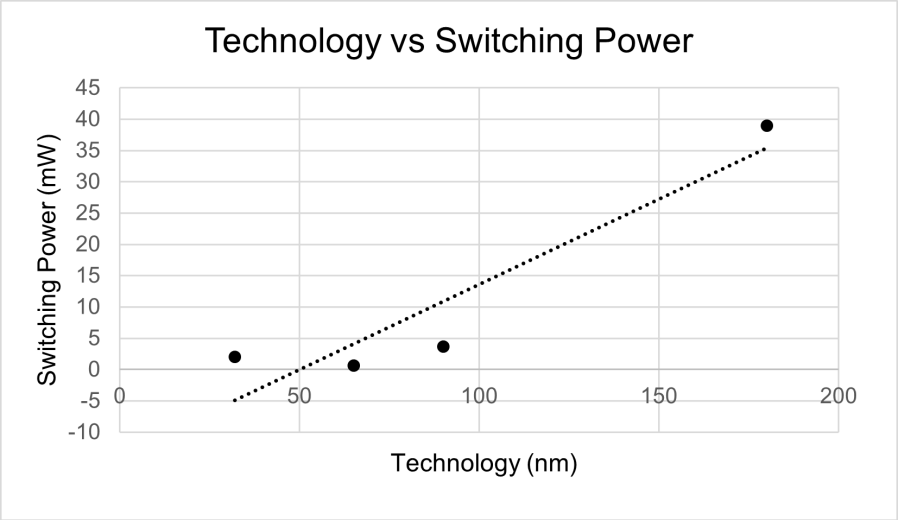

Figure 5.9: Post-Scan Insertion Technology vs Switching Power

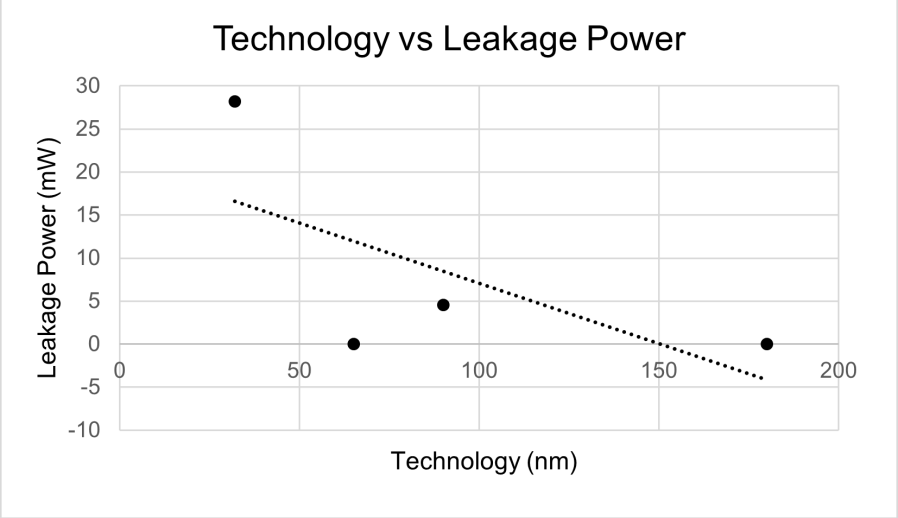

Figure 5.10: Post-Scan Insertion Technology vs Leakage Power

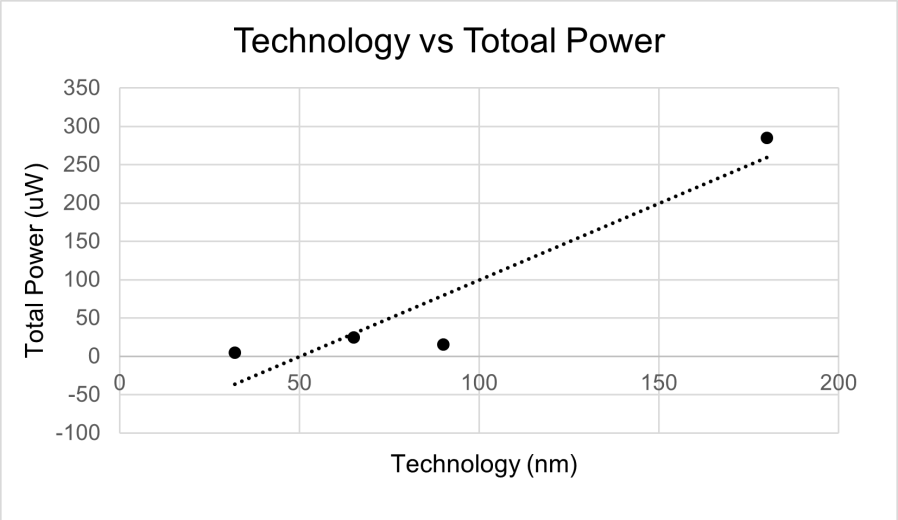

Figure 5.11: Post-Scan Insertion Technology vs Total Power

by increasing the total power for the 32 nm technology which changed from 18.72 mW to 5.2432 mW. The 65 nm Technology did not have a drastic change in power consumption between pre and post scan insertion. The 90 and 180 nm technology also shows significant increase in power by roughly a factor of 5 for 90 nm and almost a factor of 2 for 180 nm technology.

The last parameter analyzed by PrimeTime is slack time. This is shown in 5.6 and 5.12.

5.3 Integrated Custom Compiler Results

The Synopsys tool Integrated Circuit Compiler to place and route the digital beamforming design. The IC Compiler will plan the hierarchical design. This tool will efficiently find where congestion is most prevalent and reorganize the design. This compiler uses clock tree synthesis and route node convergence. The resulting design after the place and route tool is used is shown in Figure 5.13.

This shows the whole design routed with 11 different metal layers. For a detailed view of

Table 5.6: Post-Scan Insertion Timing Results

Technology (nm)	Slack
32	0.0138
65	6.6449
90	0.0298
180	0.0371

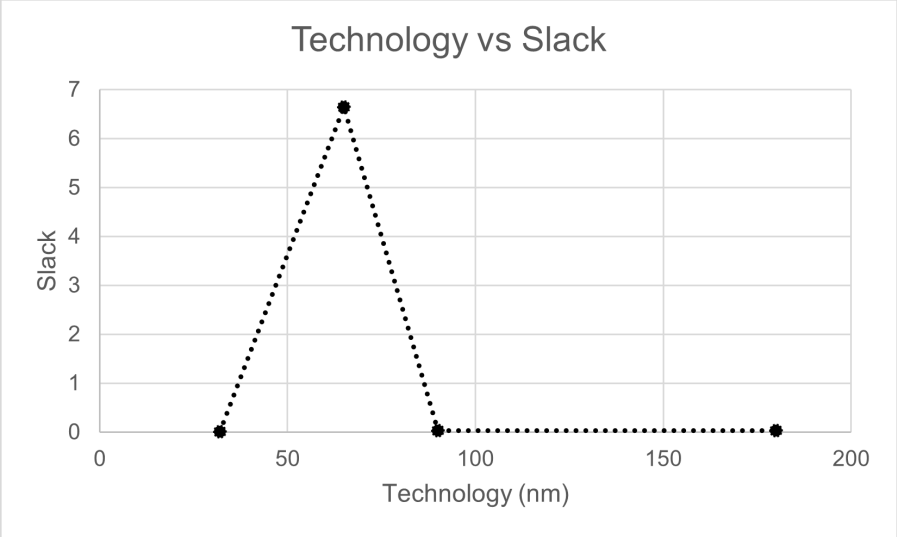

Figure 5.12: Post-Scan Insertion Technology vs Slack

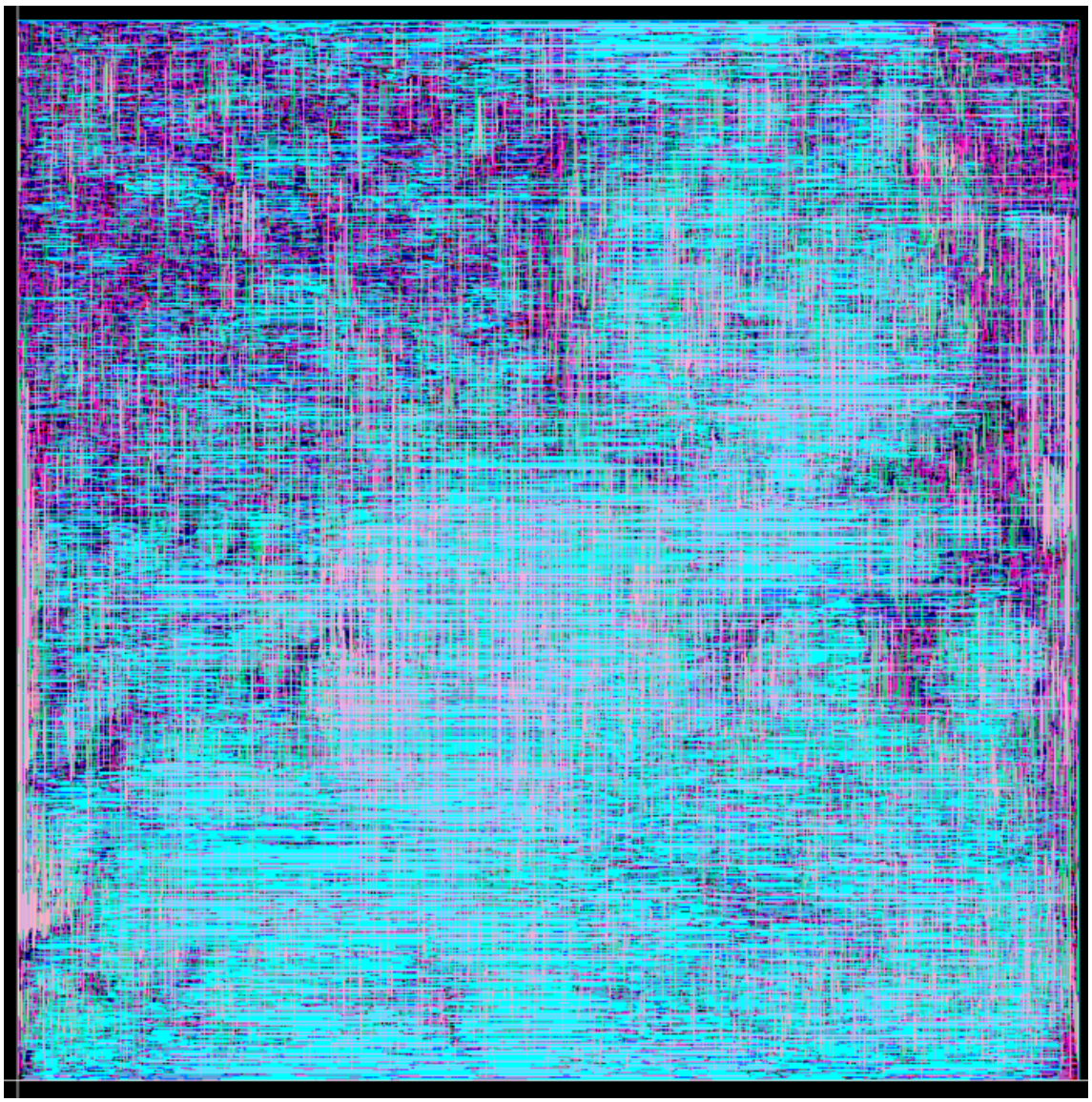

Figure 5.13: Place and Route of the Digital Beamforming Design

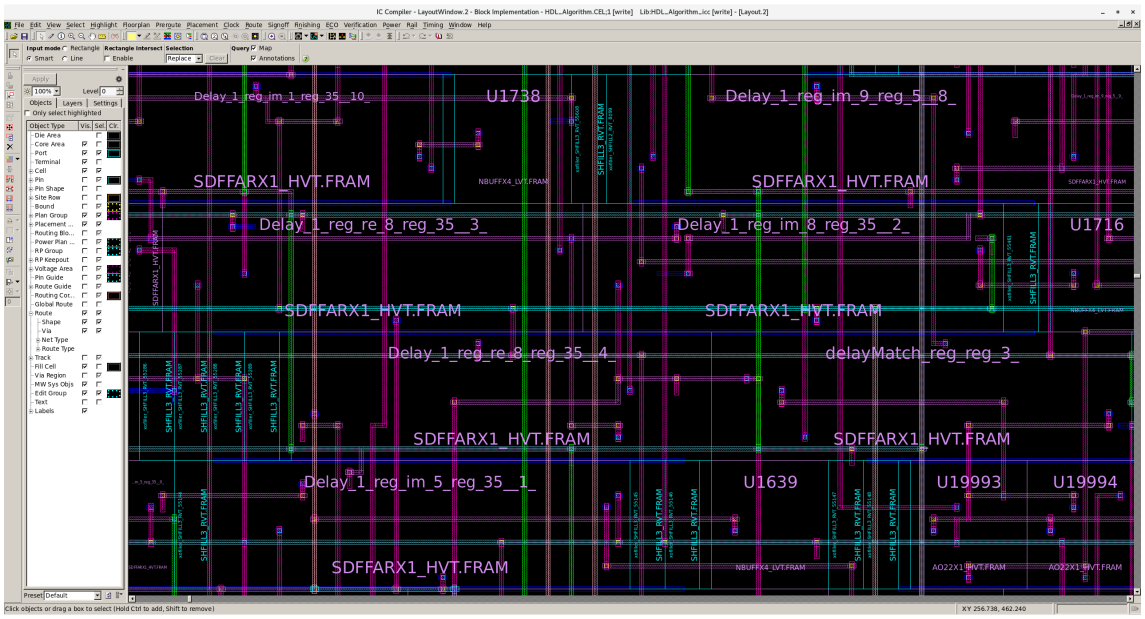

Figure 5.14: Detailed View of the Place and Route Result

Table 5.7: Results From the IC Compiler

Combinational Area:	198144.8937
Noncombinational Area:	234484.1831
Cell Area:	432629.0768
Design Area:	791062.0837

the routing a zoomed in part of the place and route design is shown in Figure 5.14.

The results from running the IC Compiler tool are shown in Table 5.7.

These results show that the total cell area is 432629.0768 nm^2. Whereas, the total design which incorporates power rails is roughly double the cell area at 791062.0837 nm^2. Combinational area refers to combinational circuitry which uses sequential logic or any circuit where the output is dependent on the state of the input signals. The IC Compiler shows results that the combinational and non-combinational area are roughly the same. It is a difficult balance to

incorporate sequential logic since the circuitry can decrease complexity of the design but will increase propagation delay.

Chapter 6

Conclusion

A beamforming algorithm has been studied and implemented in software to simulate incoming signals received by an array of antenna elements. The results from this simulation are shown in Figure 4.1. Using the HDL Coder application in Simulink the HDL Algorithm is generated using HDL synthesizable Simulink blocks. The result of the HDL and behavioral model's beamformed output signal is shown in Figure 4.2. The code generated from HDL Coder has a testbench file that simulated an incoming signal with a specific direction of arrival (DOA) that is used within the algorithm to form a receiving radiation pattern that will maximize the information received. Unfortunately, the co-simulation feature in Simulink did not successfully run with errors shown in Figure 4.4. The algorithm is verified through the Cadence Incisive waveform simulation environment as shown in Figure 4.7. Lastly, after verifying the functionality of the HDL Algorithm the code is synthesized using Cadence DC compiler then applied to different ASIC technology such as the 32, 65, 90 and 180nm technology. The HDL project could not have been implemented on an FPGA board since there was not a board available for the dimensions of this project. The data extracted from the synthesis tool reports are displayed in 5.2 and shows the varying results compared to the technology.

6.1 Future Work

Further experimentation from the work done in this project would involve implementing beamforming using an antenna to transmit a narrowband signal to an array of isotropic antenna elements. The signal information from all the antennas would then be transferred to the FPGA or ASIC platform to generate the beamforming signal results. Unfortunately, the materials required for this experiment were not attainable at the time of this research.

Wireless communication continues to use beamforming techniques that must continually be improved on to meet consumers bandwidth demands. The hardware cost of a digital beamforming model and power consumption is built up of smaller radio frequency chains which is not efficient [34]. Hardware components necessary in a digital beamforming design such as phase shifters are a relatively new technology that is expensive and not supplied at the commercial level. There is future research work done in finding efficient alternatives to costly phase shifters such as switches [15, 24] .

Research done in [35] shows an effective solution to improve the signal-to-noise ratio in radar images by using a hybrid strp-map/spotlight mode. This mode allows multilook processing that enhances the SNR value of the digital beamformed result. Additionally, this method does not require further complexity to increase the SNR.

Reconfigurable intelligent surfaces (RIS) are a novel technology used in communication systems. A RIS consists of 2D array of cells that are movable and can adapt to how a transmit or receive wave is reflected, refracted, absorbed or modulated. RIS platforms use phase shifting with multiple antennas to optimize the channel information. Phase shifting and beamforming is difficult to express within the cascaded channels however, the research published in [36] uses joint phase shift and beamforming that will account for individual channel's information received by the reconfigurable intelligent surface.

References

[1] P. K. Bailleul, "A New Era in Elemental Digital Beamforming for Spaceborne Communications Phased Arrays," *Proceedings of the IEEE*, vol. 104, no. 3, pp. 623–632, 2016.

[2] X. Xiao and Y. Lu, "Data-Based Model for Wide Nulling Problem in Adaptive Digital Beamforming Antenna Array," *IEEE Antennas and Wireless Propagation Letters*, vol. 18, no. 11, pp. 2249–2253, 2019.

[3] G. Malamal and M. R. Panicker, "VLSI architectures for Delay Multiply and Sum Beamforming in Ultrasound Medical Imaging," in *2020 International Conference on Signal Processing and Communications (SPCOM)*, 2020, pp. 1–5.

[4] H. L. V. Trees, *Optimum array processing: Part IV of detection, estimation and modulation theory*. John Wiley and Sons, 2002.

[5] S. H. Talisa, K. W. O'Haver, T. M. Comberiate, M. D. Sharp, and O. F. Somerlock, "Benefits of Digital Phased Array Radars," *Proceedings of the IEEE*, vol. 104, no. 3, pp. 530–543, 2016.

[6] C. A. Balanis, *Antenna Theory Analysis and Design*. John Wiley and Sons, Inc., 2005.

[7] F. Sohrabi and W. Yu, "Hybrid Digital and Analog Beamforming Design for Large-Scale

Antenna Arrays," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 3, pp. 501–513, 2016.

[8] D. B. Casas, *Digital beamforming implementation on an FPGA platform*, 2007.

[9] W.-H. Ma, K.-Y. Chang, K.-T. Chen, Y.-T. Hwang, and J.-F. Lin, "Projection Matching Pursuit based DoA Estimation Scheme and its FPGA Implementation," in *2019 International SoC Design Conference (ISOCC)*, 2019, pp. 109–110.

[10] A. Chinatto, C. Junqueira, and J. M. T. Romano, "Low cost smart antenna array hardware implementation," in *2011 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC 2011)*, 2011, pp. 784–788.

[11] T. Roy, D. Meena, and L. G. M. Prakasam, "FPGA based Digital Beam Forming for Radars," in *2009 IEEE Radar Conference*, 2009, pp. 1–5.

[12] J. Zhang, W. Wu, and D.-G. Fang, "Single RF Channel Digital Beamforming Multibeam Antenna Array Based on Time Sequence Phase Weighting," *IEEE Antennas and Wireless Propagation Letters*, vol. 10, pp. 514–516, 2011.

[13] R. N. Biswas, A. Saha, S. K. Mitra, and M. K. Naskar, "Realization of adaptive beamforming in smart antennas on a reconfigurable architecture," in *2018 Emerging Trends in Electronic Devices and Computational Techniques (EDCT)*, 2018, pp. 1–7.

[14] A. Hamza and H. Attia, "Fast Beam Steering and Null Placement in an Adaptive Circular Antenna Array," *IEEE Antennas and Wireless Propagation Letters*, vol. 19, no. 9, pp. 1561–1565, 2020.

[15] J. Zhang, X. Yu, and K. B. Letaief, "Hybrid Beamforming for 5G and Beyond Millimeter-

Wave Systems: A Holistic View," *IEEE Open Journal of the Communications Society*, vol. 1, pp. 77–91, 2020.

[16] S. Lavdas, P. K. Gkonis, Z. Zinonos, P. Trakadas, and L. Sarakis, "An Adaptive Hybrid Beamforming Approach for 5G-MIMO mmWave Wireless Cellular Networks," *IEEE Access*, vol. 9, pp. 127 767–127 778, 2021.

[17] B. K. Imtiyaz Ahmed, D. F. Jabeen, and D. Thangadurai, "Performance Comparison and FPGA Synthesis of MNLMSNCMA Adaptive Beamforming Algorithm," *International Journal of Emerging Technologies in Engineering Research (IJETER)*, 2018.

[18] N. H. Noordin, T. Arslan, B. W. Flynn, A. T. Erdogan, and A. O. El-Rayis, "Single-Port Beamforming Algorithm for 3-Faceted Phased Array Antenna," *IEEE Antennas and Wireless Propagation Letters*, vol. 12, pp. 813–816, 2013.

[19] A. Liu, W. Sheng, and T. Riihonen, "Per-Antenna Self-Interference Cancellation Beamforming Design for Digital Phased Array," *IEEE Signal Processing Letters*, vol. 29, pp. 2442–2446, 2022.

[20] Z. Li, F. Yang, Y. Chen, S.-W. Qu, J. Hu, and S. Yang, "Wideband Receive Beamforming Based on 4-D Antenna Arrays With Postmodulation," *IEEE Antennas and Wireless Propagation Letters*, vol. 21, no. 4, pp. 740–744, 2022.

[21] G. Ni, Y. Song, J. Chen, C. He, and R. Jin, "Single-Channel LCMV-Based Adaptive Beamforming With Time-Modulated Array," *IEEE Antennas and Wireless Propagation Letters*, vol. 19, no. 11, pp. 1881–1885, 2020.

[22] W. Ren, J. Deng, and X. Cheng, "MMSE Hybrid Beamforming for Multi-User Millimeter Wave MIMO Systems," *IEEE Communications Letters*, vol. 27, no. 12, pp. 3389–3393, 2023.

[23] T. Liang, Y. Pan, and Y. Dong, "Compact Multimode Beamforming Antenna for Space-Division Multiple Access Application," *IEEE Antennas and Wireless Propagation Letters*, vol. 22, no. 8, pp. 1907–1911, 2023.

[24] G. Zang, L. Hu, F. Yang, L. Ding, and H. Liu, "Partially-Connected Hybrid Beamforming for Multi-User Massive MIMO Systems," *IEEE Access*, vol. 8, pp. 215 287–215 298, 2020.

[25] M. Fosberry and M. Livadaru, "Digital Synthetic Receive Beamforming with the Xilinx ZC1275 Evaluation Board," in *2019 IEEE International Symposium on Phased Array System and Technology (PAST)*, 2019, pp. 1–2.

[26] B. Lam, M. Price, and A. P. Chandrakasan, "An ASIC for Energy-Scalable, Low-Power Digital Ultrasound Beamforming," in *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2016, pp. 57–62.

[27] "FPGA-Based Beamforming in Simulink: Algorithm Design." [Online]. Available: https://www.mathworks.com/help/phased/ug/design-an-hdl-beamforming-algorithm-in-simulink.html

[28] "Guided Code Generation." [Online]. Available: mathworks.com/help/hdlcoder/hdl-workflow-advisor.html

[29] "FPGA-Based Beamforming in Simulink: Code Generation." [Online]. Available: https://www.mathworks.com/help/phased/ug/hdl-code-generation-and-verification-of-a-beamforming-algorithm-in-simulink.html

[30] *7 Series FPGAs Data Sheet: Overview (DS180)*.

[31] *DC Ultra Concurrent Timing, Area, Power and Test Optimization Datasheet*.

[32] *PrimeTime Golden TimingSignoff Solution and Environment*.

[33] *Power Compiler User Guide*.

[34] D. C. Gaydos, P. Nayeri, and R. L. Haupt, "Adaptive Beamforming With Software-Defined-Radio Arrays," *IEEE Access*, vol. 10, pp. 11 669–11 678, 2022.

[35] Z. Chen, Y. Zhou, J. Qiu, W. Wang, Z. Zhang, and R. Wang, "A Novel Approach to Further Enhancing SNR in Digital Beamforming SAR Utilizing Hybrid Strip-Map/Spotlight Mode," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[36] K. Li, C. Huang, Y. Gong, and G. Chen, "Double Deep Learning for Joint Phase-Shift and Beamforming Based on Cascaded Channels in RIS-Assisted MIMO Networks," *IEEE Wireless Communications Letters*, vol. 12, no. 4, pp. 659–663, 2023.