UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

REAL-TIME NARROWBAND AND WIDEBAND BEAMFORMING TECHNIQUES FOR FULLY-DIGITAL RF ARRAYS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

DANIEL GARRY THOMPSON

Norman, Oklahoma

2017

REAL-TIME NARROWBAND AND WIDEBAND
BEAMFORMING TECHNIQUES
FOR FULLY-DIGITAL RF ARRAYS

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Mark Yeary, Chair

Dr. Caleb Fulton

Dr. Nathan Goodman

Dr. Jon Bredeson

Dr. John Antonio

Table of Contents

List of Tables

List of Figures

Abstract

Elemental digital beamforming offers increased flexibility for multi-function radio frequency (RF) systems supporting radar and communications applications. As fully digital arrays, components, and subsystems are becoming more affordable in the military and commercial industries, analog components such as phase shifters, filters, and mixers have begun to be replaced by digital circuits which presents efficiency challenges in power constrained scenarios.

Furthermore, multi-function radar and communications systems are exploiting the multiple simultaneous beam capability provided by digital at every element beamforming. Along with further increasing data samples rates and increasing instantaneous bandwidths (IBW), real time processing in the digital domain has become a challenge due to the amount of data produced and processed in current systems. These arrays generate hundreds of gigabits per second of data throughput or more which is costly to send off-chip to an adjunct processor fundamentally limiting the overall performance of an RF array system.

In this dissertation, digital filtering techniques and architectures are described which calibrate and beamform both narrowband and wideband RF arrays on receive. The techniques are shown to optimize one or many parameters of the digital transceiver system to improve the overall system

efficiency. Digitally beamforming in the beamspace is shown to further increase the processing efficiency of an adaptive system compared to state of the art frequency domain approaches by minimizing major processing bottlenecks of generating adaptive filter coefficients. The techniques discussed are compared and contrasted across different hardware processor modules including field-programmable gate arrays (FPGAs), graphical processing units (GPUs), and central processing units (CPUs).

Chapter 1

Introduction

Radar systems with high-speed analog-to-digital and digital-to-analog converters (ADCs/DACs) behind each element are enabling a new class of broadband software defined radar and communications systems spanning from S-band to X-band. Such systems are preferred over traditionally analog dominated radar systems in that digital components allow multiple simultaneous beams, software reconfigurability, and tighter systems integration. To support a multi-function RF capability, scalable RF transceiver systems have begun to support a diverse set of applications including automotive [1], [2], 5G [3]–[8] industries, synthetic aperture radar (SAR) [9]–[13], space [14]–[17], sonar [18], [19], and ultrasound [20].

As sampling rates, instantaneous bandwidths, and number of channels increase, so does the amount of data to process in a digital beamformer (DBF). This limits the performance of DBF due to two bottlenecks that arise. The first bottleneck is the physical aspect of the hardware, i.e. how fast and how many cores? These properties vary across processing nodes depending on transistor density and power profile. The second bottleneck is the input/output (IO) speed and efficiency. All the data to be processed at any instance in time may not reside in the memory space of a single device which requires costly data

transfers between processors. These bottlenecks limit the processing efficiency of a digital beamformer and the exact boundaries vary according to processing architecture.

1.1 Digital Beamforming Architectures

There are many possible instantiations of state-of-the art digital beamforming processing architectures that share several common characteristics. An

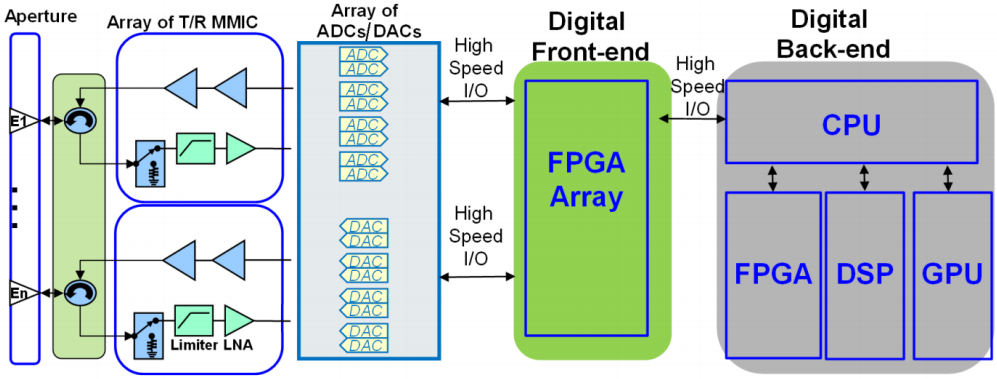

Figure 1.1: Typical block diagram of a fully digital at every element radar system. Three main hardware components make up the array. First the analog RF front end comprises of the antenna array, filters, amplifiers, mixers, etc. and conditions the RF signal to be digitized by the digital front end. The digital front end consists of ADCs and DACs and some high throughput digital operations such as beamforming and calibration. The digital back end follows the digitization process where application specific processing is done in a downstream digital processor. Reprinted from Thompson et al. (2017) © 2017 IEEE.

example of a representative multi-function radar and communications system architecture is shown in Fig. 1.1 whereby data is received and transmitted through an array of antenna elements. Analog power and signal conditioning are applied through transmit/receive (TR) modules, and the signals are digitized through an array of ADCs and DACs. High speed IO on the order

of 100s of gigabits per second (Gbps) is used to transmit this data to the "front-end" FPGA array. In some cases the FPGA array is on the same module as the ADCs [21], [22], and in other architectures, an optical IO may be used for transfers longer than many feet of distance. The FPGA array is typically used for data processing/compression algorithms such as digital down conversion (DDC), decimation and resampling to smaller bandwidths, pulse compression, demodulation, matched filtering, and beamforming. These algorithms are further discussed in Chapters 2-4. Finally, the data is sent to the digital "back-end" where algorithms typically requiring less latency or more memory are implemented. Note in this diagram, a direct connection from the front-end to the back-end is shown only through the CPU, which is typical of today's architectures and will be a key discussion point in Chapter 5.

1.1.1 Beamforming in an Array

The traditional approach of beamforming has been studied extensively [23]–[26] and such systems have been previously demonstrated [27]–[29]. The beam shape in terms of angle θ is dependent on frequency, ω, as seen by the following beampattern equation.

$$E(\omega, \theta) = \sum_{m=1}^{M} exp(-j(\phi_m + \frac{\omega m d}{c} \sin\theta)) \tag{1.1}$$

Where m is an integer between 1 and M array elements, d is the half wavelength element separation based on the smallest wavelength (i.e. highest frequency component), c is the speed of light, and ϕ_m is a phase shift applied at element m. A rule of thumb to prevent beamsquint in wideband arrays

is to keep the instantaneous or operation bandwidth to 1% of the carrier frequency and has been shown in [25].

$$B << \frac{c}{L \sin \theta_0} \tag{1.2}$$

Where B is the instantaneous system bandwidth, L is the length of the array, related to the operating frequency such that $L = M\frac{\lambda}{2}$, and θ_0 is the steering angle of the main beam. Commonly, phase shifting has been utilized for beamforming in narrowband arrays. In a wideband system, however, the phase shift approximation is not adequate due to beamsquint. Given certain number of elements M, general element spacing d, waveform bandwidth B, and the waveform duration T, beamwidth of a timed array and a phased array can be derived. In the narrowband analysis, beamwidth is solely determined by the number of elements in the array, M, assuming an element spacing matched to the particular frequency. The beamwidth can then be estimated by $\theta_{3dB} = 102/M$ [23] at broadside.

1.1.2 Frequency Dependent Beamsquint

Bandlimited signals associated with digitized radar waveforms can be classified either as narrowband or wideband. The ratio of bandwidth to carrier frequency defines which category a signal falls into. The distinction between each type is evident while beamforming as beamsquint becomes problematic. Beam patterns of the corresponding narrowband and wideband beamformers, are shown in Fig. 1.2 and Fig. 1.3. Array factor patterns are steered to -25 degrees for a 10 element array using a.) phase shifting and b.) true time delay (TTD) beamforming. Patterns are shown for frequencies centered at 10 GHz

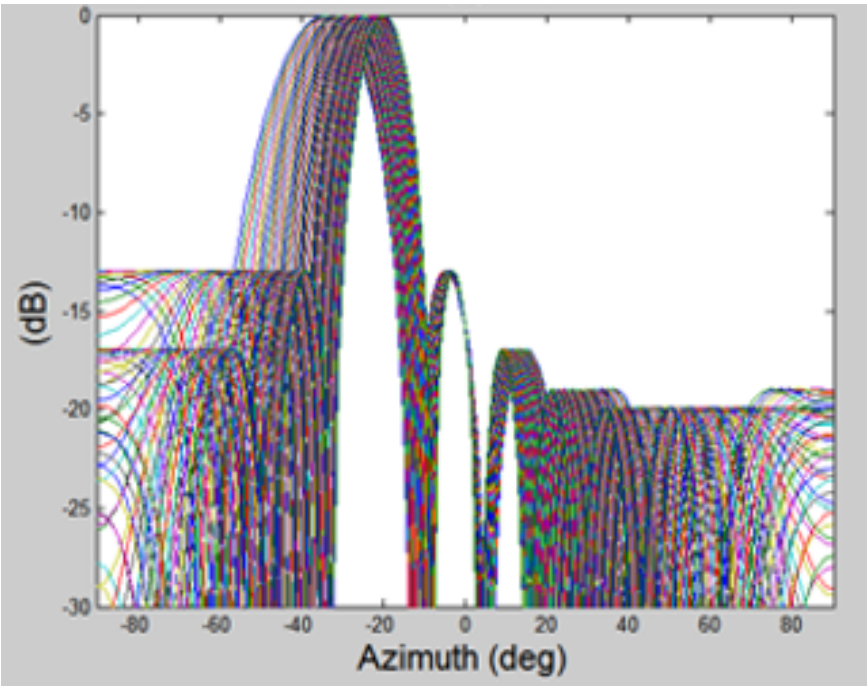

Figure 1.2: Beam patterns of individual frequencies across a bandwidth with phase shift beamforming to -25 degrees off boresight. The main beam pointing direction and the main beam width increase as the beam steers off boresight.

with a bandwidth of 5 GHz..

$$E_{narrowband}(\omega,\theta) = \frac{1}{M}\sum_{m=1}^{M} |exp(i\frac{\omega md}{c}(\sin\theta - \frac{\omega_0}{\omega}\sin\theta_0)))| \tag{1.3}$$

$$E_{wideband}(\omega,\theta) = \frac{1}{M}\sum_{m=1}^{M} |exp(i\frac{\omega md}{c}(\sin\theta - \sin\theta_0)))| \tag{1.4}$$

Where ω_0 is the angular frequency at the center of the band and θ_0 is the beam steering direction. Classification is determined by the ratio of bandwidth to carrier frequency In the narrowband case, steering off boresight with wide operation bandwidths results in the beam steering off the intended direction (i.e."beamsquint") [30]. Time delay beamforming can overcome the beam squint limitation by adding some complexity to the narrowband beamformer.

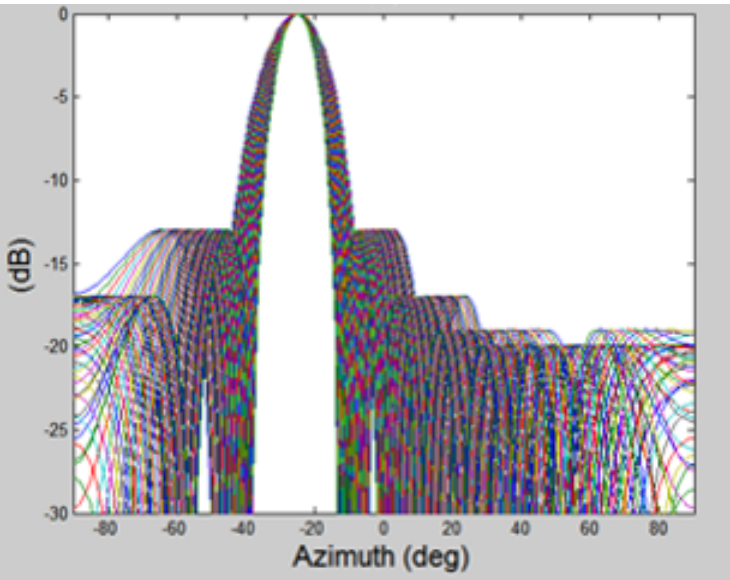

Figure 1.3: Beam patterns of individual frequencies across a bandwidth with TTD beamforming to -25 degrees off boresight. The main beam pointing direction is aligned at all frequencies.

Typically, the added complexity is in the form of a filter in the digital domain. Alternatively, the narrowband implementation can also be generalized as a simplified wideband filter with a single tap.

1.2 DBF Processing Hardware

The next generation of communications (i.e. 5G) and the automotive industry are driving the need for wideband digital processing techniques. These applications are pushing the center frequencies of digital transceivers into the milimieter wave regime [3], [4], [7], [8] but are limited to high frequency applications. At these high frequencies, beamforming is crucial to overcome the free space propagation loss. To limit power consumption and increase efficiency, tightly integrated packages in SiGe have been developed [1], [2]. The SiGe process is beneficial for high transistor switching speeds;

however, the analog circuits depend on wavelength and are not optimized in terms of die area, limiting the total number of channels and beams that can be processed on a single transceiver module. Today, monolithic microwave integrated circuits (MMICs) built in III-V semiconductors and other heterogeneous technologies [31], [32] are becoming more common. However, recent advances in silicon have pushed modern CMOS processes to a point of similar performance compared to SiGe [5], [6], [33]–[35], enabling RF transceivers to be built in the digital domain with mature fabrication processes. These advances include tightly integrated transceiver modules such as in [36], [37] are enabling better power efficiency in the digital processing than ever before.

In a digital processor, the basic element of a beamformer is the digital filter; an example finite impulse response (FIR) filter is shown in Fig. 1.4. This dissertation studies four main types of real time digital beamforming processors that support such filter structures:

- ASICs
- FPGAs
- GPUs
- CPUs

Each processor is an application specific integrated circuit (ASIC) in itself and wafer run fabrication costs can be high for custom designs usually attributing to long turn around times and challenging first pass attempts in yielding a successful design. An ASIC layout of a single multiply and accumulate operation is shown in Fig. 1.5. The corresponding register

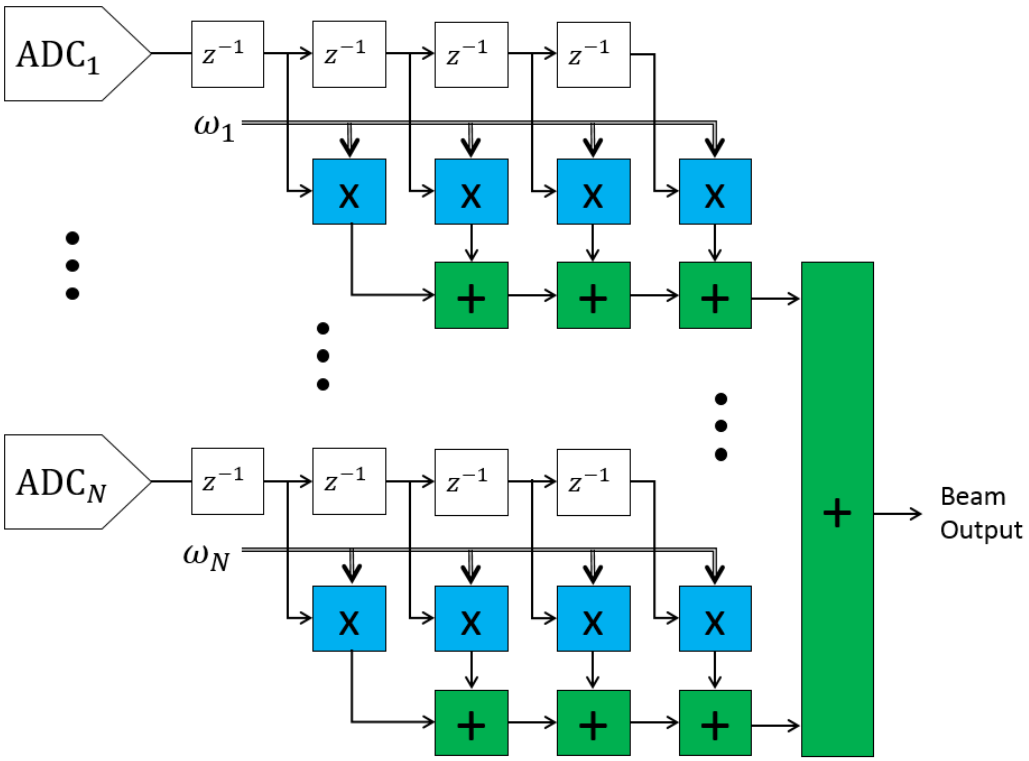

Figure 1.4: Functional block diagram of a digital FIR filter in hardware using delays, multiply-accumulate, and sum functions. The beamforming weights, w_m are real or complex , depending on the transceiver architecture (i.e. complex baseband vs. real IF, or direct RF conversion). The length N of each filter across M elements determines the supported bandwdith (i.e. either narrowband or wideband). In adaptive beamforming applications, ω_m is updated in time.

transfer level (RTL) diagram is shown in Fig. 1.6. Many instantiations of the multiply and accumulate (MAC) operation have been demonstrated [38]–[44] demonstrating that the fundamental purpose of the MAC is consistent across all implementations, i.e. the building block of the digital filter. ASICs are typically more efficient that reconfigurable devices while supporting faster clock speeds [45], however, modern digital processors offer access to hard multipliers on chip. The processors listed utilize this common processing

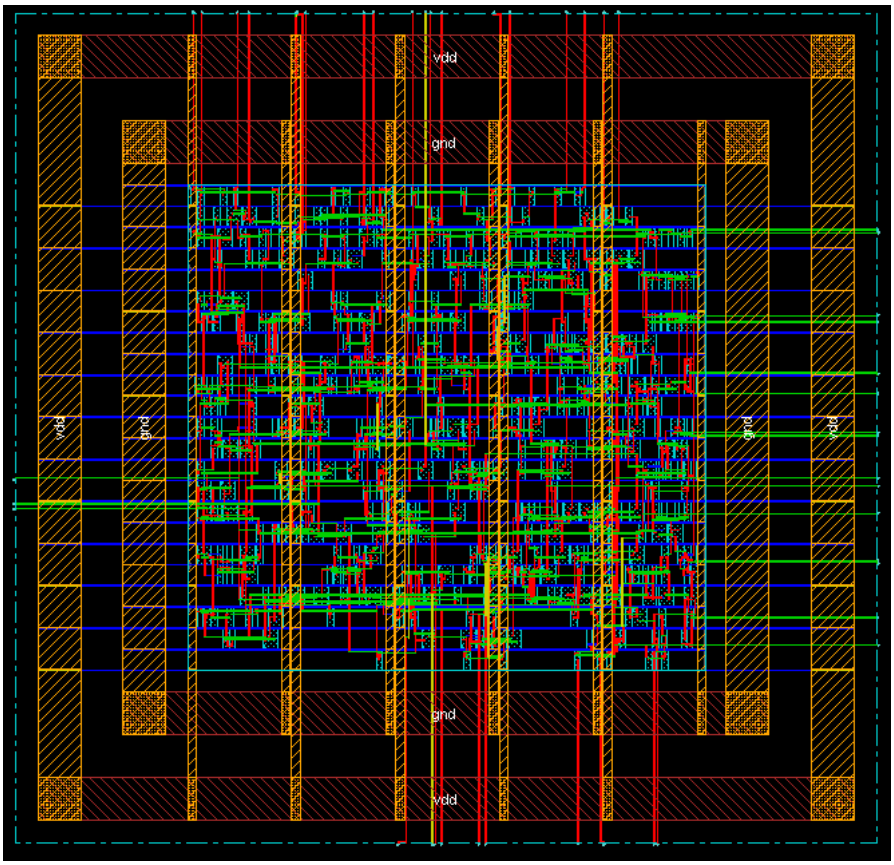

Figure 1.5: Routed standard cell ASIC layout of a multiply-accumulate function in 45 nm CMOS. Modern ASICs and FPGAs have thousands of MAC resources configured in a pipelined array.

function and offer arrays of MACs in different configurations, architectures, and speeds. A majority of the digital beamforming functionality discussed in this dissertation focuses on the FPGA hardware due to the reconfigurability and high throughput capability.

Advances in commercial-of-the-shelf (COTS) FPGA technology are enabling reconfigurable digital beamforming techniques in an embedded device operating in real time for the first time. The two leading FPGA manufacturers, Xilinx and Intel (formerly Altera), have begun to provide commercial access to 3-dimensional complementary metal oxide semiconductor

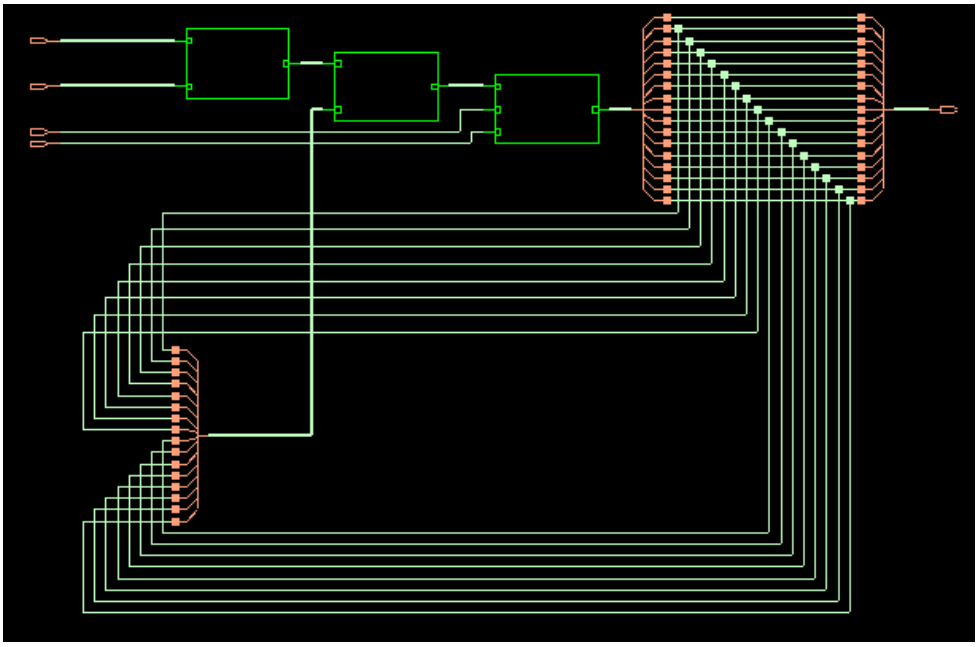

Figure 1.6: RTL of the multiply-accumulate function. This dataflow diagram shows a simplistic model of a digital MAC. Only 3 main circuits are required: multiply, add, and a register. Abstraction of the digital architecture becomes important for routing and timing analysis of digital processors.

(CMOS) transistors beginning with the 20 nm process node from Taiwan Semiconductor Manufacturing Company (TSMC). Hundreds of millions such transistors make up the fundamental building blocks required to perform thousands of simultaneous digital multiply and accumulate operations on a single device. The power efficient to cost ratio of COTS FPGAs has eliminated the need for digital ASICs, which have traditionally been designated to process real time data. In this dissertation, practical digital beamforming techniques for modern radar and communications applications operating on a digital processor in real time is discussed.

Modern FPGAs have improved the high speed transceiver data transfers by implementing hard cores in the silicon of the chip opposed to software defined interfaces which require longer latencies. The 3 main high-speed protocols

employed today in FPGAs are PCI Express (PCIe), multi-Gigabit Ethernet, and high-speed memory interfaces, such as DDR3. These silicon defined interfaces are well suited for device communication over long distances or backplanes, however, these interfaces are plagued with high overhead costs and relatively slow speeds compared to more optimized chip-to-chip protocols such as Interlaken implemented in Altera and Xilinx UltraScale FPGAs. Although multi-gigabit speeds can be achieve with the Interlaken protocol, interface latency in the serial transmitter and receiver ultimately inhibits the data communication speed.

The increasing processing power and interface speed in FPGAs is enabling the front-end radar functions, such as equalization and beamforming, to be performed in the digital domain [46]. Modern FPGA devices have greatly improved the practical IO limitations and are capable of processing throughputs up to hundreds of Gbps which are required to process wideband beams for reusable array applications [35], [47]–[50]. The Altera Arria 10 device is one of the first FPGA technology nodes providing the size, weight, power, and processing efficiencies suitable to interface to an analog RF front-end, pushing the digital domain closer to the aperture.

1.3 Outline of the Dissertation

As digital components continue to replace analog circuits in RF systems, the processing efficiency of the digital circuits is a major contributor to the performance and number of beams a single device can synthesize simultaneously. Chapter 2 discusses an optimization technique that reduces the number of bits required in phase shift beamforming, effectively increasing

the number of simultaneous beams that can be computed on a single embedded device. Processing wide band waveforms introduces an additional challenge in that the data rate is much higher that in the narrowband case. Not only are the number of simultaneous wide band beams limited per device, the data rate of a single beam requires a much higher data rate since the beam processing is frequency dependent and can not be solved with typical phase shift beamforming techniques such as in Chapter 2. In Chapter 3, a multiple simultaneous wideband beamforming and equalization technique shown to support real time data rates is discussed. In addition to the fixed coefficient beamforming techniques discussed in Chapters 2 and 3, radar and communications applications typically require the beamforming coefficients to also be updated in real time. Processing timelines considered "real-time" can be associated at the sample, pulse, coherent processing interval (CPI), or video frame rate. It is of particular interest to study coefficient adaptation at the sample level since this provides the highest data rate requirement. In Chapter 4, assumptions of the adaptive problem and a beamspace approach building on the concepts in Chapters 2 and 3 when traditionally a scalable approach for larger and larger arrays becomes a challenge. Chapter 5 benchmarks common DBF algorithms such as filtering and the fast Fourier transform (FFT) across FPGA, GPU, and CPU processing hardware platforms. Finally, source code to perform wideband beamforming and matrix inversion are included in the appendices. Appendix A includes the MATLAB code listing that is used to generate wideband calibrate and beamforming coefficients. This code was useful in creating the beamforming figures shown in Chapter 3. Appendix B includes the Simulink block diagram that has been designed to invert matrices with low latency.

Chapter 2

Narrowband Digital Beamforming Techniques and Experiments

Narrowband beamforming is a critical component in many S, C, and X-band radar systems. In narrowband systems, IO efficiencies are improved if decimation sub-systems appear closer to the digital front end [51]. This chapter presents an efficient real time architecture of a 2 channel digital transceiver. Limiting the IO of the narrowband beamformer was also studied. Array hardware was configured with bit-limited phase shifters and shown to perform equivocally to a higher precision configuration, saving 5 bits of information per datastream.

2.1 Narrowband Beamforming Digital Hardware

The Atmospheric Radar Research Laboratory (ARRC) of the University of Oklahoma (OU) is building a teamwork-oriented, cylindrical, dual-pol, phased array radar, and this section addresses the hardware development of the waveform generation and digital receiver portions of the digital transceiver. Direct digital synthesizers (DDS) are being utilized to generate digital waveforms for the radar. DDSs will allow for smooth communication

between array nodes and efficient beamsteering in this application so long as a synchronization technique ensures the operation of an accurate master clock. A synchronous technique for generating waveforms over multiple channels of the radar and the architecture of each channel's transmitter is described. The functions, utilization, and synchronization scheme of DDSs in this application are also discussed. Finally, a digital receiver solution is explored. Combining these two ideas introduces a low-cost, custom digital transceiver with a small form factor. This transceiver has been designed and built at the ARRC and utilizes waveform generators and digital receivers to be used in multi-channel radar platforms. The printed circuit board (PCB) is shown in Fig. 2.1.

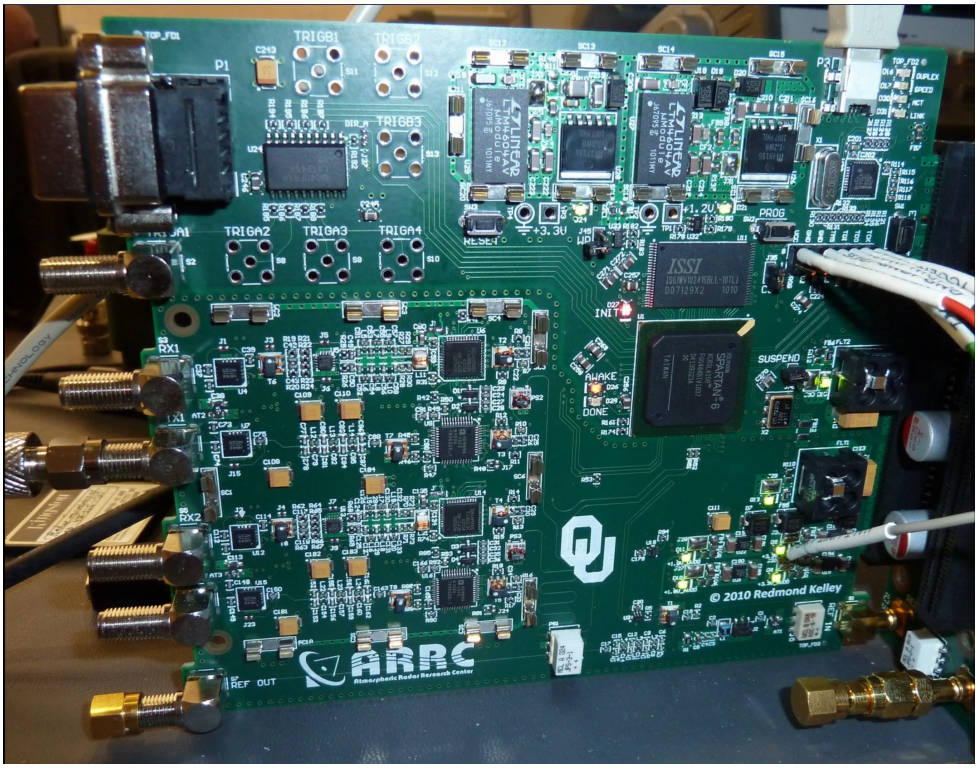

Figure 2.1: Digital 2 channel transceiver PCB designed at the University of Oklahoma. Supporting 50 MHz IBW on receive and DDS waveform generation on transmit. Reprinted from Thompson et al. (2011) © 2011 IEEE.

Fig. 2.2 shows a simplified transmit side of each transceiver unit. The clock routing of the system is carefully organized such that the clocks entering all FPGAs synchronized and in phase with each other. Similarly, reference clocks from the FPGA to each DDS are devised to be in phase with each other. Careful planning matches the clocks entering and leaving the DDSs, but transmit triggers for each DDS throughout the array need to be generated synchronously as well. Additional timing mismatches are alleviated in the calibration procedure.

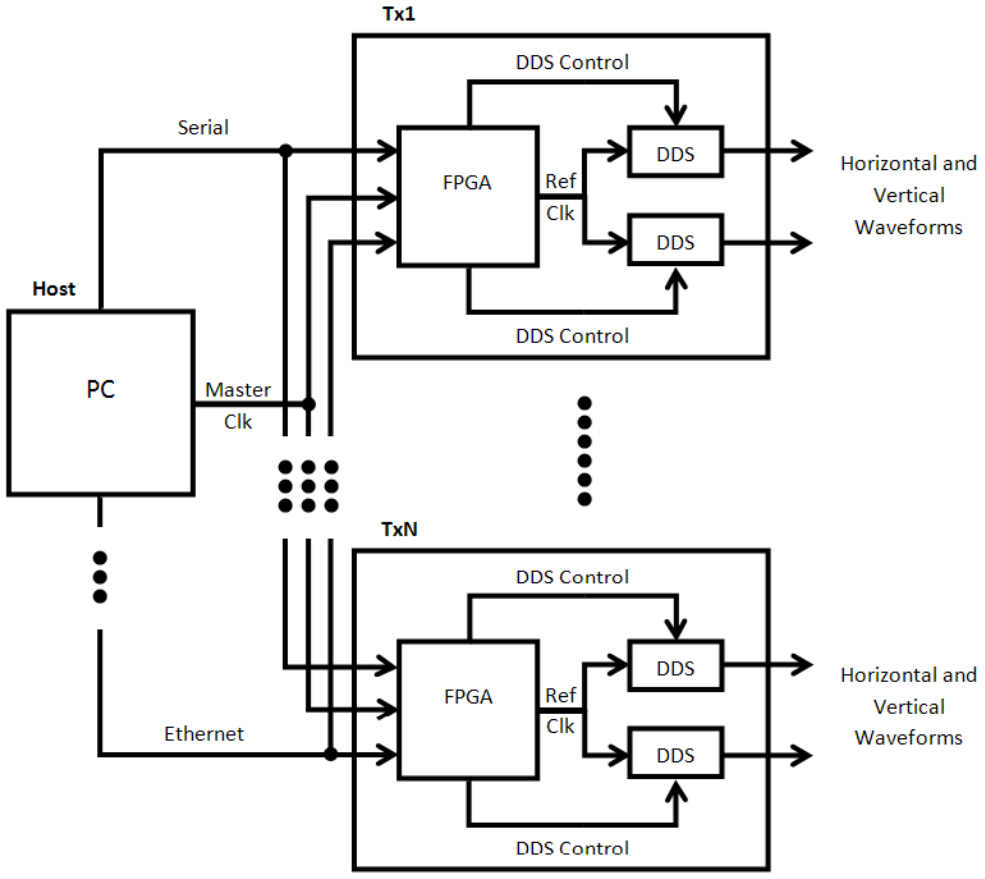

Figure 2.2: Digital transmitter block diagram showing and array of 2 channel TX modules capable of supporting horizontal or vertical polarization channels. Reprinted from Thompson et al. (2011) © 2011 IEEE.

The AD9954 DDS presents an effortless interface to create frequency sweeps. A start, stop, and step frequency are programmed into the device. After a trigger is initiated, the DDS generates the chirp waveform. This technique allows for lower pulse power while improving range resolution. The time series data of the chirped waveform is shown in Fig. 2.3 The transmitter

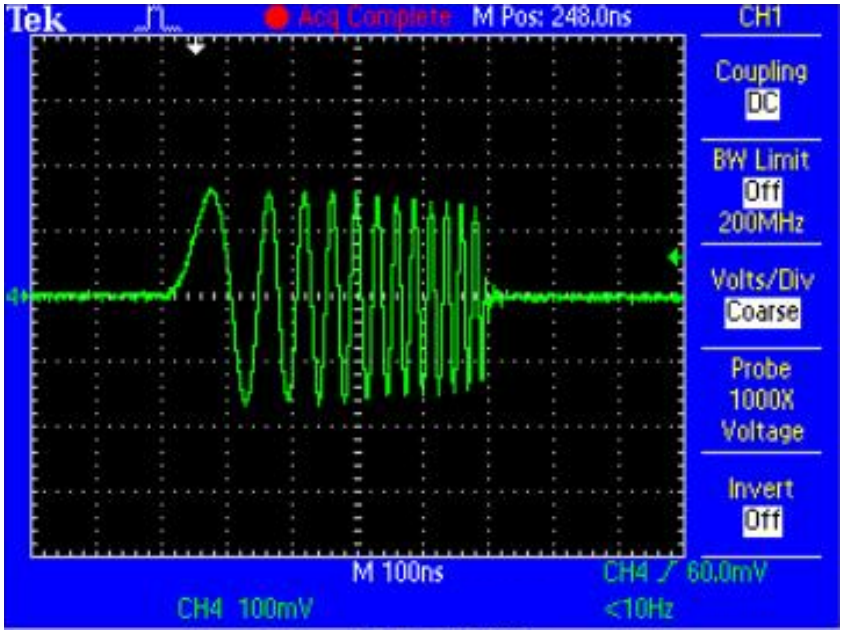

Figure 2.3: DDS generated waveform 0.5 μs pulse with 1 MHz to 50 MHz frequency sweep. Reprinted from Thompson et al. (2011) © 2011 IEEE.

includes two DDS subsystems and an FPGA controller. A host computer provides a master clock to all nodes and each node serially interfaces with the host. Additionally, the host can communicate with each node over an Ethernet connection using a standard Trivial File Transfer Protocol (TFTP) interface. The FPGA is configured with a Microblaze subsystem which interfaces the host and DDS's to the FPGA. Beamforming functions are also implemented on the FPGA, discussed later in this chapter. The FPGA is configured with a Microblaze soft microcontroller subsystem which interfaces

the host and DDSs. The physical design of system is carefully organized such that the clocks entering all FPGA's are in phase with each other. Similarly, reference clocks from the FPGA to each DDS are devised to be in phase with each other. Careful planning matches the clocks entering and leaving the DDS's, but transmit triggers for each DDS throughout the array need to be generated synchronously as well. Thus, a simple synchronization technique was devised.

An array of digitizers, referenced from a common distributed clock source, imposes careful timing considerations on a fully digital array. In each digitizer, the source clock latches the analog voltage of the detection circuit onto a capacitor which is then registered by the digital circuitry. The copper traces routing the clock to each digitizer in the array cannot be expected to have equally matched lengths throughout the array and hence, the latches will open and close asynchronously. Ideally, all of the sample-and-hold (S-H) circuits in the entire array would open and close at the exact same time. In reality, a distributed reference clock for a sizable array will have delays in the clock distributing lines, on the order of hundreds of nanoseconds. This makes it necessary to implement delay estimation techniques in order to achieve sub-nanosecond clock accuracy. To address this problem, modern digitizers have the ability to correct these offsets by calibrating internal timing parameters. This is typically achieved with fine phase control of a second sampling clock internally derived from the common data clock routed to each digitizer. This synchronization functionality can be implemented using internally generated reference clocks and a master/slave configuration between each ADC in the array. The technique identifies a single master and many slaves where each slave device requires an additional input reference

clock generated from the master. The slave devices use the master reference to tune the phase of their internally generated sampling clock using an adjustable analog delay creating a feedback loop that can be used to identify the phase offset between the master and slave. The drawback of this and similar techniques is that the additional clocks be routed throughout the system. An all-digital array can circumvent this requirement by precisely measuring the phase of the individual array channels available from the raw in-phase/quadrature (I/Q) generated in the digital receiver. After the relative phase has been measured in each channel, the analog phase controller of the digitizer can then be tuned to align the digitizers using the control offered by the FPGA without requiring additional system hardware modifications.

Furthermore, synchronization among transmit nodes is essential for correct operation of the phased array. Each FPGA has a counter driven by the master clock and generates waveforms at specific counter values. In order for every DDS to output waveforms synchronously at these values, an initialization of the system is required. Firstly, the serial interface relays a synchronization pattern from the FPGAs to the host. The host then computes how far each node's data is off from one another and sends an Ethernet packet to each node containing the number of clock cycles to delay required to align the data. This process aligns the FPGAs' serial communications with the host. Next, each FPGA serially relays their counter value to the host. Finally, through the Ethernet connections, the host relays individual offsets to nodes in order to synchronize the values of the FPGA counters. FPGA output reference clocks are aligned and DDS triggers will happen synchronously at this point. Commands issued by the host will inform the transmitters which counter value to issue the waveform generation trigger to the DDS. All DDS

output waveforms will be generated synchronously once that count is reached.

Following DDC, the I/Q data for each channel is serialized and sent to a host computer. The host computer includes a digital acquisition (DAQ) PCIe-6537 from National Instruments card that is capable of serially acquiring 32 total channels all synchronized to one clock running less than 50 MHz. In order to simplify hardware implementation, this sample clock was divided by a factor of two in the FPGA from the ADC's sample clock of 80 MHz, yielding 40 MHz. 16 bit I/Q pairs, forming 32 bit signals, are generated for each receive channel. The FPGA serializes the data and outputs differential signals for each channel along with a 40 MHz clock. These signals come out of the FPGA to a custom backplane through a PCIe interface. The backplane is able to combine both channels from 16 different transceiver boards and route them through a very-high density cable interconnect (VHDCI) cable to the DAQ. The serial signals are synchronous to the first transceiver board's clock. Finally, LabVIEW acquires the signals and deserializes each channel. The data is now available to be displayed and processed further.

2.2 Narrowband Channel Decimation

Fig. 2.4 shows the basic architecture of a single channel digital receiver. Upon entering the receiver, an amplifier is employed to further amplify the analog signal coming from the previous stage of the radar (typically a RF downconverter). Next, a low pass filter conditions the analog signal for a high speed ADC. The specific ADC being used is Analog Device's AD9265. The ADC samples at a rate of 80 mega samples per second (MSPS). A serial port programs the ADC to enable flexible conversion options. Following the

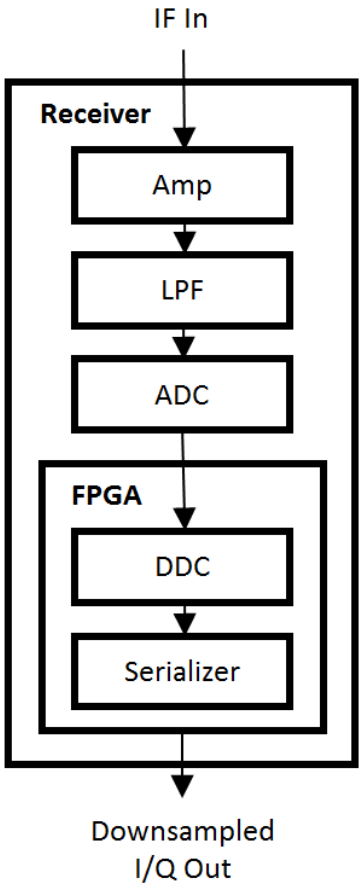

Figure 2.4: Narrowband receiver high level architecture. Reprinted from Thompson et al. (2011) © 2011 IEEE.

ADC, the digital signal enters the FPGA to be digitally processed. The FPGA implements a DDC to extract the frequency band of interest. All processing blocks are implemented on the FPGA's hardware and are fully modular. Fig. 2.5 shows a block diagram of this process for both channels of the transceiver board. First, I/Q demodulation is achieved through dedicated hardware multipliers. Samples of a 10 MHz sinusoidal signal at specific intervals allows this signal to be digitally stored in the FPGA in a small table of values. This is a simple alternative to having dedicated hardware to

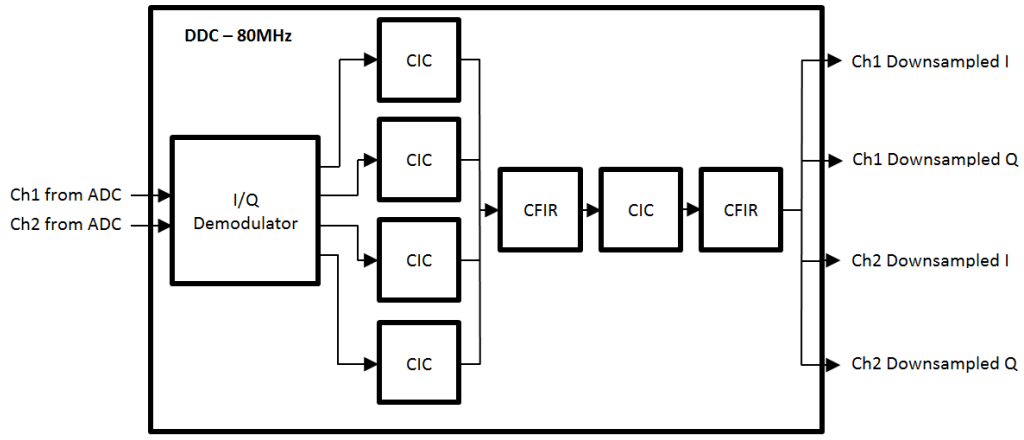

Figure 2.5: Digital down converter architecture showing 2 stages of decimation. After demodulating to complex baseband, the I/Q streams are filtered with a multiplierless CIC filter where the datastream is concurrently decimated. A low order FIR filter follows that corrects the amplitude droop of the CIC operation. A second stage of decimation follows to further improve the stop and passband characteristics of the overall frequency response of the system. Reprinted from Thompson et al. (2011) © 2011 IEEE.

implement an oscillator for the purpose of demodulation. A 10 MHz reference frequency was chosen because the center frequency of interest is 70 MHz. Incidentally, sampling a 70 MHz signal at a rate of 80 MHz results in a 10 MHz signal due to aliasing.

Furthermore, demodulating by 10 MHz yields baseband signals and higher frequency content which can be easily filtered out in later stages. This can be explained further using the simple trigonometric equation, Equation 2.4.

$$\cos\theta\cos\phi = \frac{\cos(\theta-\phi)+\cos(\theta+\phi)}{2} \tag{2.1}$$

Taking θ as 10 MHz (70 MHz aliased) and ϕ as 10 MHz, it can be seen from the equation that 0 MHz and 20 MHz signals are produced. This technique yields a baseband signal of interest while the 20 MHz signal can be removed

by a low pass filter (LPF).

The next stage of the DDC downsamples the baseband signals I and Q to be fed to the host computer for display and further signal processing. This stage's specifications are motivated by the serialization of the final output. Down conversion is needed to take a 16-bit input rate of 80 MSPS down to a serial output of 1.25 MSPS. This is achieved by a series of linear, digital filters. Next, a cascaded integrator-comb (CIC) filter running at 80 MHz decimates the input signal by 8. CIC filters are of importance due to the absence of multipliers implemented in the FPGA. The CIC filter has a general transfer function of Equation 2.2, a *sinc* function.

$$H(z) = (\frac{1 - z^{-RM}}{1 - z^{-1}})^N \tag{2.2}$$

The implemented hardware uses $R = 8$ as the decimation ratio, $M = 1$ as the number of samples per stage, and N as the number of stages in the filter. As N increases, the roll-off of the filter's frequency response increases as well. This value of N was chosen as 6. The main purpose of this filter is to decrease the sample rate to 10 MSPS, while removing out-of-band signals which would alias. Unfortunately, the CIC's filter response rolls off immediately and has a narrow usable passband. This is the reason the CIC is followed by a compensation finite impulse response (CFIR) filter. This filter inversely matches the amplitude response of the CIC in the passband to normalize the gain in this region. Equation 2.3 shows the amplitude response of this filter for the passband under study (which is an inverse *sinc* function).

$$G(\omega) = \left| sinc^{-1}\left(M\omega\right) \right|^N \tag{2.3}$$

At this stage, another decimation stage of the CIC and CFIR combination. Again, the next CIC decimates the input signal by a factor of 8. This takes the total decimation factor to 64. Similarly to the previous CFIR, the next stage compensates for the CIC's "drooped" amplitude response. A decimation rate of 64 is needed due to the 32 bits of I and Q (16 each) with an input frequency of 80 MHz matched to a serial output of a 40 MHz frequency (details will be described later). Fig. 2.6 and Fig. 2.7 shows the MATLAB simulation of the overall frequency response of these 4 filters cascaded together. Both figures show identical magnitude responses for the filter chain, however, Fig. 2.7 presents a detailed view of the passband and transition frequencies. The legend shows the colors representing each of the four individual filter responses, the cascaded response, and the acceptable noise floor. It is interesting to note in Fig. 2.6, that the filter removes all frequency content outside our region of interest, the stopband. This leads to an acceptable signal bandwidth (denoted Sig BW in the figures) of nearly the Nyquist sampling rate of 40 MHz. Fig. 2.7 shows the cutoff frequency, F_c, as 0.63 MHz, a passband ripple (ripple) of 0.19 dB, and a transition frequency bandwidth, F_{trans} of 0.632 MHz. These are all acceptable values for testing the prototype transceiver board. In order to realize these filters in hardware, filter coefficients are taken from MATLAB and implemented into Xilinx IP cores on the FPGA. The initial variables used were chosen for rapid prototyping purposes. More specific specifications will be investigated in the future. Nevertheless, these coefficients are fully reloadable and can be updated by the transceiver's Ethernet interface. Reloadable, in other words, allows the filter characteristics to be easily changed "on-the-fly" instead of applying hardware changes to the FPGA in order to accommodate different transmit pulses. The implementation of this chain of

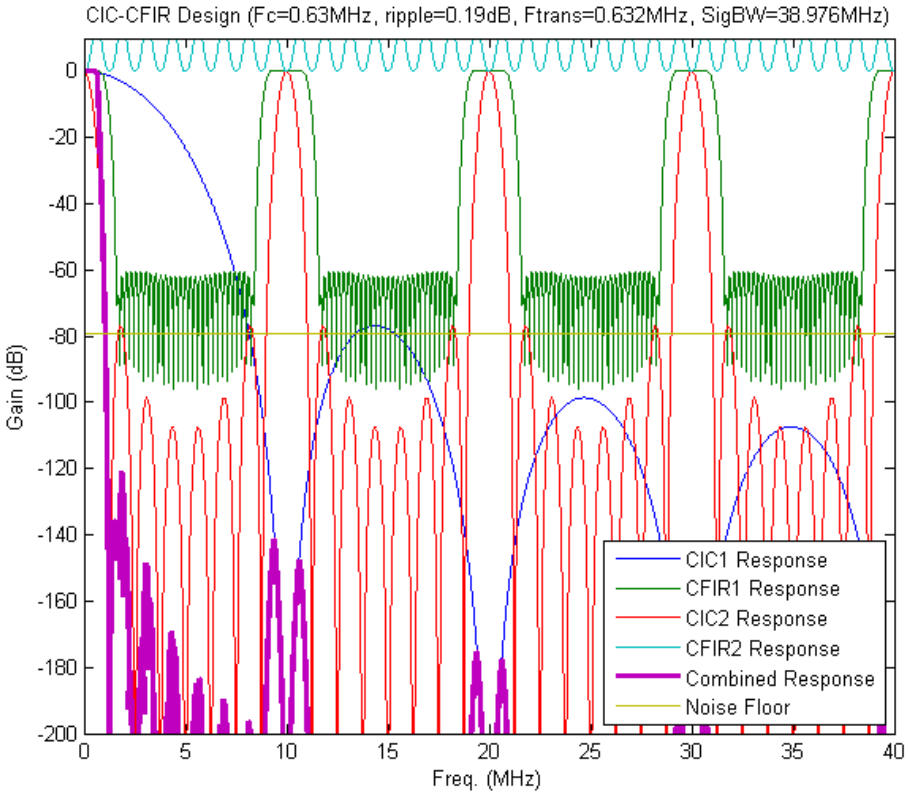

Figure 2.6: Digital receiver downconversion filter cascaded magnitude response (per channel): Full bandwidth. The combined response exhibits a tight rolloff with low sidelobes and stopband performance that once decimated produces an efficient digital filter. Reprinted from Thompson et al. (2011) © 2011 IEEE.

filters is demonstrated by the block diagram of the previous figure, Fig 2.5. The figure shows the architecture for both channels of the transceiver board. The first stage requires four separate CIC filters for the I and Q samples of two channels. The following stages can be combined into a single dataflow. This is because the filter rate after the initial decimation is divided by a factor of eight. This means that the following filters can accommodate up to eight channels. Multiplexing the four channels on the board to the eight empty filter channels allows the FPGA to reuse valuable logic (mainly multipliers used in

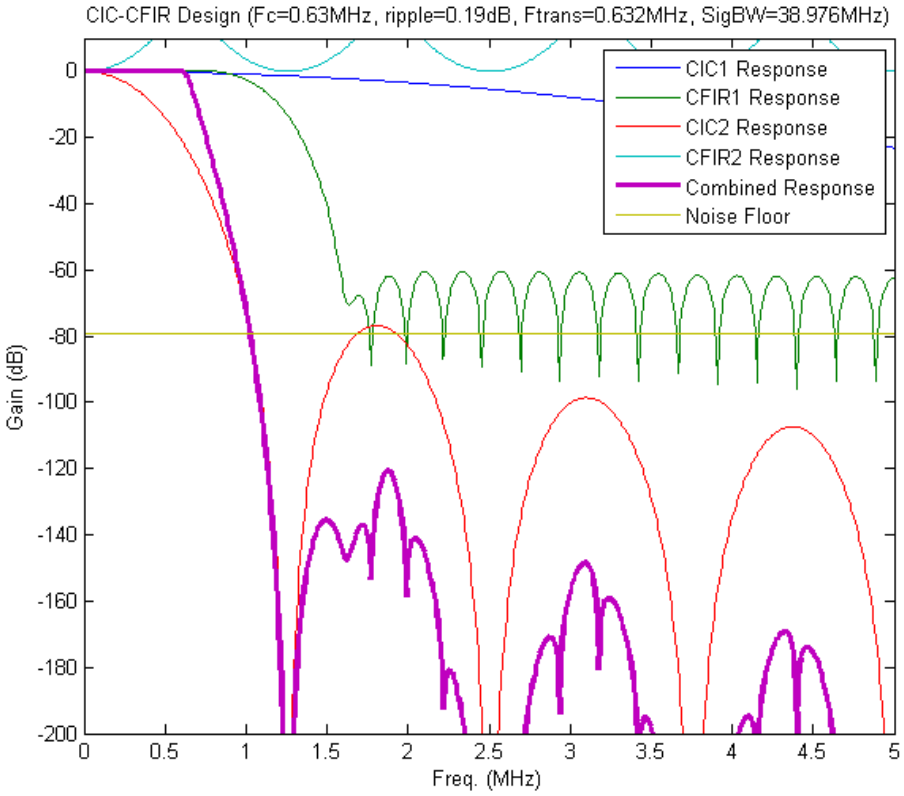

Figure 2.7: Digital receiver downconversion filter cascaded magnitude response (per channel): Detailed Bandwidth. The cutoff frequencies of the compensation filters are chosen so that the first sidelobe response is suppressed below 120 dB. Reprinted from Thompson et al. (2011) © 2011 IEEE.

the FIR filters).

2.3 Phase Shift Beamforming

In addition to minimizing data rates with frequency decimation, narrowband digital systems use beamforming to further decreases data rates by decimating in the spatial domain. In this section, a new paradigm of assigning bits to a set of digital phase shifters is proposed. Modern day digital phase shifters associated with each TR module typically relies on four to six bits. Each

of these phase shifters provide a desired resolution of 22.5 degrees to 5.625 degrees, respectively. When an array is steered in one of the canonical angles of the phase shifters, the error between the desired angle and the actual angle is essentially zero. For example, steering 11.25 degrees with the family of six bit phase shifters would be ideal. However, when an array is requested to steer in a non-canonical direction, say 15 degrees, then errors will occur. These errors occur because each of the phase shifters receive the same bit assignments. In other words, the shifters operate independently and improvements will occur if the set of shifters operate collectively. In particular this section seeks the solution for this problem: for a set of M phase shifters, each defined by B bits, an optimal approach could be derived to distribute the $M \times B$ bits so that the array performance is maximized. In order to achieve this, the simulated annealing algorithm is used to search for a nonlinear phase gradient solution that optimizes the maximum sidelobe level performance given a certain beamsteering angle.

Approaches for optimum bit assignments have been applied for digital filter design with success, yet optimal approaches for assignments to digital phase shifters has largely been overlooked. These include computationally efficient decimation filter design [42], [52]–[54]. For example, based on the team's previous research, it has been shown that searches may be described as "adaptively pruned tree searches" [53] or "greedy" [54]. In addition, the so-named Multiple Constant Multiplication (MCM) problem, first introduced by Potkonjak et al. [43], [44], has been rigorously studied over the last two decades. [38]–[41]. These MCM approaches are tangential to the direct phase-shifter problem at hand, but lessons can be learned from them. This proposed approach may have direct uses for radars that require a high degree

of beam pointing accuracy, given a limited number of bits in each phase shifter in each radar's array, such as [9], [11], [12], [14], [16].

Motivation for a computationally efficient search algorithm is a consequence of the phase assignment controller complexity. The controller, typically a FPGA or microcontroller, can have high computational and power cost, especially for non-integer calculations. As a result, an extended search algorithm is unfeasible in embedded systems. Optimization techniques exist that take advantage of Monte Carlo methods when deterministic algorithms are unfeasible. One such method is the Metropolis–Hastings algorithm, known as simulated annealing (SA) [55], [56]. The computational cost of this algorithm can be controlled by limiting the number of search iterations and thus is a good algorithm candidate for the scenario of finding globally optimal phase assignments on board an embedded processor.

Digital phase shifter values are assigned in software. Some work has been done in an attempt to manipulate the array pattern through software [57]–[60]. These references digitally manipulate phase shifter values and amplitude weights in order to control main lobes, sidelobes, null angles, etc. Additionally, phased array patterns have been studied in order to improve other array properties such as resolution [61]. However, instead of adaptive nulling and other techniques, this approach emphasizes correspondence between phase elements to achieve better array performance.

Optimization of a phased array's performance while beam steering can be defined in multiple ways. For one such metric, this section focuses on improving maximum sidelobe levels and discusses the effects on other metrics such as root mean squared (RMS) sidelobe level, beamsteering angle error, and main beam width. These properties of a phased array antenna can be

derived from the array factor (AF), which is a model of an array's electric field pattern in the far field. Given an array of identical antenna elements, the AF, E, becomes a summation of each individual element's AF and is defined as

$$E(\theta) = \sum_{m=1}^{M} w_m e^{jkd_m a_\theta} \quad . \tag{2.4}$$

Where m is the element number with respect to a reference element, with the first entry, $m = 0$, w_m is an amplitude weighting function, k is the wave vector, $(\frac{2m}{\lambda})$, d_m is the position of the mth array element, and a_θ is the beam steering angle vector. In this study, w_m is taken as the rectangular window, i.e. all elements have equal weights. Analysis has also been performed with different windows (Hamming, Hanning, etc) with similar results. The phase angles ϕ_m needed to steer a beam in the direction of θ is defined in Equation 2.2.

2.4 Phase Quantization Effects on Beamforming Performance

Since digital phase shifters possess a finite number of control bits, valid phase values are bound by B. This constrains valid phase shifter values, ϕ_m to finite steps within 360 degrees.

$$\phi_n = \frac{360}{2^B} M \tag{2.5}$$

Where M is the integer value that approximates ϕ_m the closest. Table 2.1 shows valid phase increments available for B bits. Quantization of phase control has been shown to affect antenna beam performance (beamsteering angle error, increased sidelobe levels, and quantization lobes, in particular) and much work has been done in an attempt to solve these problems [13],

Table 2.1: Phase shifter resolution in degrees for different quantized bit lengths.

Number of Phase Shifter Bits (B)	Angle Offsets
3	45°
4	22.5°
5	11.25°
6	5.625°
7	2.8125°
8	1.4063°

[62]–[66]. By modifying the error function of the quantized phase gradient, a new phase front can be developed to create nonlinearities that can be shown to remove the periodic property of the quantized phase front and therefore mitigate quantization lobes. The theoretical phase front is a reflection about the center of the array. It will be shown that the maximum sidelobe levels can be reduced by modifying this phase gradient, thereby removing the symmetric property. Disrupting this attribute will also increase sidelobe levels for angles far away from the steering angle. Nonetheless, this allows a possible phase gradient solution that sacrifices higher average sidelobe levels for lower maximum sidelobe levels.

A method to search a subspace of all valid phase offset combinations has been developed.

$$\phi_m = kd_m \sin\theta \tag{2.6}$$

For a linearly oriented, one dimensional array with $\lambda/2$ element spacing, the phase offsets simplify to become

$$\phi_m = \pi(m-1)\sin\theta \tag{2.7}$$

It can be seen that the phase offsets create a linear function with slope of

$m \sin \theta$. This relation is referred to as the "phase gradient" or "phase front". The phase front results in a set of phase offsets across the array in which the values would be assigned to a real phased array system's set of digital phase shifters. Given a phase shifter's finite degrees of freedom (i.e. bits), the phase front will be quantized and errors will ensue when phase offsets are rounded to the nearest acceptable value. The comparison of ideal phase gradients and quantized phase gradients are shown in Fig. 2.8.

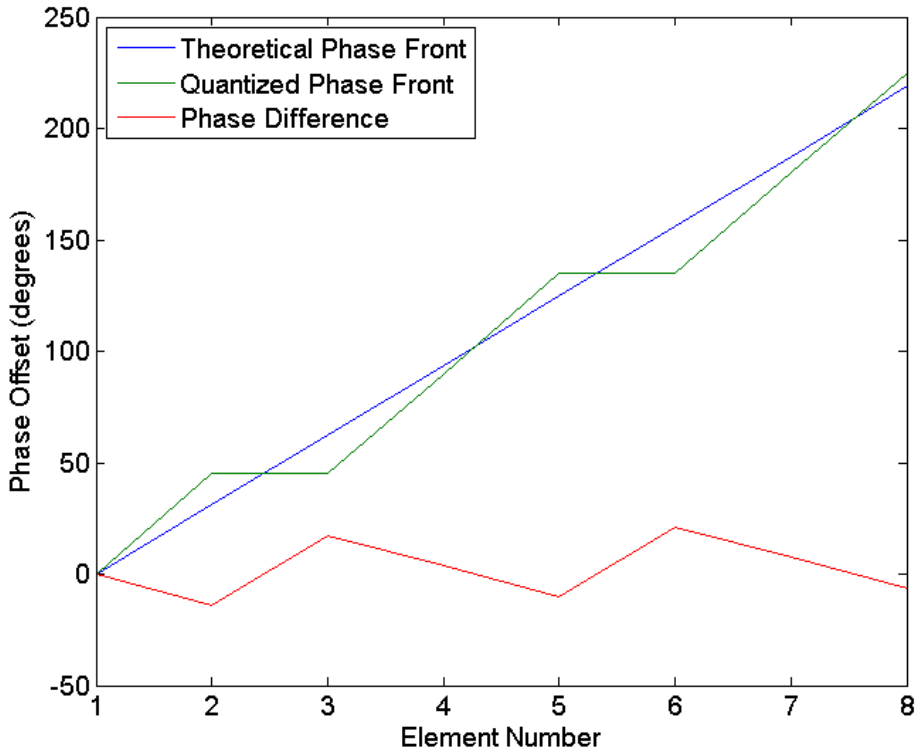

Figure 2.8: Quantization effects on a phase gradient steered to 10 degrees off broadside of an 8-element phased array with 3-bit phase shifters. The quantization error is shown in red. Reprinted from Thompson et al. (2015) © 2015 IEEE.

2.4.1 Efficient Quantized Phase Shift Beamforming

By modifying the error function of the quantized phase gradient, a new phase front can be developed to create nonlinearities that can be shown to remove the periodic property of the quantized phase front and therefore mitigate quantization lobes. The theoretical phase front is a reflection about the center of the array. It will be shown that maximum sidelobe levels can be reduced by modifying this phase gradient, thereby removing the symmetric property. Disrupting this attribute will also increase sidelobe levels for angles far away from the steering angle. Nonetheless, this allows a possible phase gradient solution that sacrifices higher average sidelobe levels for lower maximum sidelobe levels. In order to test this approach, a method has been developed that searches the space of all valid phase offset combinations.

2.4.2 Using Simulated Annealing to Search for Psuedo-Optimal Phase Offsets

A method to search for a set of quantized phase shifter values that improves the overall AF is presented in the section. The SA optimization algorithm has been studied extensively [55], [56]. The algorithm is used here to search for a nonlinear phase gradient solution that optimizes the maximum sidelobe level performance given a certain beamsteering angle. Fig. 2.9 depicts a flowchart of how the algorithm incorporates SA into the optimization problem. Initially, the array is steered to an ideal configuration. As the temperature is decreased, the beamsteering coefficients are randomized and the AF is analyzed to see if a valid beam is produced. If so, the temperature is decreased and the next iteration begins, If not, the next phase state is accepted as the best state

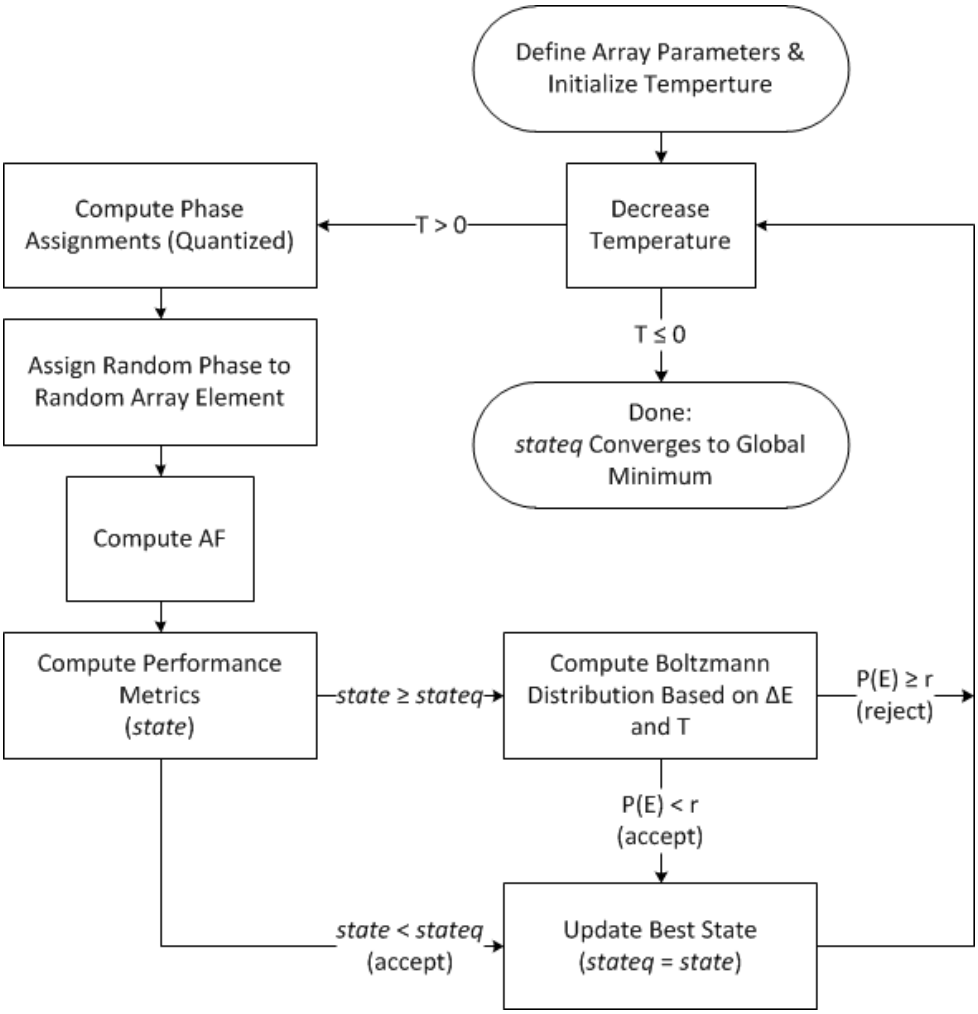

Figure 2.9: Flow chart of the optimization algorithm using simulated annealing. Reprinted from Thompson et al. (2015) © 2015 IEEE.

based on the Boltzmann variable. Once the next best state is determined, the temperature is decreased and the algorithm iterates. It is assumed that the best phase state converges at the end of the temperature schedule which is chosen to be long enough to find at least a local minima.

SA defines a "temperature schedule" i.e. the number of iterations used in the algorithm and a starting temperature. In each iteration, the temperature is "cooled" as it decreases with time. A best state is initialized with the

quantized theoretical phase front. During each iteration, a random element of the current best state is chosen to accept a random phase offset, resulting in the current state. The AF of the current state is then used to find the maximum sidelobe levels of the corresponding phase front. If the sidelobe level found is less than the current state's sidelobe levels, then the current phase gradient is saved as the best state and the algorithm continues searching for more optimal possibilities. If this new state has worse performance, however, the algorithm uses the Botzmann distribution (Equation 2.8) to determine a final criteria on whether or not to save the state as the best state, even though it may perform worse.

$$P(E) = e^{\frac{-\Delta E}{T}} \tag{2.8}$$

In Equation 2.8, $P(E)$ is a test statistic, ΔE is the change in error of the current state compared to the best state, and T is the current temperature. Once $P(E)$ is computed, the value is compared to a uniformly distributed random variable. The state is saved if $P(E) < r$ holds true, where r is the random variable. In this case, a suboptimal phase gradient is allowed to become a psuedo-optimal state. Otherwise, the test is rejected and the algorithm continues without saving the state.

It can be seen from Equation 2.8 that as the temperature is decreased, there is a lower chance of suboptimal states being considered as optimal. In other words, at the beginning of the algorithm (i.e. iterations with higher temperatures), worse performing states have a higher probability of becoming the current best state. This way, the algorithm is able to search through the search space with a better chance of achieving the optimal phase assignment possible without getting caught in a local minimum. Once the temperature

schedule is finished (i.e. a temperature value of zero), the algorithm is complete and the result will converge to the optimal phase front for the tested steering angle. The state space of the convergence over 10K iterations is shown in Fig. 2.10.

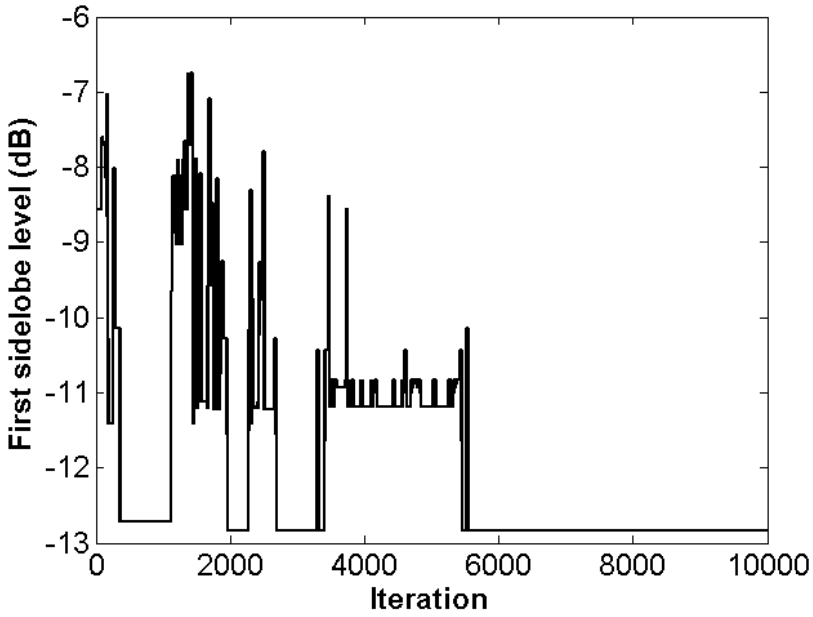

Figure 2.10: Simulated annealing with 3 quantization bits converging over 10000 iterations. The convergence parameter was chosen agressively to test the optimization locality. The SA algorithm is shown to find adequate sidelobes levels after nearly 200 iterations. A forgetability factor is programmed into the algorithm which is seen at iteration 1800 when the SA algorithm finds a -7 dB sidelobes level. As the iterations progress, the forgetability factor decreases and enables the SA algorithm to converge at slightly better sidelobes levels after 5500 iterations. Reprinted from Thompson et al. (2015) © 2015 IEEE.

An array with 12 elements and 3-bit phase shifters has been simulated in Fig. 2.11. The approximated global optimum phase gradient of the array needed to steer the beam to 3.75 degrees off broadside and its resulting AF pattern is shown. The theoretical and modified phase fronts have different phase offsets in the first and sixth elements only. Since the number of elements

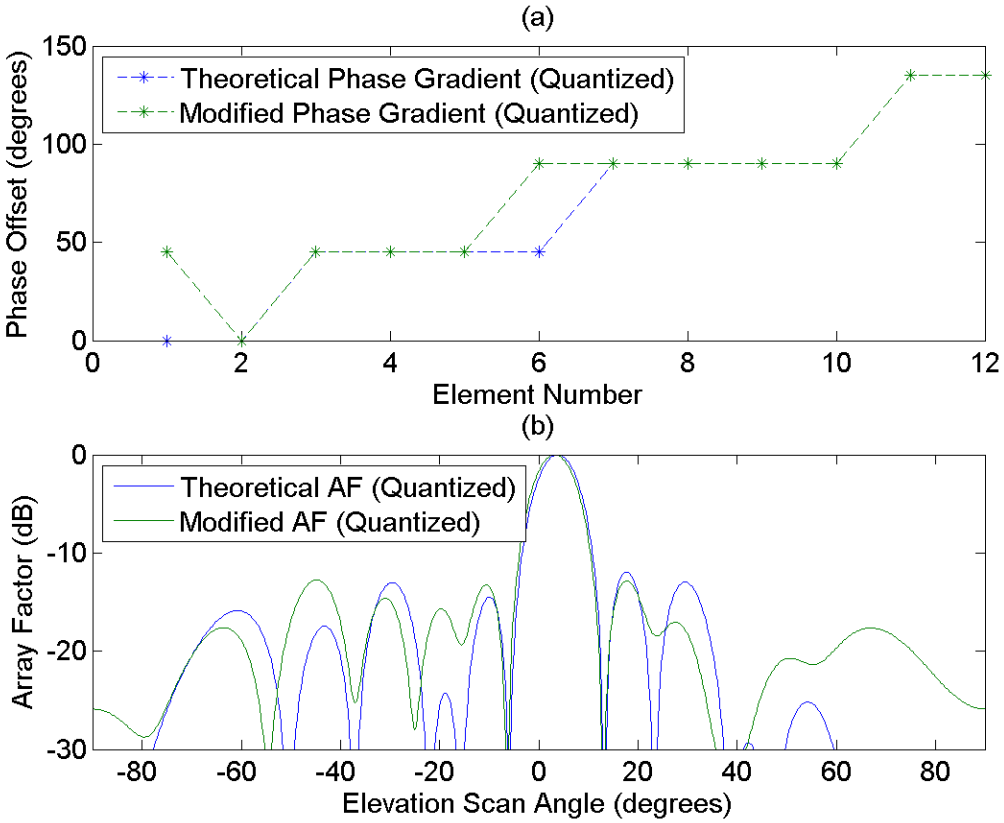

Figure 2.11: In a.) Traditional and optimum quantized phase gradients across a 12-element phased array and in b.) array pattern of corresponding phase gradients. Reprinted from Thompson et al. (2015) © 2015 IEEE.

is identical and the phase fronts are nearly similar, the main beams for both cases have similar beamwidths. Note, the steering angle has shifted slightly due to a change in the phase front's symmetric reference point. Additionally, the sidelobe right of the main beam has been reduced while the sidelobe left of the main beam has been raised. Angles furthest away from broadside increase in power significantly.

Fig. 2.12 demonstrates the sidelobe effects of quantizing phase shifter assignments shown by the AF across azimuth angles from 0 degrees to 45 degrees. Notice the symmetry of the quantization lobes at angles away from

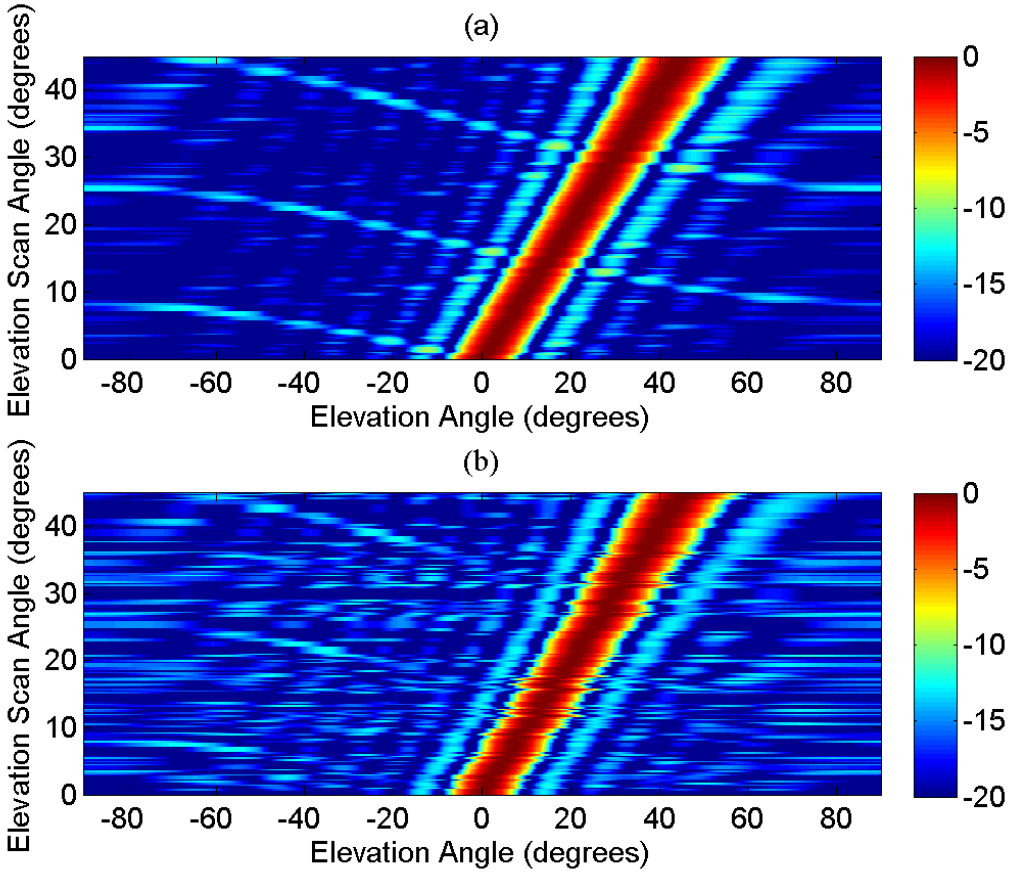

Figure 2.12: Array pattern when steering off broadside for a.) traditional 3-bit quantized phase shifters and b.) optimal phase gradients. Reprinted from Thompson et al. (2015) © 2015 IEEE.

the steering angle before modification. After introducing the optimal phase front, the shape of these quantization lobes is disrupted and sidelobe levels away from the main beam are raised. Also able to be seen is a decrease in power of the sidelobes adjacent to the main beam.

Fig. 2.13 demonstrates the peak sidelobe levels (including improvement) with respect to steering angle. The traditional, quantized phase gradient is shown in the top plot of Fig. 2.13. At many steering angles, the maximum sidelobe level can raise above -10 dB. This is a result of the 3-bit quantization

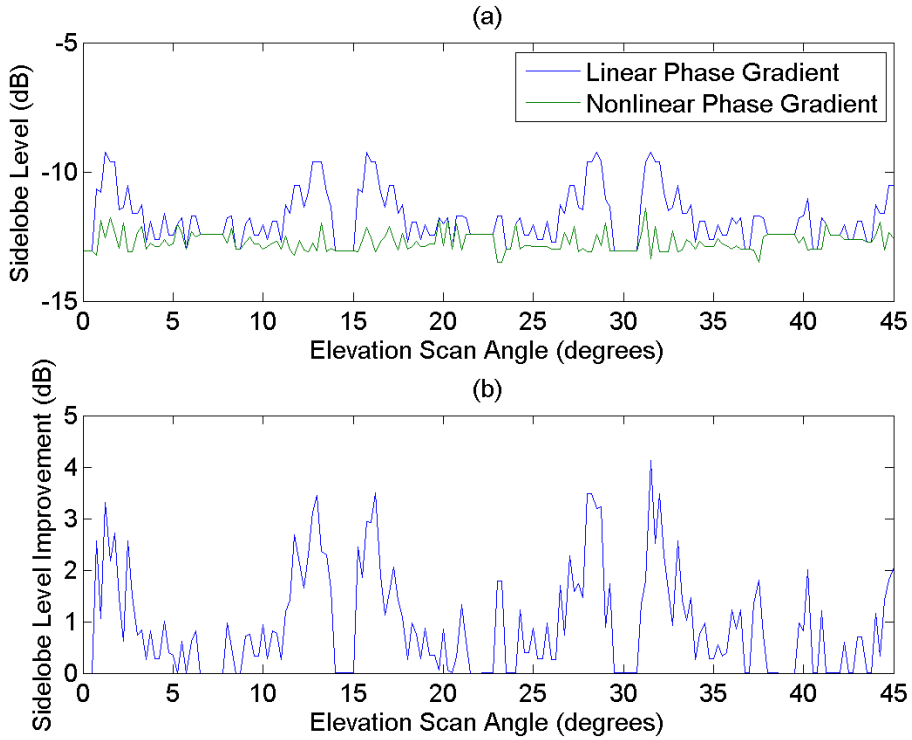

Figure 2.13: Array pattern first sidelobe a.) maximum levels and b.) improvement from trandional quantized phase gradients to optimal phase gradients. Reprinted from Thompson et al. (2015) © 2015 IEEE.

of the phase values. Altering the phase front removes the periodicity of the quantized phase errors, the first sidelobes are lowered by the optimal phase gradient found. The bottom chart of Fig. 2.13 shows the improvement across steering angles from 0 degrees to 45 degrees. Improvements of over 3 dB can be achieved when the traditional method performs poorly at a given angle. The optimal phase gradient searching algorithm is shown to perform well when quantized phase fronts for a given steering angle do not produce maximum sidelobe levels near the theoretical maximum of 13.2 dB down (a result of the rectangular amplitude window function).

Tradeoffs are necessary with better sidelobe levels, however. For each

optimal phase gradient found for a given steering angle, the main beam position was allowed to drift from the intended steering angle. When introducing a nonlinear phase gradient, the average angle tangent to the slope of the gradient (i.e. the steering angle) will not be preserved. The results of this error are shown in Fig. 2.14. It has been shown in [62] that the

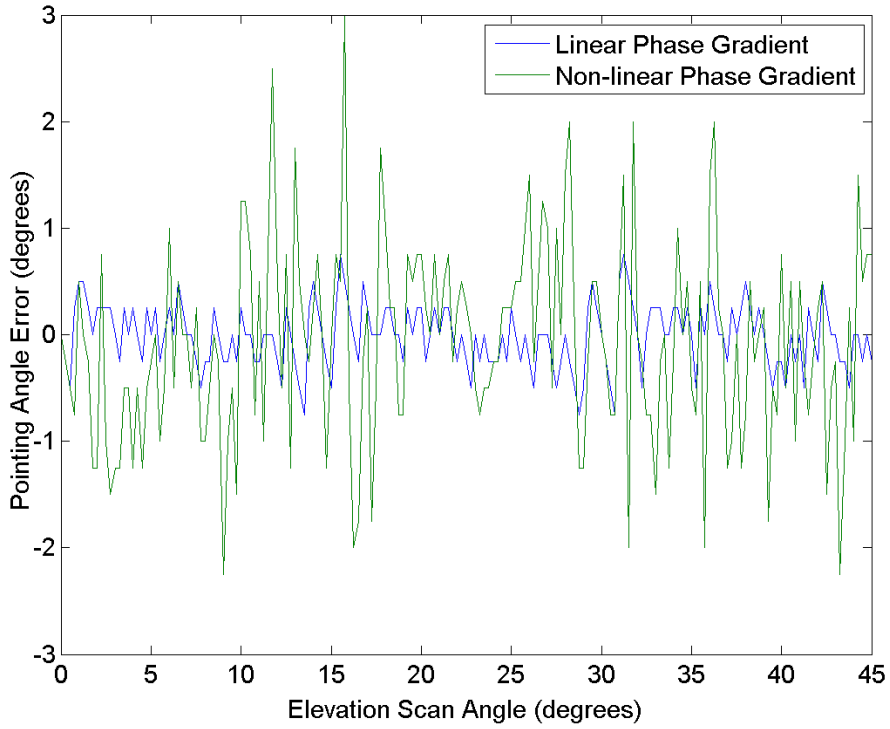

Figure 2.14: Main beam pointing errors due to modified phase gradients for a 12-element phased array. Reprinted from Thompson et al. (2015) © 2015 IEEE.

linear, quantized phase front will incur errors that are dependent on beam steering angle. The pointing errors created with the optimal, non-linear, quantized phase fronts are not dependent on steering angle and are controlled by the randomly introduced phase offsets. Additionally, these errors can be dependent on the number of elements in the array, M. The freedom of

altering bit assignments in a 12 element array is minimal and therefore can produce significant beam steering angle errors. More degrees of freedom are introduced with larger values of M and as a result will produce smaller errors. This undesired effect can be controlled by the SA algorithm, however, and can be done by only considering states with errors below than a defined threshold.

Another property of the AF that is dependent on M is the sidelobe level improvement. This property is shown in Fig. 2.15 as a function of M. Both

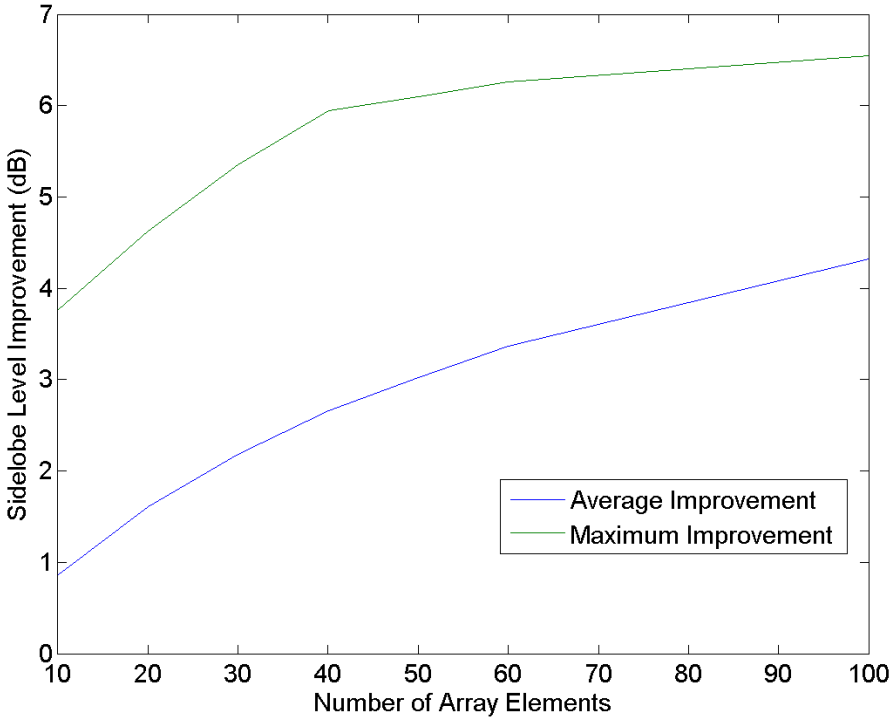

Figure 2.15: First sidelobe level improvement for different phased array sizes. Reprinted from Thompson et al. (2015) © 2015 IEEE.

the average and maximum improvement values are shown for all beam steering angles between 0 degrees and 45 degrees. The algorithm can never introduce a phase front that has higher sidelobes than the traditional quantized example

due to the SA algorithm. Therefore, the approach guarantees at least the same or better performance than previous methods. In the average and maximum cases, it can be seen that the sidelobe improvements converge with high values of M.

2.4.3 Narrowband Experiments with the Army Digital Array Radar

The Army DAR hardware platform [22] was used to verify the narrowband beamforming experiments discussed in this chapter. The narrowband experiments performed with the DAR are shown in Fig. 2.16. Beam patterns were cut using scripts written in Labview to control the positioner angle and update beamsteering coefficients on the DAR. The antenna array consists of a 9x9 patch array, shown in Fig. 2.17. The outermost elements were terminated.

2.5 Chapter Summary

A digital transceiver solution for a new cylindrical, dual-pol, phased array radar has been introduced. The DDS is an integral part in generating waveforms for the transmitter portion of the transceiver. Waveform generation for multiple channels of the array is easily implemented with a low-cost, flexible DDS integrated circuit. A simple synchronization technique throughout the array allows consistent and robust waveforms to be transmitted to multiple channels within the system. Simplicity of the DDS design and implementation allow for easy generation of two signals per transceiver allowing for dual-pol waveform transmissions. Additionally,

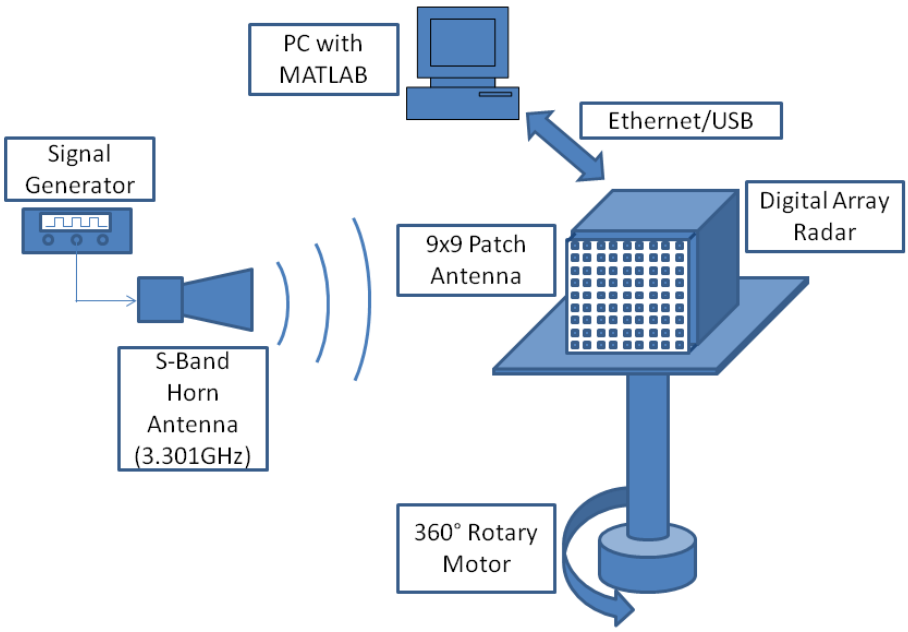

Figure 2.16: Narrowband experimental setup in the chamber with the Army DAR. The DAR chassis and 9x9 element aperture (detailed in Fig. 2.17 sit on a positioner that pivots -90 to 90 degrees in azimuth with a resolution down to 0.1 degrees steps. The I/Q data is streamed to the PC over a 1 GbE interface where the beam patterns are synthesized in Matlab offline. Reprinted from Thompson et al. (2015) © 2015 IEEE.

a compact digital receiver circuit is integrated alongside the transmitter on the same circuit board. The surface mounted integrated circuits used for the transmitter, receiver, and other circuitry allow many channels to be physically implemented with a small area profile. The techniques described for the dual-pol, phased array radar is a novel and unique realization of new weather radar systems.

The beam steering capability of phase array radars has been analyzed and improved. A new method of assigning phase shifter values in phased array

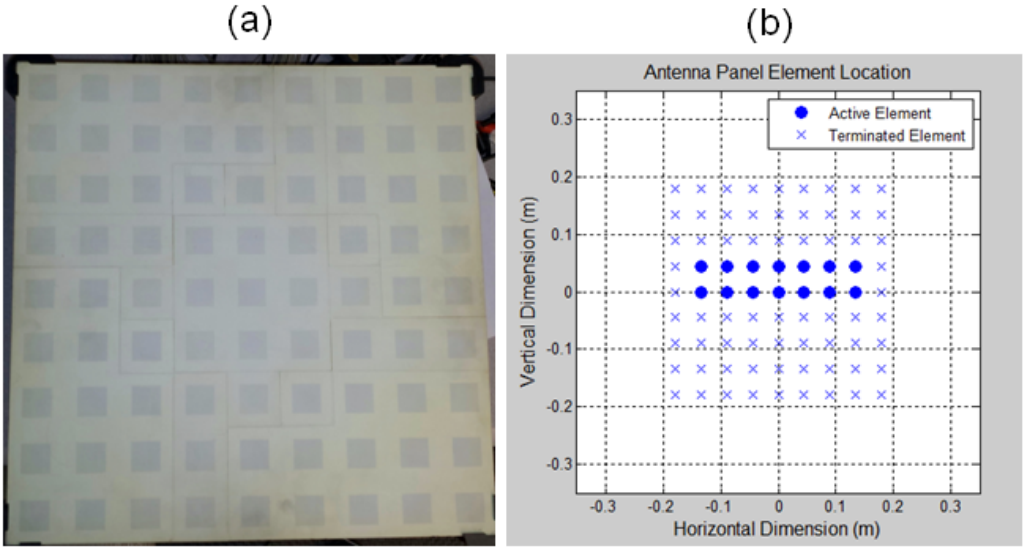

Figure 2.17: In a.) a photograph of the DAR antenna panel and in b.) the spatial configuration of the active and terminated elements. Reprinted from Thompson et al. (2015) © 2015 IEEE.

radars was developed by analyzing the AF for multiple steering angles. It has been shown that array sidelobe levels can be reduced when using a minimal number of phase shifter degrees of freedom (i.e. bits) while the average sidelobe levels are increased. The real beam steering angle error can increase as well with small array sizes. The main beam width was shown to not be affected by the technique. This chapter's proposed approach can always achieve an improvement greater than or equal to traditional beamsteering methods. The proposed method is appealing due to the software controlled architecture that assigns values to digital phase shifter hardware. With these important implementation concepts in mind, the next chapter explores wideband beamforming.

Chapter 3

Wideband Digital Beamforming Techniques and Experiments

The general concept of wideband beamforming in the digital domain has been well studied for some time [24], [25], [28], [67], [68]. Additionally, many RF systems have been developed in the past [34], [35], [47]–[50], [68], [69], though these systems are still not general and/or high performance enough to be used for multiple RF functions.This section introduces new wideband processing techniques to improve the performance of RF systems that digital processing technology has recently introduced. Beam squint over a wide frequency band is experienced with phase shifting since phase is frequency dependent and only an approximation of a time delay (e.g. IBW greater than 10% of the carrier frequency). The beam will "walk" off the intended pointing direction at angles off boresight, degrading the total power on target. Thus, limiting the overall system performance, since beam squint is a function of beam width angle, center frequency, IBW, etc. [30]. In the past, optical approaches to the wideband beamforming problem were pursued [70]–[75], however, circuit area presented a challenge for practical systems with many channels. Today, digital hardware enables the wideband processing to be performed in the digital

domain in either the frequency [76]–[79] or time domain [80]–[82]. Careful consideration of the technique should be made in order to minimize or reduce the hardware resources so that lower power utilization, more beams, wider frequency coverage, etc. is achieved. For example, conversion to and from the frequency domain is done with the FFT and inverse FFT, respectively. These operations are performed in addition to a matrix of vectorized MAC operations used to apply the weights. It is realized that such an architecture utilizes the same DSP resources as a time domain DBF technique using interpolating FIR filterbanks [83], [84] without adding operations to convert between domains.

The wideband model is shown for the $\lambda/2$ uniformly spaced linear array (ULA) in Fig. 3.1 assuming a wavefront impinging an array at an angle θ and element-level beamforming is available. A beam, y, is formed from the combination of M time-delayed elements in the FPGA. In the receive chain of the TR module, the RF front-end filters, mixes, and amplifies multiple analog RF channels in 180 nm SiGe. The signals are then conditioned for digitization by an array of interleaved successive approximation register (SAR) ADCs built in 65 nm CMOS. The final stage of the digital array model is front-end digital processing in an FPGA before offloading beam data to the digital backend for further processing. Here, the 20 nm Arria 10 FPGA from Altera has been used to implement digital calibration and beamforming algorithms in real-time. Manufacturing and packaging errors in the receiver chain can change an element's amplitude and phase response from channel to channel. These errors are inverted with an equalization routine running on the FPGA and are attributed to linear and non-linear effects in the antenna aperture, TR modules, and digitizers. In the remainder of this section, we introduce the integrated multi-use phased array common tile (IMPACT) common module

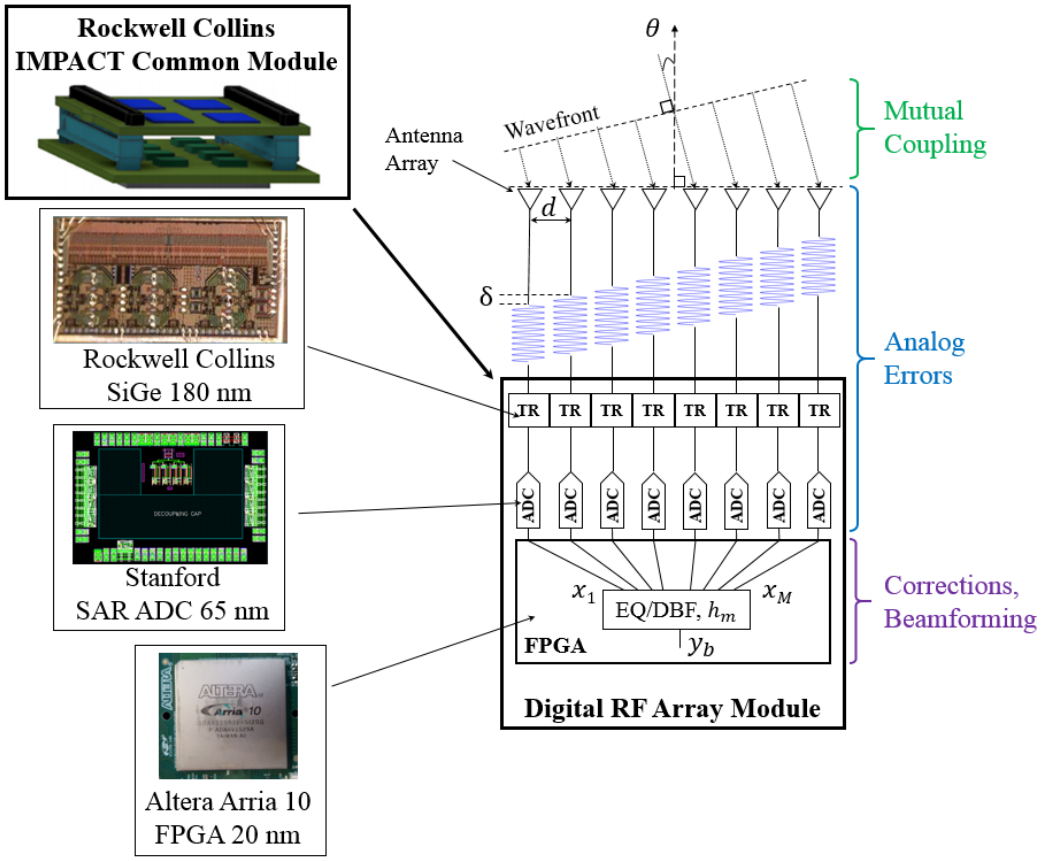

Figure 3.1: Model of beamforming in a digital array. The IMPACT module utilizes elemental digital beamforming on receive and analog beamforming on transmit. Mutual coupling effects and analog errors contribute to the array error which is modeled by a transfer function, g_m. Errors are corrected for in the FPGA by the calibration routine.

hardware architecture and its functionality. The DBF engine of the IMPACT module resides on the FPGA and is discussed at the end of this section.

By creating a single module and reusing the same hardware design as COTS technology scales, non-recurring engineering costs of building a custom array are significantly lowered [85]. The Defense Advanced Research Projects Agency (DARPA) Arrays at Commercial Timescales (ACT) program has supported the construction of a reusable RF-to-digital transceiver module (shown in the 3D concept module of Fig. 3.1) that can be reconfigured using

software to support to a number of different center frequencies, bandwidths, and personalities (e.g. radar and communications). The RF-to-digital module is a low-cost, compact device supporting 16 full-duplex transceivers. Quad channel analog SiGe front-end RF transceivers are represented by the blue squares on the top PCB layer of the module. RF signals traverse from the top layer through a differential, high frequency connector and interface to the 16 individual ADCs residing on the bottom PCB which are designed by Stanford, fabricated in the 65 nm CMOS TSMC process, and packaged at Rockwell Collins. A single FPGA resides underneath the lower PCB layer and is electrically connected to the each ADC with 8 low voltage differential signaling (LVDS) pairs for data and a single LVDS pair for a 350 MHz clock.

3.1 True Time Delay

The effects of beamsquint on the array factor is demonstrated in Fig. 3.2. The resulting beamforming patterns were processed using phase shift only beamforming and TTD DBF techniques. In a.) the TTD was not applied and shows the beam drifting in azimuth for higher frequencies. In b.) the beamsquint is corrected for with TTD.

A time domain technique to implement TTD is the Farrow structure which precisely generates the delay. However, an exact delay value is not required to form high-fidelity beams and can be estimated. The authors have previously shown in [52] that quantization effects of phase shifters can be mitigated when using a limited number of degrees of freedom (i.e. bits). In this case, the degrees of freedom are the number of filter taps in an individual channel. Extending this realization to TTD, exact time delay is not required for large

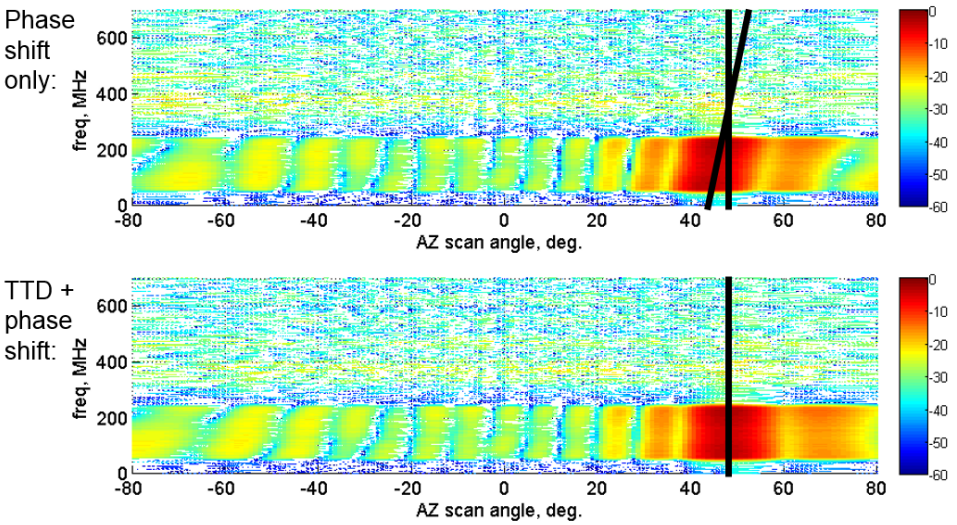

Figure 3.2: Correcting beamsquint with digital true time delay. In a.) the response using a phase shifter only, and in b.) shows the response with TTD included. The main beam "pivots" based on the IF center frequency and the main beam is aligned.

arrays (e.g. > 8 elements in a single dimension). Therefore, in the time domain, an array of vectorized MACs is an efficient architecture to perform TTD DBF. Although the delay value may not need to be exactly precise, quantization resolution must be sufficient so as to not further degrade the estimation. Altera has demonstrated a time delay based beamforming solution in hardware [86] using polynomial interpolation FIR filterbanks to estimate element-level TTD. The polynomial coefficient calculation approach to TTD limits the linearity for the wideband case once quantization errors further degrade the polynomial estimate for high frequencies. For RF signals, Fig. 3.3 shows typical TTD values across practical center frequencies and numbers of elements.

In hardware, Fig. 3.4 shows time delayed signals that were asynchronously captured in random access memory (RAM) and overlaid 2 digital RF memory

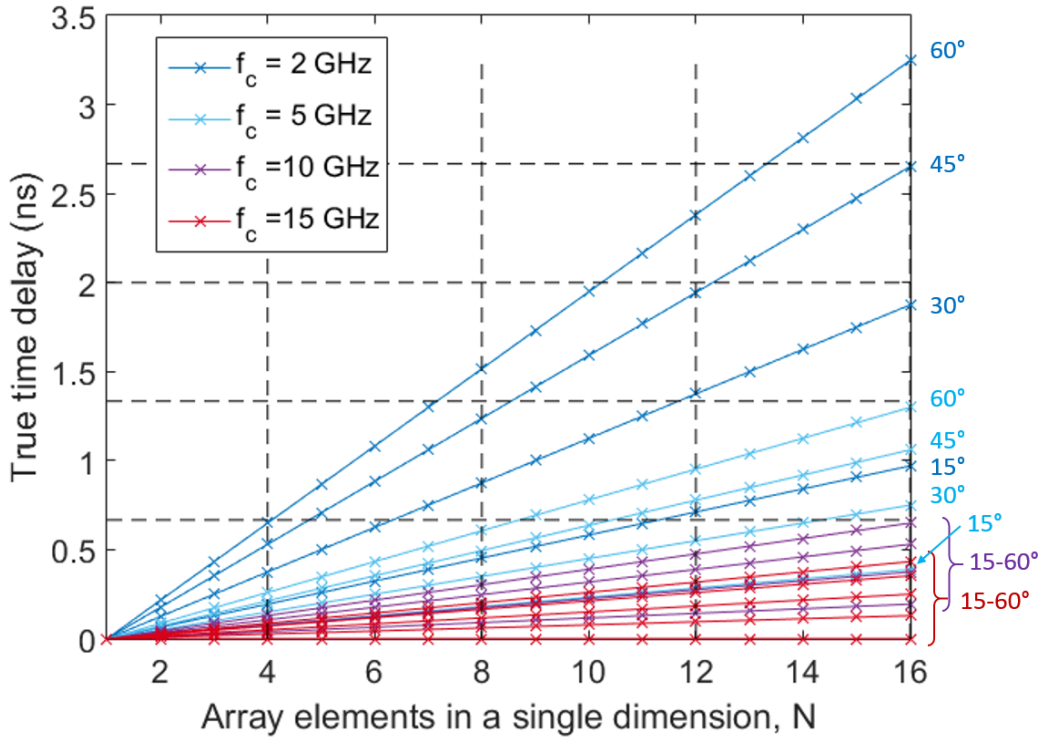

Figure 3.3: True time delay requirements for coarse and fine delays over a range of center frequencies and array sizes for systems configured from scalable IMPACT modules. Horizontal dashed lines represent integer numbers of coarse delays in the FPGA at 1.4 GHz, δ_{coarse}, and vertical dashed lines shown the integer number of IMPACT modules required to support the number of channels in a single dimension.

channels support 40 μs of IF data. At the low frequencies (e.g. the beginning of a pulse) the TTD effect is apparent. The minimum fractional delay resolution is $\frac{1}{Nf_s}$. Max fractional delay is $\frac{N-1}{Nf_s}$. For a sampling frequencyof $f_s = 1.5\ Gsps$ and $N = 25$, $t_{min} = 26.7\ ps$ and $f_{max} = 640\ ps$. Fig. 3.5 shows the minimum and maximum delay using 22 nonzero taps.

3.1.1 Digital Filter Design

True time delay in the digital domain is achieved using clock sample delays and fractional delays. Fractional delay FIR filters, achieving a linear phase

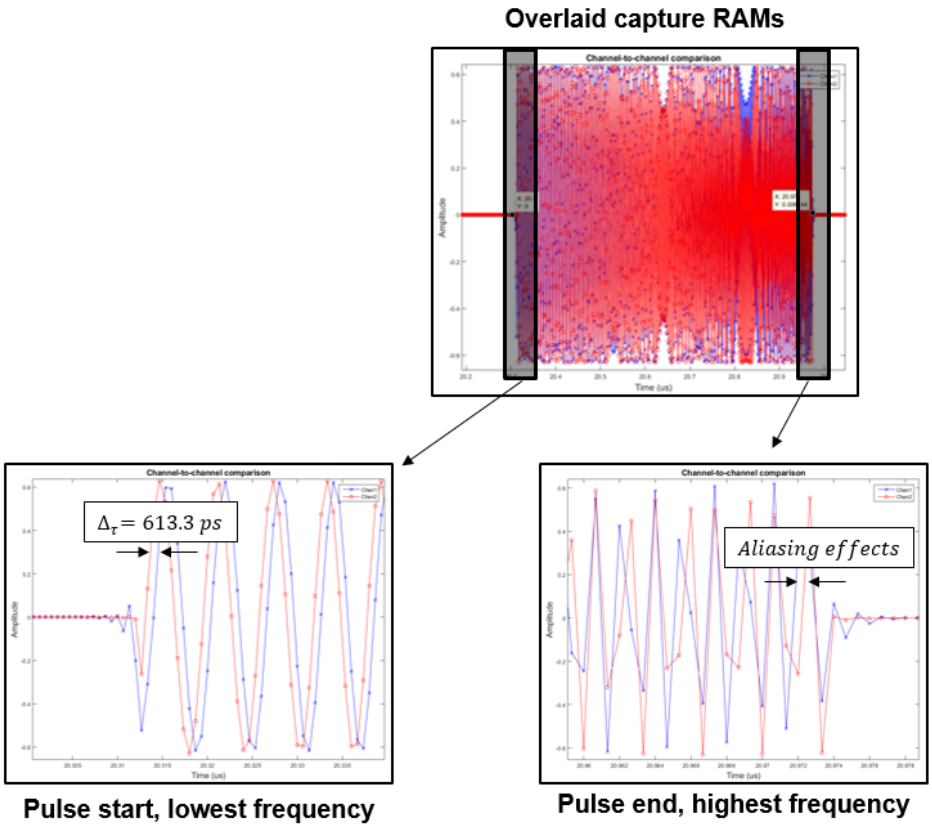

Figure 3.4: Measured RX time series data for two channels showing the fractional delay between channels.

response in the band of interest, can be used to create temporally fine delays in a processing channel [87]. The construction of these FIR filters entails computing impulse response coefficients using techniques such as oversampling or interpolation. A wideband fractional filter is constrained such that all frequencies are passed with constant amplitude response while incurring a constant group delay throughout the entire processing band. Alternatively, infinite impulse response filters (IIR) have been used to perform efficient TTD [88], [89]. However, these techniques typically require a high oversampling factor to ensure flat group delay across the digital bandwidth.

For an interpolation filter that may be used in a timed delayed array, there

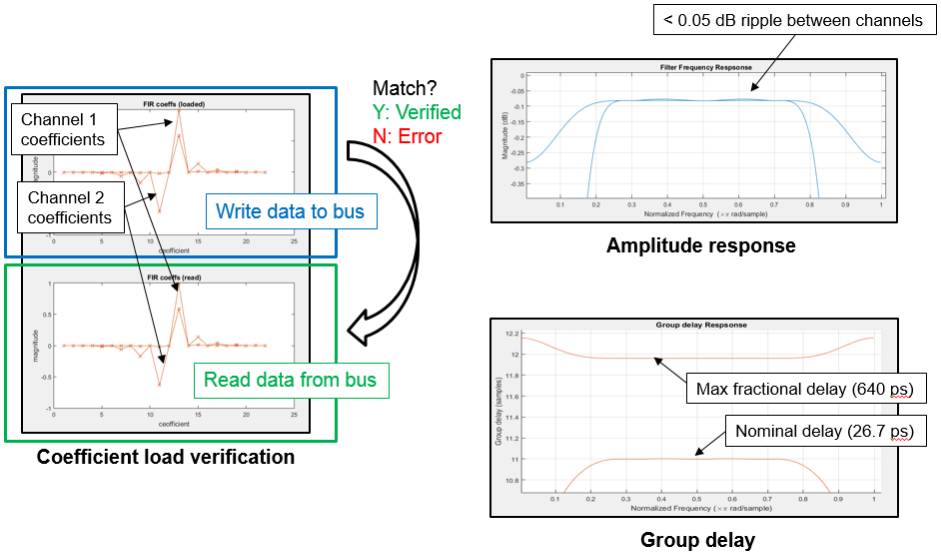

Figure 3.5: Two filters with minimum and maximum fractional delay with 26.7 ps of resolution and P filters.

are five main design criteria to consider when designing a time delay FIR filter. These include group delay error, phase error, amplitude error, filter length and coefficient resolution. In deriving a set of filters for a time delayed array, mismatches between filter sets can degrade the overall beamformer performance. For example, a set of coefficients was found to satisfy the specifications shown in Table 3.1. In the frequency response, it is desired

Table 3.1: True time delay filter bank properties for RF beamforming.

Number of elements (M)	10
Number of filter taps	128
Fractional sample resolution	$.01 f_s$
Amplitude window	Blackman

to have a flat amplitude response across the entire band (i.e. all-pass filters) but is not achievable with finite length filters. To lengthen the filters and effectively double the length, the filters are modulated to a center frequency

that is a quarter of the sampling rate (e.g. half the Nyquist rate) [90], [91]. Modulating the results to center of the band to this frequency in particular allows twice as long filtering lengths with the same number of DSP resources as a result of alternating coefficients being zero.

To align the signals between elements, a filter h_m is designed with arbitrary time delay control. The frequency response of h_m requires a flat amplitude response to support wide instantaneous bandwidths and a linear phase response (i.e. $|H_m(e^{j\omega})| = 1$ and linear group delay). In the digital domain, an estimate of this filter is constructed from a finite length *sinc* function such that

$$h_m = \text{sinc}(\omega_m t - \delta) \tag{3.1}$$

for any TTD value, δ. Fig. 3.6 shows a set of *sinc* functions with subtly offset group delays corresponding to sub-nanosecond TTD resolution. The filterbank consists of all-pass interpolation filters that provide a constant group delay that is slightly offset in delay from adjacent rows in the filterbank. A low pass prototype filter was designed to estimate these parameters with high resolution (i.e. oversampled) and is constructed in the form of a digital FIR filter to ensure linearity in the digital domain, shown in Fig. 3.7 a.). The magnitude responses and group delays are shown in Fig. 3.7 b.) and c.), respectively. The figure shows a coefficient set of 22 non-zero tap FIR filters. The pass band of the filters was designed to occupy 650 MHz of instantaneous bandwidth (86.7% of a 750 MHz Nyquist rate) with 0.05 dB of passband amplitude ripple and 66.7 ps of group delay resolution, as shown in Fig. 3.7. Fig. 3.7 a.) shows the normalized filter coefficients after a Blackman window is applied. The plot in Fig. 3.7 b.) shows the flat magnitude response of each filter, closely

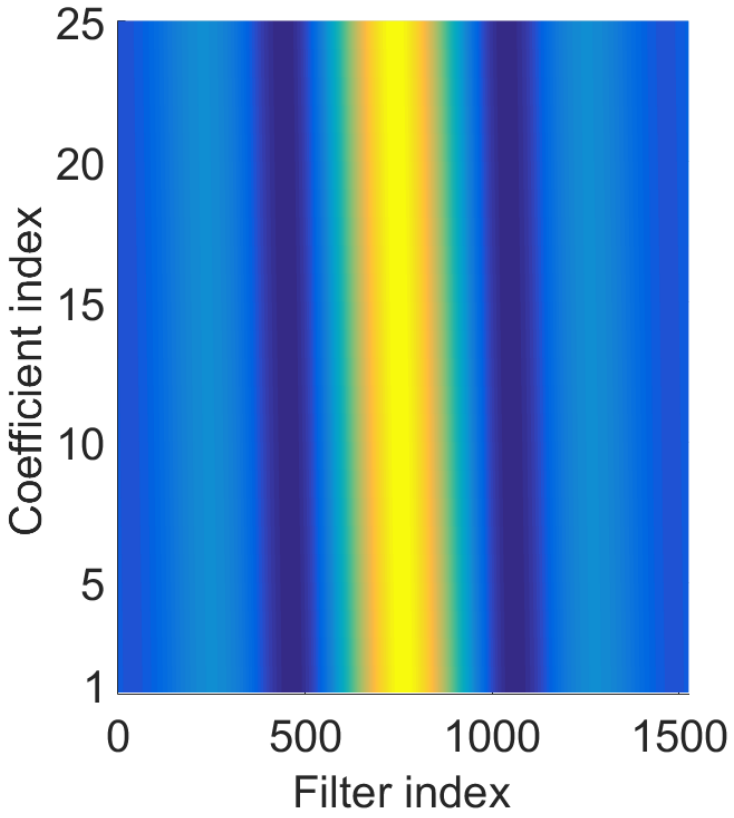

Figure 3.6: TTD *sinc* coefficients for high precision delay values.

overlaid with one another. Fig. 3.7 c.) shows the group delay of a set of 20 filters demonstrating the flat fractional delay over the wide bandwidth, each at a slightly offset value.

Length of the filter defines the degrees of freedom in manipulating the amplitude and phase characteristics of an array channel. In the case of complex baseband signals, complex filters can be used that require three times more DSP operations compared to a real, intermediate frequency (IF) filter. Additionally, DSP routing fabric cannot be utilized which limits the clock speed of the filtering circuits contributing to the complex baseband processing drawbacks. In an FIR filter, typically about 128 taps offers a

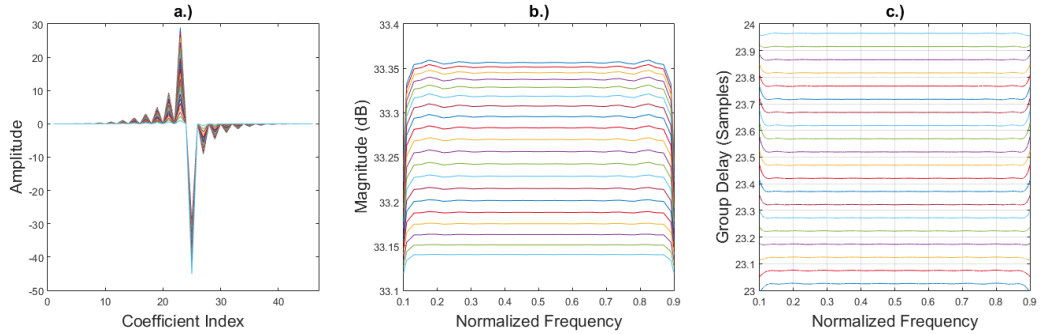

Figure 3.7: A TTD filterbank set composed of 20 filters with 35.7 ps of resolution. In a.), the real-valued time domain coefficients are shown, in b.) the amplitude response of the filter set is shown with less than 0.2 dB of amplitude differences between any two filters, and in c.) a flat group delay across the passband for each filter in the set.

stopband isolation of better than 60 dB. However, due to the wideband characteristic of a channel, an all pass nature of the equalization filter needs to be taken into account. As a result, the additional degrees of freedom can be applied to the phase of the response, further increasing the equalization ability with longer filters. Ultimately, the DSP count of an FPGA limits the number of taps in either a real or complex filter. Current state of the art devices typically have on the order of a few thousand DSP resources and it is expected that this number will steadily increase with Moore's law in future devices. Multiple FPGAs can be systolically combined to further increase the DSP capacity but does not offer increased power efficiency so this technique will not be discussed further. Currently, all DSP resources operate in a fixed point mode where the filter coefficients are quantized, typically taken as 18 bits in both Altera and Xilinx devices. With the introduction of the Arria 10 devices from Altera, fixed point DSPs have been replaced with floating point implementations. This means that the floating point arithmetic logic unit (ALU) is configured in silicon as opposed to the traditional cumbersome

implementation in logic fabric which previously increased the utilization area and lowered the power efficiency of floating point operations. A benefit of the new DSPs includes reduced quantization errors in the processing. However, since the digitizer resolution is bit limited, system performance will not be significantly improved by using floating point devices.

Amplitude tapering (i.e. windowing) is used to shape the frequency response and flattening the group delay; a 46 tap Blackman window is used in Fig. 3.7 since the first and last coefficients are zero. The coefficients in Fig. 3.7 have a signed, 18 bit fixed point representation. For the filter sets generated here, the memory utilization to store h on chip is small compared to the overall capacity of an Arria 10 device. For example, to achieve a resolution of $f_s/20$ (e.g. 35.7 ps), 20 filters with 22 taps require 7.9 Kb of memory. A set of 1000 filters is sufficient for the radar and communications applications studied here. This requires 400 Kb of RAM in order to achieve 714 fs of TTD resolution.

The amplitude response and group delay show the errors of the set relative to each individual filter. In the frequency response, it is desired to have a flat amplitude response across the entire band (i.e. all-pass filters). However phase errors in the all-pass case (seen as ripples in the group delay) urge the implementation of a bandpass filter set centered at 1/4th the sampling frequency, effectively maximizing the passband of the Nyquist bandwidth and pushing errors close to the Nyquist frequency. Shifting the center of the band to this frequency in particular, enables twice as long filtering lengths in the FPGA implementation since alternating coefficients are zero. Careful routing and multiplexing of the data through the filter can be employed effectively halving the DSP resource requirement with the same filter performance

characteristics. Longer filters are paramount to minimizing the ripples seen in both the amplitude and phase relative to other filters in the set. Additionally, amplitude tapering is effective in reducing the errors seen. Windowing techniques can sharpen or flatten the transition band roll-off for better or worse performance with respect to amplitude and phase ripples. Lastly, coefficient resolution can affect the performance of the beamforming channel. Typically, the resolution of the RF system is determined by the ADC and DAC. This is especially true for wideband digitizers since higher frequency sampling rates are typically associated with smaller bit widths.

3.1.2 Wideband Array Calibration

Before beams can be formed, the array must be well calibrated due to amplitude and phase errors across elements. Beam patterns at boresight were formed with and without calibration in Fig. 3.8. In the case that phased arrays have multiple independent transmit and receive channels, it is expected there be misalignments between channels. Calibration, then, is required to equalize these mismatches and enable the collective array to operate as expected. Additionally, absolute calibration properly determines absolute accuracy of the array performance required for calculation of the estimated radar moments. Failure to properly calibrate at all or periodically during system operation can degrade sidelobe performance, directivity, steer beams in the wrong direction, and decrease/increase main beam/null locations, respectively. Authors in [92] highlight the need for calibration where these issues can have a significant impact on the beam fidelity due to the combination of many errors. Calibration is then required to correct the

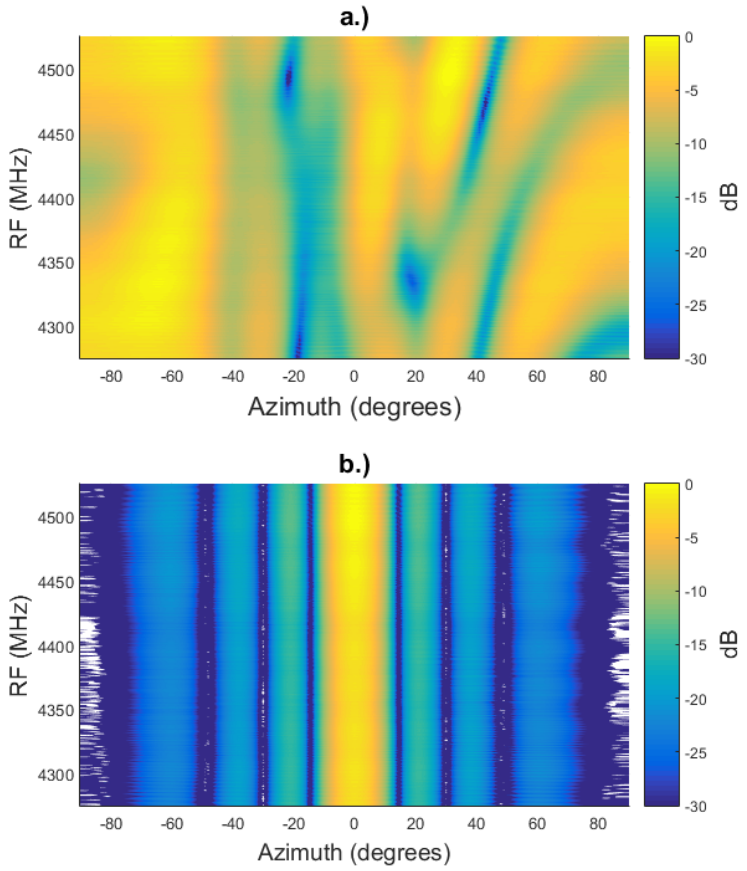

Figure 3.8: Uncalibrated, a.) and calibrated, b.) beam patterns at boresight. A beam is unable to form due to the phase mis-alignments between channels but is formed once the phase is aligned with respect to frequency.

relative mismatches. This is typically achieved with complex valued filters in the time domain, or subbanding and applying phase offsets in the frequency domain. After calibration, remaining errors in phase and amplitude are the main metrics used to assess array performance. Typical state of the art values in past arrays yield less than 1 dB of remaining amplitude ripple and less than 2 degrees of remaining phase mismatches [93]. In the digital domain, the calibration fidelity is directly related to the length of the equalization filters where longer tap length more closely aligns channels. The reconfigurability of digital techniques enables reloading of filter coefficients offering tight

control of the residual amplitude and phase mismatches where the calibration effectiveness can be sacrificed for shorter filter lengths.

Digital at every element arrays have the ability to intimately analyze the phase and amplitude characteristics at each channel. Individual channel control allows channels to operate in either transmit or receive mode at any given time, regardless of the overall array operation. Combining these two features of all digital arrays enables a flexible and convenient calibration routine that can be easily implemented in software that requires no additional hardware. Many existing calibration techniques have been reviewed and summarized in [93], where it has been determined that on-line, in-situ calibration relying on mutual coupling can be effectively used to equalize the array. One advantage of digital FPGA processors is their rapid reconfigurability property. With this in mind, separate calibration firmware builds can be constructed with plenty of processing resources without inhibiting regular processing modes. Alternatively, if enough processing resources are available, the calibration mode can be done on-chip or offline when utilizing the high-speed serial transceivers on the FPGA. On-chip calibration warrants real-time equalization and near real-time equalization if offloaded to a supporting processor. In either case, the calibration mode can run during regular system operation.

There are many factors including temperature, environmental factors, and non-ideal electronics in manufactured components that are non-static and change with time as the array operates. For example, transmitting high power from the array's analog front end can significantly increase front end analog electronic components' temperature altering their operational properties. In particular, analog amplifiers are susceptible temperature increases while

in operation which in turn can affect the gain characteristics. Similarly, environmental issues such as weather or water/ice present on a radome can alter the nominal array performance. Finally, non-ideal electronics such as oscillators used in digitization components (namely ADCs and DACs) and local oscillators in mixers can have small errors in terms of drift. It should be noted that these electronics are prone to instability due to temperature variations in addition to the amplifiers in the system. Over time, these seemingly small errors can accumulate to present significant errors on the array. These factors can be time independent and drive the notion of constant system calibration even while in operation. Luckily, full digital control of the array enables this to be performed regularly during or in between CPIs. Calibration of the modern CM with true time delay differs from conventional approaches. Additional calibration procedures requiring more processing resources and processing time to account for the spectral leakage between channels emanate from the subbanding operation of the FFT in these frequency domain approaches. Larger FFT samples can mitigate these leakage effects but in turn require longer integration times. Additionally, channelization structures based around polyphase filters can be used to further enhance channel isolation compared to the FFT only operation but require significantly more processing resources.

A well know reference waveform is then transmitted and received simultaneously where the relative phase and amplitude at each active receiver is measured. The physical location of the elements is precisely known relative to the transmitter (to a degree of accuracy of the array panel fabrication process, typically much less than a single wavelength of the highest radiated frequency). In-situ calibration techniques are useful for relative calibration

between elements and that a more robust technique is required for absolute calibration. Absolute calibration can be performed at system operation time where well know targets in the far field, such as building, clutter, and reflective targets are illuminated in either the far or near field of the array. Using the well-defined reflection properties of the targets and possibly the relative distance of the said targets from the array enables the array to calculate the absolute transmit and receive power levels needed for radar moment generation. The combination of these relative and absolute calibration techniques is able to properly correct the errors apparent in a phased array radar system permitting the system to operate as expected.

During system operation, a digital array can operate under a "self-calibration" operating mode which relies on mutual coupling measurements is used to equalize the array in both the time and frequency domains. While frequency domain processing requires additional processing, time domain techniques map well to the FPGA stream processor and can be translated readily to an all-digital array where filters are used in beamforming. The self-calibration technique assumes that multiple channels are capable of operating in either transmit or receiver mode simultaneously between other channels which is one benefit of all-digital arrays. Additionally, all-digital arrays can achieve the self-calibration in a fully integrated approach during the normal system operation provided there are enough processing resources for a simultaneous receive channel or beam on the digital processor. In this way, auxiliary channels switched across the array can be utilized to assess the calibration state. In-situ techniques like this relieve the requirement of having additional transmit and/or receive beacons and sensors in the near or far field of the array, simplifying and streamlining the calibration process.

Using a similar calibration technique as in [94], linearly modulated frequency waveforms are used to probe the amplitude and phase response of the IMPACT module during chamber calibration tests. Fig. 3.9 shows a block diagram of the calibration routine. A 4.4 GHz centered upchirp with

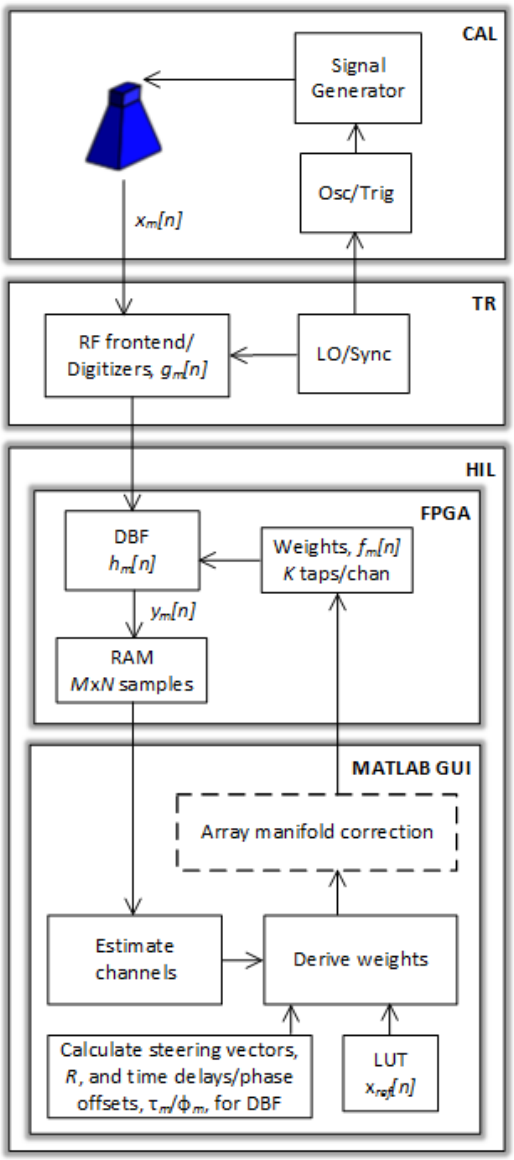

Figure 3.9: Block diagram of the wideband calibration routine. A horn positioned in the far field generates a chirped reference signal that the receiver channels of the IMPACT module are able to align to. Hardware in the loop interfaces the FPGA to the controller PC running Matlab.

200 MHz of bandwidth and pulse width of $\sim$3 μs is generated by an RF signal generator and amplified before driving a standard gain horn antenna positioned in the far field of the chamber with respect to the Vivaldi array aperture. Received chirps are manually triggered to be closely synchronized centered in the capture RAM of the FPGA. Once the phase is extracted, the analog channel errors can be estimated and corrected for with digital calibration. The calibration procedure is divided into 2 parts: correcting for ADC interleaving offsets between sub-channels and phase and amplitude equalization across the array.

Interleaving an array of sub-ADCs is a well known technique used to increase the effective sampling rate of a receiver [36]. The ADC input is split into P sub-ADCs and undersamples each channel by a factor of P, effectively parallelizing the digitization procedure. Phase offsets are applied to the sampling clock at the p^{th} channel as ϕ_p, such that $\phi_p = 2\pi(p-1)/P$. Consequently, voltage biasing errors between sub-ADCs result in mismatched polyphased channels in the FPGA. The average voltage values are calculated for each polyphased channel in the array of ADCs which are subtracted from the corresponding polyphase data streams, effectively removing the interleaving offsets between sub-ADCs. The array is then equalized in amplitude and phase once the interleaving offsets between sub-ADCs have been removed. The procedure is summarized in Table 3.2 as follows:

Table 3.2: High-level flow of the equalization routine

For each array element:
1. Measure channels
2. Construct reference signal
3. Estimate phase and amplitude errors
4. Invert effects of errors
5. Compute TTD and phase values for beamforming
6. Filter channels and beamform
end

3.2 Experiments with the Rockwell Collins Common Module

A reusable RF array transceiver module useful for radar and communication applications has been developed by Rockwell Collins [36], [95]. A single array module supports 16 single-pol channels operating from S to X-band with a 200 MHz wide instantaneous bandwidth and a projected cost of roughly $45 per channel. Wideband calibration measurements and element-level true time delay based beam patterns at C-band on receive are presented in this chapter. The functional block diagram of the IMPACT common module is shown in Fig. 3.10.

The chamber experimental setup is shown in Fig. 3.11 In the experiments discussed here, the Rockwell Collins balanced antipodal Vivaldi antenna (BAVA) aperture [96] interfaces directly to 16 channels of the IMPACT module over coaxial cabling to create a singularly polarized ULA (not shown). The BAVA array is spaced for $\lambda/2$ spacing at 18 GHz; however, the IMPACT module channels were populated such that every other element in the center row was used, achieving $\lambda/4$ spacing at 4.4 GHz. The unused elements of the

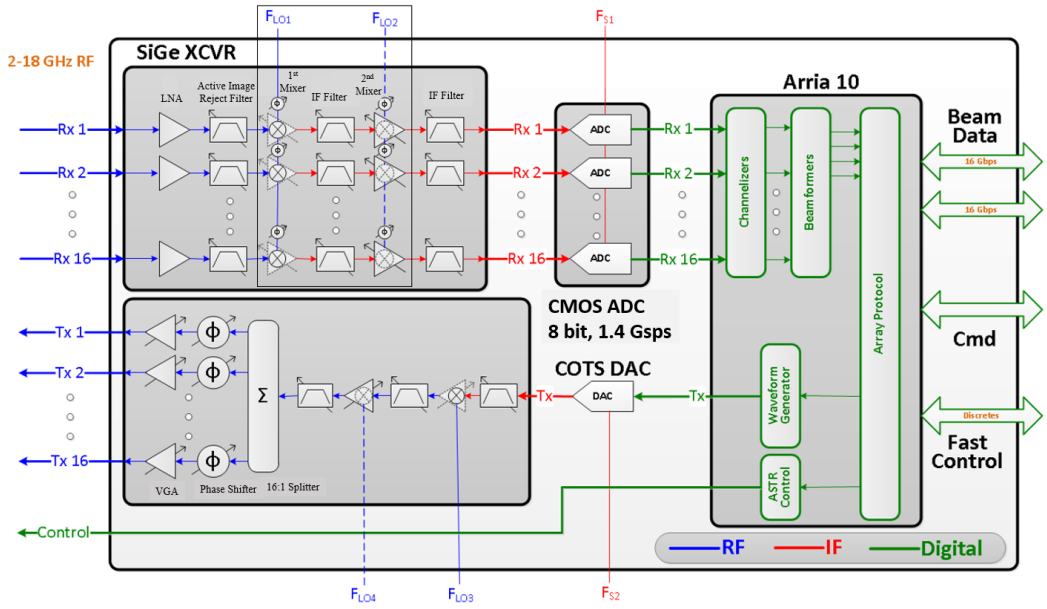

Figure 3.10: The block diagram of the IMPACT common module digital front-end featuring a 16 channel SiGe analog transceiver array, CMOS ADC/DAC array, and Arria 10 FPGA. Reprinted from Hoffman et al. (2016) © 2016 IEEE.

BAVA array are terminated. The clocks and power sources are supplied to the test fixture externally. A 4.4 GHz center is mixed to 1.2 GHz using the local oscillator (LO) of the SiGe transceiver tuned to 5.6 GHz and filters the image resulting from the high-side mixing operation. The ADC operates at 1.4 Gsps (i.e. 700 MHz Nyquist rate) and samples in the 2nd Nyquist zone with 8 bits of precision. The digitized data is sent from the ADC to the FPGA over a parallel LVDS bus where signals are buffered and fanned out into individual processing streams of the DBF polyphased filterbank engine. System characteristics and performance metrics used during DBF tests and measurements are shown in Table 3.3. It should be noted that the IBW is lower than the sampling frequency, f_s. The DBF algorithms are intended to operate over the entire sampling bandwidth, however, this work does not

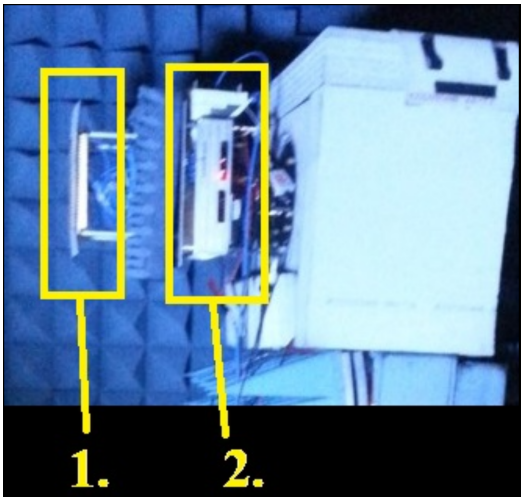

Figure 3.11: IMPACT common module test fixture and BAVA aperture in the chamber.

Table 3.3: Demonstrated system parameters of the IMPACT common module prototype

Center Frequency	4.4 GHz
Sampling Frequency	1.4 GHz
Polyphase Channels	4
Instantaneous Bandwidth	200 MHz
Number of Beams	2
Throughput	696 GMACs
Input Data rate	180 Gbps
Output Data rate	44.8 Gbps
Module Power	< 40 W

demonstrate the full processing capability of the wideband beam case due to the front end RF filter bandwidth limitations.

The total TTD, τ_m, at the mth channel required to steer to angle θ is determined by the array geometry in Fig. 3.1 where d is the spacing between adjacent elements.

$$\tau_m = \pi(\sin(md) - \sin(\theta)) \quad . \tag{3.2}$$

We then denote

$$\Delta_m = \text{round}\{\tau_m f_s\} \tag{3.3}$$

as the integer portion of the desired delay on m^{th} channel; this is trivially implemented in digital hardware by a register delay block. The fractional delay is then

$$\delta_m = \tau_m - \Delta_m / f_s \quad . \tag{3.4}$$

At boresight (e.g. $\theta = 0$) the beam is formed by combining all the elements with $\tau_m = c$ for any constant value spanning the pass band. In practice, the lowest latency group delay filter is chosen as the constant reference. The sum of delayed inputs, x_m forms a coherent beam, y, in (3.5).

$$y(t) = \sum_{m=1}^{M} x_m(t - \tau_m) \tag{3.5}$$

The delay in (3.5) can be commutated from x and applied in a filter, h, as

$$y(t) = \sum_{m=1}^{M} x_m(t) * h_m(t - t_m) \quad . \tag{3.6}$$

This is a beamforming approach suitable for direct RF conversion architectures. However, in a heterodyne architecture such as the IMPACT module, a downconversion mixer introduces additional complexity to the beamformer in (3.6) and adds an additional phase offset term that depends on the beamforming angle and frequency.

$$\phi_m = 2\pi\tau_m(-f_{RF} + f_s/4) \quad . \tag{3.7}$$

The TTD coefficients and phase offsets are combined into a single structure

in hardware. The filter is updated to include the combined EQ filter, time delays, and phase shifts. as

In the chamber, the IMPACT common module test fixture is positioned on a pedestal and swept from -60 to 60 degrees in azimuth with increments of 15 degrees using the calibration coefficients computed at boresight. The patterns in azimuth are shown in Fig. 3.12. Once calibrated, the beam shape is maintained as the pedestal pivots, represented by the different rows in Fig. 3.12, demonstrating robust calibration. In Fig. 3.13, TTD is applied

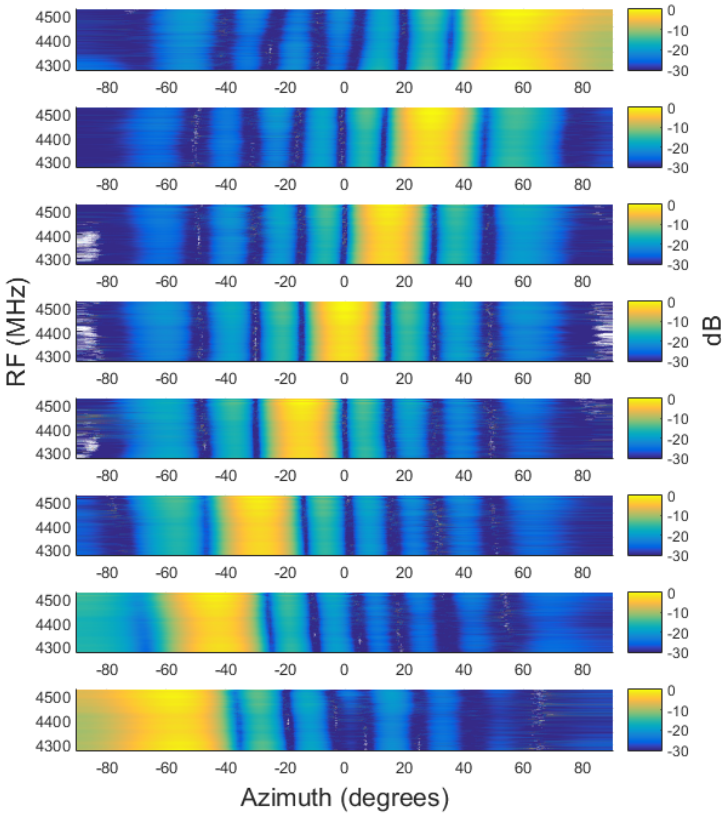

Figure 3.12: Beam patterns across the instantaneous bandwidth of 4.3 GHz to 4.5 GHz for pedestal positions -60 degrees to 60 degrees in increments of 15 degrees. The array was initially calibrated at boresight and the patterns are synthesized from the same coefficients and different pedestal positions.

electronically to steer the beam to 0 and 30 degrees. The patterns can be

generalized to show beam coverage for all azimuth and pedestal angles.

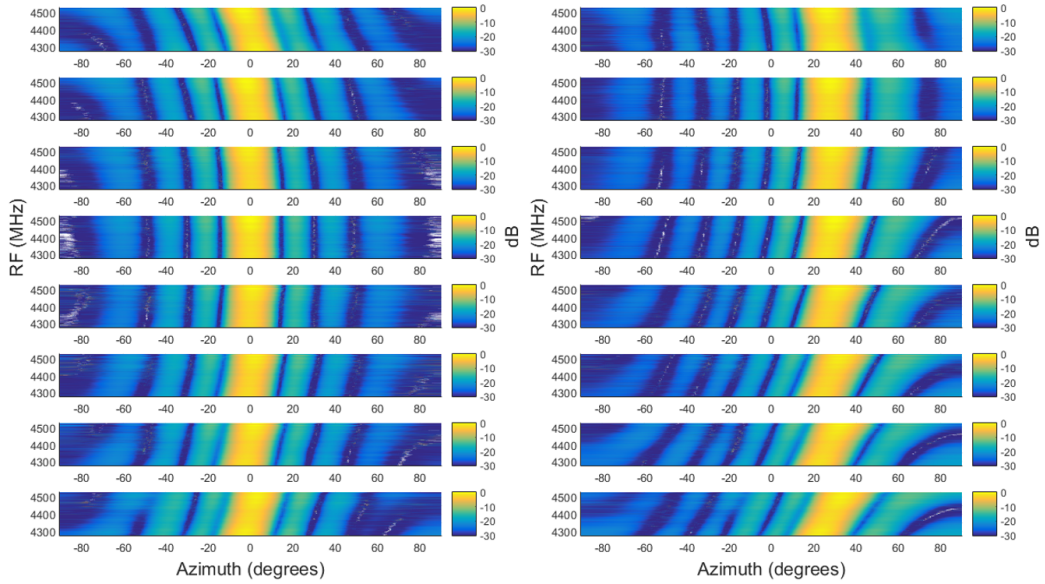

Figure 3.13: Beam patterns across the instantaneous bandwidth of 4.3 GHz to 4.5 GHz for electrically steered azimuth angles of 0 and 30 degrees over pedestal angles of -60 degrees to 60 degrees in increments of 15 degrees.

3.3 Chapter Summary

An element level wideband digital beamformer at C-band was demonstrated in the Rockwell Collins IMPACT module. Multiple simultaneous beams on receive are formed in real time in the digital FPGA processor. The 16-element RF-to-digital array common module on receiver offers low-power and high-throughput digital front-end processing engine is implemented in the 20 nm COTS Arria 10 FPGA. The FPGA was loaded to 96% capacity running at 350 MHz with an IBW of 200 MHz for two simultaneous beams. The module is shown to support a 180 Gbps sampled input data rate and processes up to 2 simultaneous output beams at 22.4 Gbps each.

DBF on the IMPACT module required precise calibration in the amplitude and phase response between the digital channels. Once calibrated with a precision on the order of hundreds of femtoseconds group delay resolution gradients across the array were shown to steer beams with high accuracy. The main digital signal processing (DSP) structures of the DBF and EQ routines are executed in an FIR filterbank that combines the frequency response of both functions into an array of 16 channel-level convolutions. The streaming outputs of the convolutions are summed to form a wideband beam.

The previous two chapters have discussed the importance of fixed beamforming in which coefficients do not necessarily change with time. The following chapter describes applications and efficient solutions for applications requiring the coefficients to adapt in time.

Chapter 4

Adaptive Techniques for High Data Rate Digital Beamforming

4.1 Adaptive Digital Beamforming and Methods

Beamforming with many array sensors enables flexible processing in both the spatial and temporal domains due to increased number of degrees of freedom. In the temporal domain, signals are processed with respect to time in that samples propagating through an operator (time domain filter or fast Fourier transform (FFT)) are related to adjacent samples in which they are separated by the sampling clock frequency. Rather than in time, spatial processors operate on adjacent samples at the array level. The spatial filter can be likened to a temporal filter in that subsequent samples are in adjacent elements where filter taps do not show any time dependence as seen by the narrowband case in [67].

Extending the spatial filter into a wideband operational mode requires the temporal filter addition demonstrated by the Frost beamformer [97]. In a completely configurable spatial and temporal filter bank where filter weights are updated in time, i.e. space-time adaptive processing (STAP), beamshape and the frequency response of the beamformer can be arbitrarily tuned to

direct a main lobe in the beam pattern in azimuth or null out interference from spatially correlated targets in a wideband manner. To null the response from an arbitrary angle using the true time delay approach, beam nulls can be applied to a beampattern using a spatial filter. Namely, spatial elements can be thought of as a "sideways" FIR filter with respect to the time domain. Techniques exist that adaptively tune the filtering coefficients in order to steer beams and place nulls, most a form of the classic Frost beamformer [97]. The Frost beamformer is widely utilized and its variations improve on the performance in one way or another, however, all suffer from the requirement to oversample the bandwidth. These approaches assume the wideband capacity is located inside a subband of the overall sampling bandwidth for a digital system or are performed solely in the analog domain. In the work described here, wideband beamforming attempts to utilize the entire Nyquist sampling bandwidth. A linear constraint on the phase response of the filtering coefficients is essential to fulfill the wideband requirement.

An efficient multi-beam DBF [23] capability enables new approaches to classical problems such as the adaptive beamspace processing [98], [99] where traditional narrowband nulling techniques do not converge to a real time solution over a wide band (e.g. inversion of large covariance matrices). Previously, narrowband techniques [57]–[60] fall short in terms of supporting the IBW requirements of multi-function RF systems. Though, the adaptive beam nulling technique has been shown to form wideband adaptive beams [100]–[103] where complexity depends on the array size (e.g. number of channels) rather than the number of temporal degrees of freedom. Beamspace processing relies on beam inputs to feed the covariance matrix with the number of beams, B, attributing to complexity and degrees of freedom of the

system [99], [102], [104], [105].

In some cases, the DBF techniques described in Chapters 1, 2, and 3 do not work well under certain operation environments. This chapter discusses adaptive DBF techniques for both narrowband and wideband applications that are able to suppress interference by forming nulls in the spatial pattern in addition to beams. It is assumed that by beamforming and nullforming together, a real time beamforming engine can operate in practical scenarios. The remainder of this chapter introduces these scenarios and describes digital architectures that can operate in such conditions.

The DBF block diagram is updated for the adaptive scenario and shown in Fig. 4.1. Similar to previous chapters, the outputs of a digital filter bank form

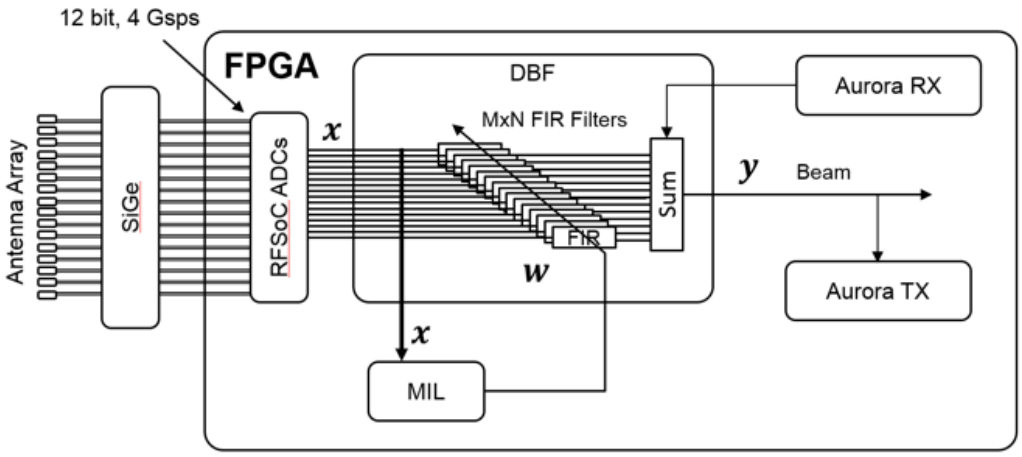

Figure 4.1: Block diagram of a an example feed-forward adaptive beamforming architecture.

a beam y by applying weights to sampled signals and combining the results.

$$\mathbf{y} = \mathbf{w}^H \mathbf{x} \tag{4.1}$$

By solving an equivalent filter design problem, beam pointing and null

directions can be designed in the spatial pattern of the array [106]. It should be noted that in a directly computed pattern, an additional direction of arrival processor is required to track interferers. An adaptive algorithm can be employed that will fine tune the array pattern in-situ based on the characteristics of the mismatched channels. This technique attempts to correlate the SOI between channels with itself while automatically steering nulls in the beampattern in the direction of the interferers. In this way, the adaptive algorithm continuously updates the filter coefficients while placing restrictions on the adaptive weights such as DOAs, linear phase filters, and others including the minimization of statistics of either the SOI or interference. Since the algorithm employs a cross correlation between channels, the algorithm can also correct channel mismatches that may exist from a lack of or poor calibration. In this case, lower system calibration requirements can be applied to the system since the calibration fidelity can be relaxed.

The number of nulls in the array's spatial pattern is determined by the rank of the spatial filter (i.e. the number of elements in the array). The nulls are created by zeros in the filter transfer function. The number of zeros determines the number of coefficients available in the filter. For a filter of order M, there are $M-1$ complex zeros available and as such, there are $M-1$ degrees of freedom with respect to nulls in the spatial pattern as defined by the Paley-Wiener theorem [26], [29]. As a result, it is true that larger arrays boast more degrees of freedom in which more precise and accurate beam and null control is available.

There exist two prevalent interference mitigation techniques used including non-coherent and coherent signals [107]. Non-coherent techniques do not rely

on knowledge of the target radar and typically implement noise jamming waveforms where the jammer radiates self-generated RF waveforms. The noise bandwidth can be configured to radiate at a single frequency or over a broad band. Additionally, noise modulation can be implemented in amplitude, frequency, or phase to further degrade the SNR performance of the target radar. Counter measure techniques that combat these non-coherent jammers employ a constant false alarm rate (CFAR) processor which can effectively minimize the jamming signal using STAP since the non-coherent signals are spread throughout all range bins in the detected CPI. This has motivated coherent jamming signals which attempt to combat the CFAR processor by selectively jamming based on properties of the target radar (e.g. pulse repetition interval (PRI), scan strategy, and frequency) as opposed to the "barrage" like nature of the non-coherent techniques. The coherent jammers typically detect the waveforms radiated by the target radar and store a copy in a digital RF memory (DRFM) onboard the jamming processor. Captured waveforms are then recalled and radiated back by the jammer toward the target where "spoofed" pulses are detected as false returns usually interpreted as either false plots or false targets, depending on the exact jamming technique. The jammer to signal ratio (JSR) is useful to assess the effectively of the jammer suppression in either an adaptive or conventional beamformer. A good assumption about the interference is that a jammer is spatially correlated (i.e. the jammer is either stationary or moving slowly).

During system operation, where the environment may not be well known, the power quantities of the SOI and jammer contributions will be estimated. One method to estimate the power of both quantities can be likened to the generalized sidelobe canceller problem, previously discussed. For example, two

beams can be generated, one looking in the direction of the SOI and the other in the direction of the jammer. Again, sidelobes will contain contributions from the other beams, so spatial filtering with nulls directed towards the opposing beam is required for a completely accurate measurement of individual beam powers. Integrating the power across valid frequencies and taking the ratio of the jammer power with the SOI power gives the JSR.

4.1.1 Forming the Covariance Matrix

A significant drawback of the pulse-to-pulse processing techniques is evident when the "slow moving" jammer assumption is violated. Such a case is demonstrated by either a.) target/jammer location is changing quickly relative to a fixed or moving platform or b.) a platform is moving while detecting ground targets. In any case, if the statistics of detections in the spatial domain vary significantly between PRIs in a CPI, the rejection capability of the radar processor will degrade. In such a case, fast time adaptive processing can be used to negate the effects of a fast moving target. This way, weights are updated on a sample by sample basis. Since the adaptive algorithm requires training data, analyzing many PRIs for the slow-time approach is required. In that a target is moving faster than or as fast as the convergence rate, the covariance matrices will not be able to keep up with the target. Typically, the convergence rate can be tuned to minimize the time it takes for the weights to settle, but multiple PRIs are still required to populate the covariance matrix. Utilizing the fast-time adaptive approach, on the other hand, can significantly reduce the adaption time of the slow-time adaptation by associating the convergence rate to the sampling rate rather than the PRI.

Slowly moving jammers or interferers and ground clutter can be mitigated through space and time adaptive processing. This technique correlates returns from multiple pulses in slow time across all channels and range bins throughout the array. In large systems, this processing can become a large burden to the memory IO resources of the FPGA. In particular, data from multiple sources is required to aggregate through a common node at some point in the processing. Computing the adaptive weights requires a matrix inversion of the covariance matrix which can be significantly large for many element arrays. It is a requirement that all covariance data must be present in the memory before performing the matrix inversion operation. At the FPGA level, this requires external memory, typically double data rate (DDR), to hold and wait for the complete data set. For high rate digitizers, the internal memory of state of the art FPGAs is not large enough to hold the large sample set. Once the data has been stored to memory, the STAP computation can be performed in bursts to compute the adaptive coefficients. Typically, for arrays with multiple FPGA processing nodes, data communication between the nodes is required to complete the inversion operation. The downfall of this data communication reduces the iteration time and increases the power consumption of the system effectively limiting the efficiency of performing the STAP operations on the FPGA. The covariance matrix is constructed from two main configurations shown in Fig. 4.2 and Fig. 4.3. As more elements or time samples are added, the covariance matrix grows linearly which incurs a $O(n^3)$ latency to invert. In general the performance and number of degrees of freedom (e.g. beams and nulls) are determined by the size of the covariance matrix.

$$\mathbf{R}_x = \frac{1}{N}\mathbf{X}\mathbf{X}^H \tag{4.2}$$

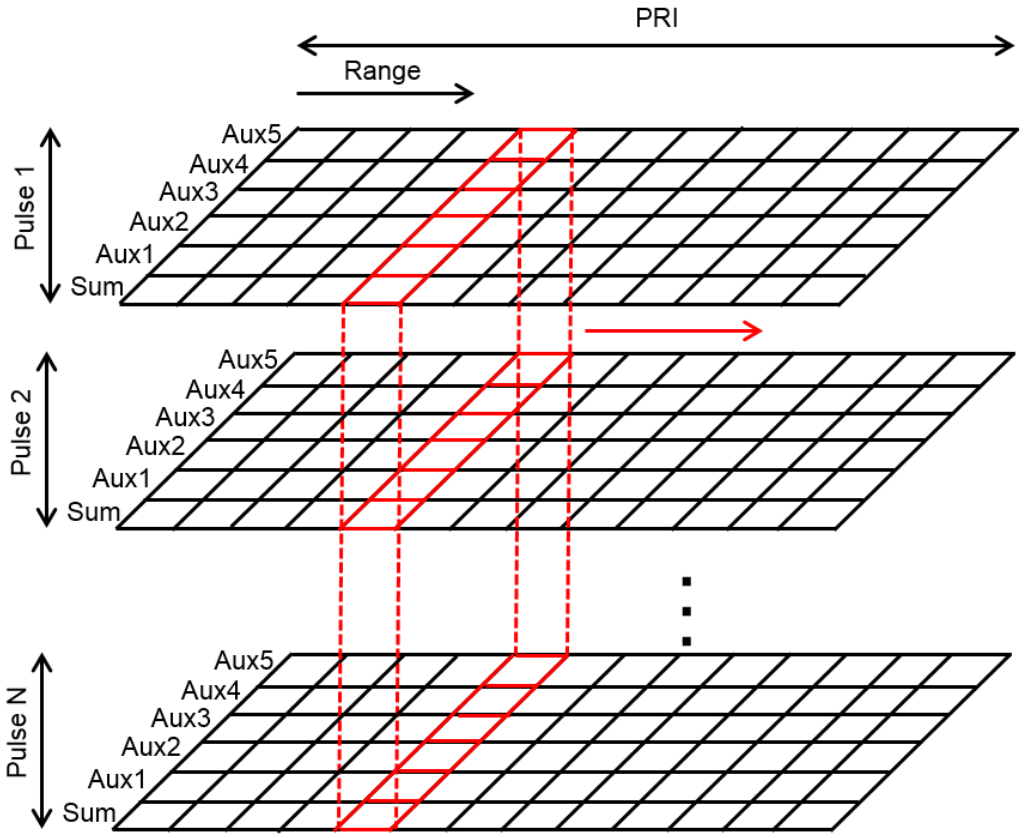

Figure 4.2: Covariance matrix built with channels and pulses in slow time.

For example, Fig. 4.4 shows how a $1 \times N$ and $M \times 1$ dataset combines to create the same computational latency if $N = M$.

4.1.2 Inverting the Covariance Matrix

In Chapter 3, a weighted least squares approach was used to find calibration coefficients. This section discusses a similar analysis in that filter coefficients are found to minimize the least squares error of the phase and amplitude responses [108]–[113]. Of particular interest is the amount of processing and latency required for each approach. Weights are found to minimize the variance

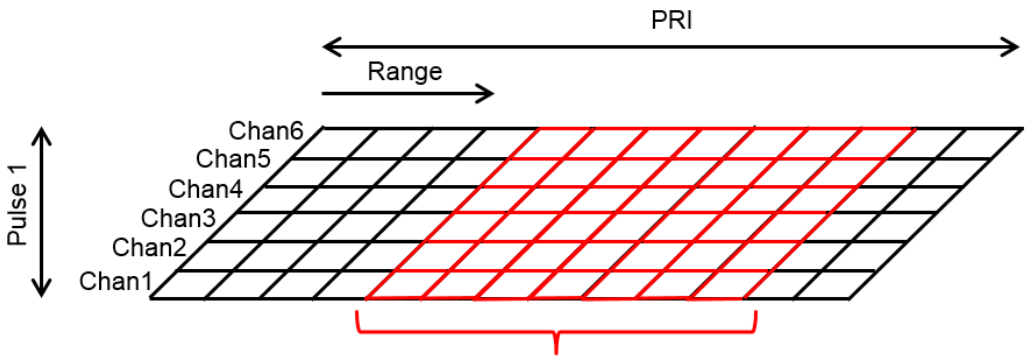

Figure 4.3: Covariance matrix built with channels and range samples in fast time.

of the uncorrelated noise source [103], [114] and computed as

$$\mathbf{w} = \mathbf{R}_x^{-1} C (C^H \mathbf{R}_x^{-1} C)^{-1} \quad . \tag{4.3}$$

In addition to the steering matrix, or constraint matrix, C, the autocorrelation and it's inverse are used to compute the complex weights, $\mathbf{w}$. In hardware, many approaches have been described that decompose the covariance matrix into a combination of triangular matrices [115]–[125].

4.1.3 Feed-forward Methods to Invert the Covariance Matrix

To relieve some of the computational load, "look-ahead" processing techniques may be required to pipeline the matrix operation so the processor can complete the adaptive processing in a streaming manner [126]. For larger arrays with multiple parallel FPGAs, data will inherently be required to transfer to other nodes in a systolic manner, significantly limiting the real time adaptive update rate. The latency associated with multiple device transfers will dominate the

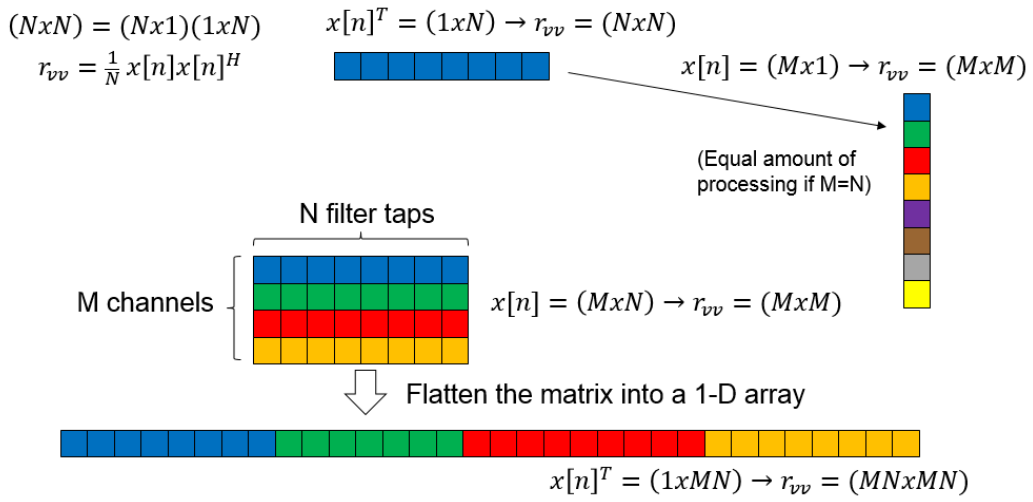

Figure 4.4: The covariance matrix can be constructed from multiple configurations of the array, however, is ensured to be a square matrix.

processing latency required to compute matrix inverses and accompanying multiply and add operations. In order to overcome the processing burden of inverting large matrices on the FPGA, the matrix inversion lemma (MIL) can be utilized so that these inversion operations need not be performed at all adaptive refresh times. Rather, for each new adaptation of the weights, the MIL states that a new set of weights can be found that is closely related to the previous value with the assumption that the matrices will not fluctuate significantly throughout the processing interval.

$$\begin{aligned}\mathbf{w}(k) = \mathbf{w}(k-1) - [\frac{\mathbf{r}_{vv}^{-1}(k-1)\mathbf{v}(k)}{\lambda + \mathbf{v}^T(k)\mathbf{r}_{vv}^{-1}(k-1)\mathbf{v}(k)}]\mathbf{v}^T(k)\mathbf{w}(k-1) \\ +\mathbf{s}(k)[\frac{\mathbf{r}_{vv}^{-1}(k-1)\mathbf{v}(k)}{\lambda + \mathbf{v}^T(k)\mathbf{r}_{vv}^{-1}(k-1)\mathbf{v}(k)}]\end{aligned} \tag{4.4}$$

4.1.4 Improving Adaptive Latency

Multiple ADBF techniques were studied and compared to identify the highest data rate throughput. The size of the covariance matrix for various scenarios

was modified and run against the following four cases.

- Matlab x86 *inverse* function
- Xilinx Virtex 7 QRD in HLS
- Xilinx Zynq Ultrascale+ QRD in HLS
- Xilinx Zynq Ultrascasle+ MIL in HDL Coder

The latency of the Matlab case was found using the *tic*/*toc* function to time only the ADBF portion of the DBF algorithm over 200 trails. Xilinx provides an ADBF capability with the high-level synthesis (HLS) software which implements inverting the covariance matrix with the QRD method. Assuming a 200 ns clock is used with 392 of 3600 DSP48s utilized, 412.4K clock cycles are required and takes 3.3 ms to process a single coefficient adaptation frame. The MIL technique in HDL Coder was compiled which includs no pipelining latency and only latency from the multipliers since it is assumed that the DSPs can be pipelined to include the addition operations required. The results are summarized in Fig. 4.5. In the manner in which the MIL is a recursive algorithm, it requires fewer DSPs and incurs less latency though is not met without criticism [127]. One advantage of performing the adaptive beamforming on the FPGA enables ultra-fast response times in the calculation of adaptive weights. Since the data need not be transferred to another digital processor, the required calculations can be made on the order of FPGA clock cycles. This is true for small matrix sizes (i.e. small arrays) where embedded FPGA RAM can be used to store data and recalled quickly. For large matrix sizes (i.e. large arrays), external RAM may be required to retain all the range samples needed.

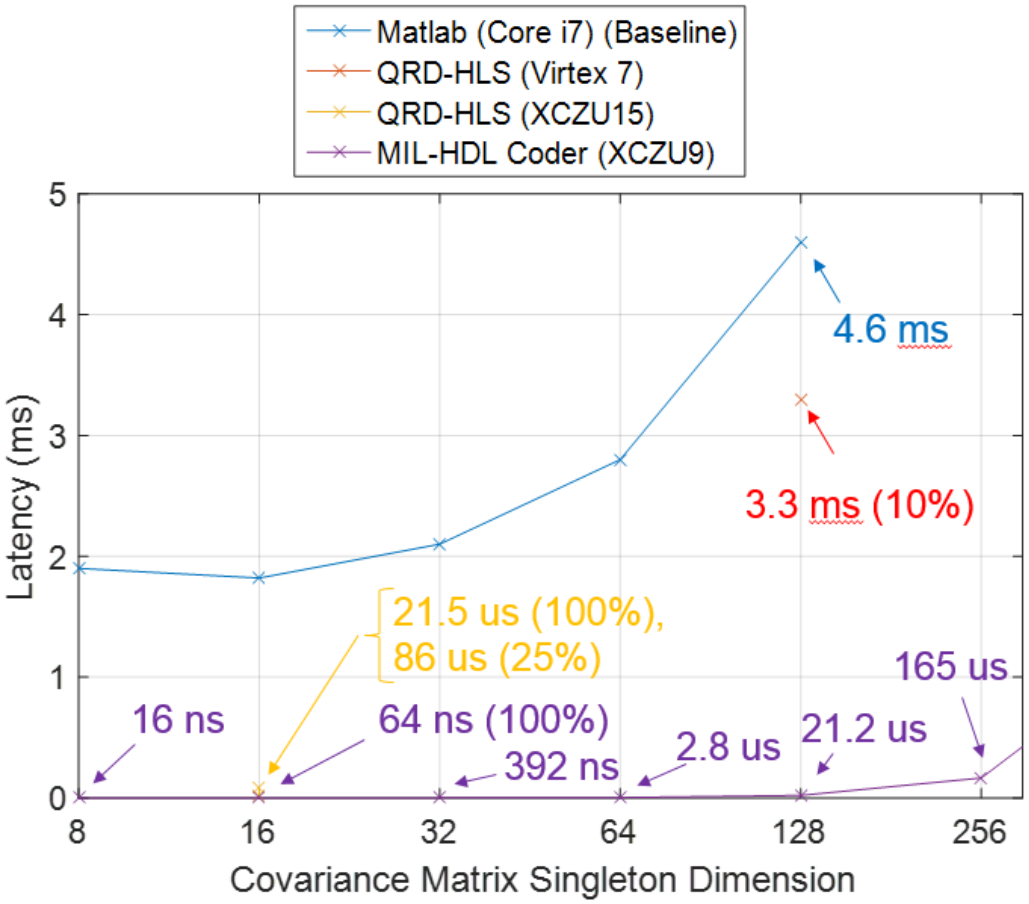

Figure 4.5: Latency for ADBF algorithms for different covariance matrix sizes.

4.2 ADBF in the Beamspace

Digital beamforming coherently combines array channels to maximize the main lobe of a steered beam to a certain azimuthal angle. However, it is typical such beam patterns will contain sidelobes pointed in a direction away from the main steering location. Interference can then contribute to main beam pattern through the sidelobes. As such, it is desired to steer nulls in the beampattern at the interference angles to minimize the energy radiated from these directions. In the analogy of a spatial filter, where element channels are time delayed taps, main beam azimuth angles correspond to poles, where

nulls correspond to zeros as seen in the transfer function of the spatial filter. The number of zeros, corresponding to the number of nulls, is reflected in the number of array elements, M. As M increases, more degrees of freedom are introduced in the beam pattern. Like beams, nulls drift in angle at different frequencies. Similarly, in the narrowband approximation, the null depth will widen and skew over frequency, essentially removing the null altogether in a wideband system. The wideband approach using filters, rather than phase shifters, prevents these drifts over frequency while keeping the depth and position constant. The null width, however, will depend on frequency, similar to how the beam width is affected by frequency. The beam patterns steered in two different directions are shown in blue and green in Fig. 4.6. Multiple frequency cuts are shown in the same overlaid colors.

4.2.1 Digital Generalized Sidelobe Canceller

Adaptive filters relying on cross correlation computations between multiple signal and interference channels discriminate the signal of interest (SOI) using additional auxiliary channels. It is possible to impose restrictions on at least one channel in order to improve the cross correlation properties between the SOI and interference. Typical restrictions include directional, statistical, and other restrictions associated with the phase and amplitude characteristics of the channel (e.g. linear phase requirements). While these techniques can be used in an adaptive system, hardware resource requirements typically prevent these operations from occurring in the time domain in a real-time manner. A single auxiliary channel canceller, for example, requires twice the processing resources of a typical non-adaptive implementation. The reason is that the

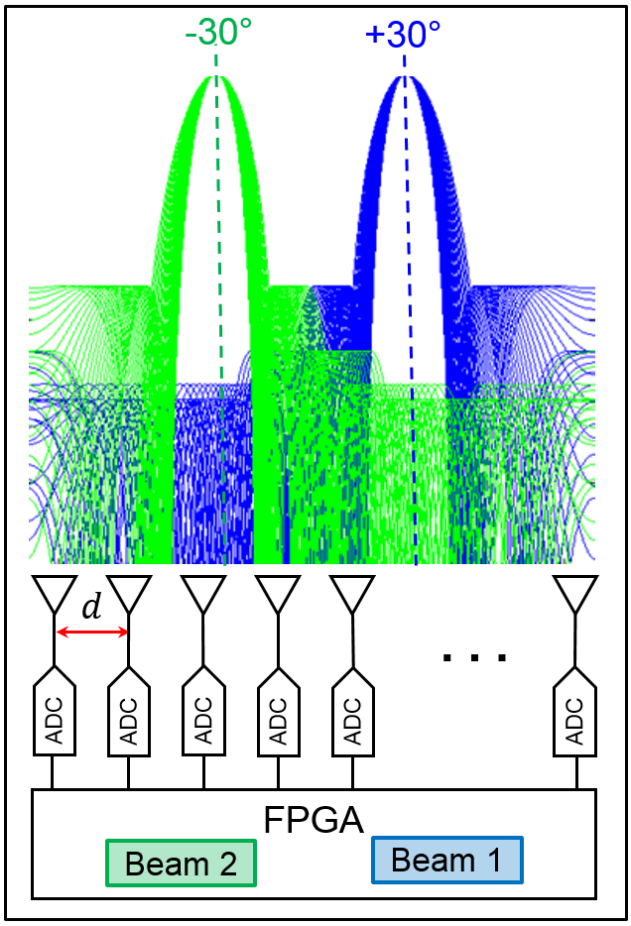

Figure 4.6: Two wideband receive beams pointed to -30 and 30 degrees, respectively.

technique effectively requires multiple beam patterns to be synthesized, one towards the angle of interest and one towards the angle of the interference where the synthesized beams are subtracted to negate the contribution from the interferer. Additionally, adaptive filters require a matrix inversion operation which prevents the implementation on an FPGA. For large arrays or large training data sets, the invertible matrix can be large and thus, require off-chip memory or significant on-chip memory resources. The streaming nature of the FPGA does not map these memory accesses in an efficient way.

The GSC is defined in the subspace of the beamspace problem where the power of an interference channel is minimized while including linear constraints and a quadratic constraint to preserve the power from the main lobe of the high-gain antenna [128]–[131]. In this scenario, a wideband jammer with variable JSR is denying the main beam spectrum, as shown in Fig. 4.7. The generalized sideband canceller (GSC) minimizes interference

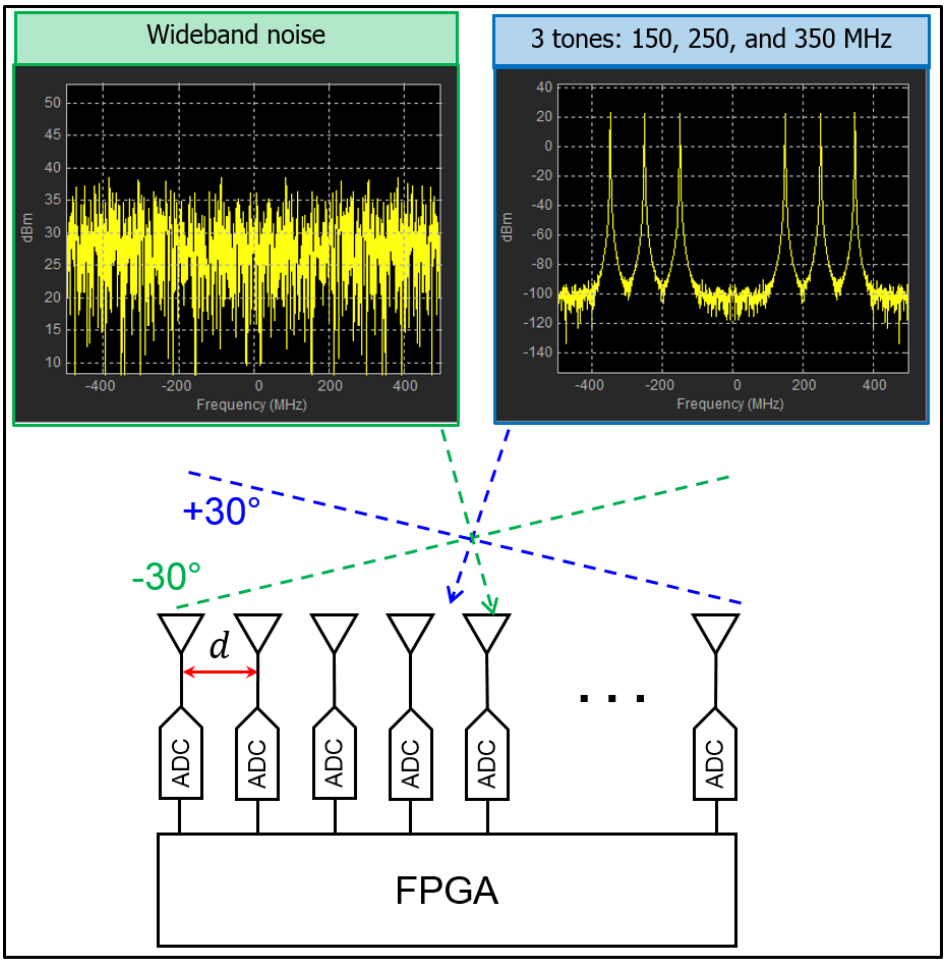

Figure 4.7: Input signal spectrum of 3 tone test signal and GSC beam output for JSR = 20 dB.

contributions from the antenna sidelobes by subtracting an interference beam from the main beam. Beamforming typically relies on spatial filtering

techniques to produce the desired beam pattern though energy from interferers can be picked up by the anteanna sidelobes. The notion of a GSC has been investigated to remove these undesired contributions of the interferers. In the GSC, a separate beam is steered at an angle towards the interferer and subtracted from the main beam. A requirement of the GSC that fails with the approach of true time delay beamforming is that the SOI be completely absent from the sidelobe cancellation channel. This is difficult to achieve in practice since the SOI can be located in a sidelobe of the cancellation beam. Thus, the cancellation procedure becomes a circular operation in which the sidelobe beam channel requires a spatial null towards the angle of the SOI. If this is achievable, then the main beam channel could, in fact, null the interferer in beam space in the first place, eliminating the need of a GSC channel. Additional cancellation, however, can be achieved by the GSC when used in conjunction with other techniques. To demonstrate the GSC operation, a Simulink simulation has been developed to analyze the contributions of interference while beamforming. The dataflow graph of the GSC is shown in Fig. 4.8. A more detailed view is shown in Fig. 4.9. The beam spectrum using native TTD and GSC is shown in Fig. 4.10 for differing JSR values. For this particular case, the GSC does not provide any additional cancellation of the jammer. In fact, energy in the sidelobes remains pointed at the adjacent source in both beam cases. Since the GSC beam contains sidelobe contributions from the signal of interest, the GSC is shown to be detrimental when reconstructing the main beam. The simulation has been configured for a linear array of 32 elements and where the sampling frequency is 1 GHz. The signal spectrum is shown for real IF signals centered at 250 MHz. Two separate beams (SOI and jammer) are generated in the

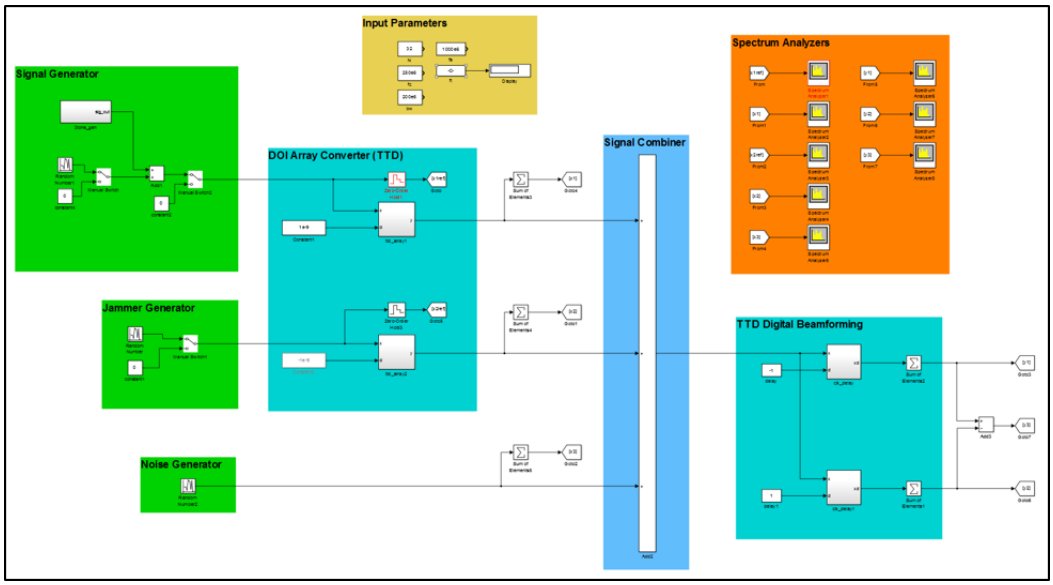

Figure 4.8: GSC architecture in high level Simulink.

continuous time domain then steered to separate angles, then combined and sampled. The SOI is made of 3 frequency components (150 MHz, 250 MHz, and 350 MHz) and the jammer has a noise bandwidth of the entire sampling bandwidth. Fig. 4.7 show the frequency spectrum of the SOI and jammer, respectively. Using TTD, the SOI and jammer are steered to -30 degrees and +30 degrees, respectively. The two beams are combined together along with simulated thermal noise having a power level of -110 dB down from the SOI. In all, the signal seen at the digitizer is a combination of 3 separate signals: 3 tones at -30 degrees, broadband jammer at +30 degrees, and thermal noise. To reproduce the original SOI beam, TTD is used. The resulting beam spectrum is shown in Fig. 4.10 for a JSR of 15 dB and 20 dB, respectively. Additionally, TTD beamforming is used to steer a beam towards the jammer implementing a channel in the GSC. The output of the GSC for a JSR of 15 dB and 20 dB, respectively, is also shown. In the GSC, the SOI beam is

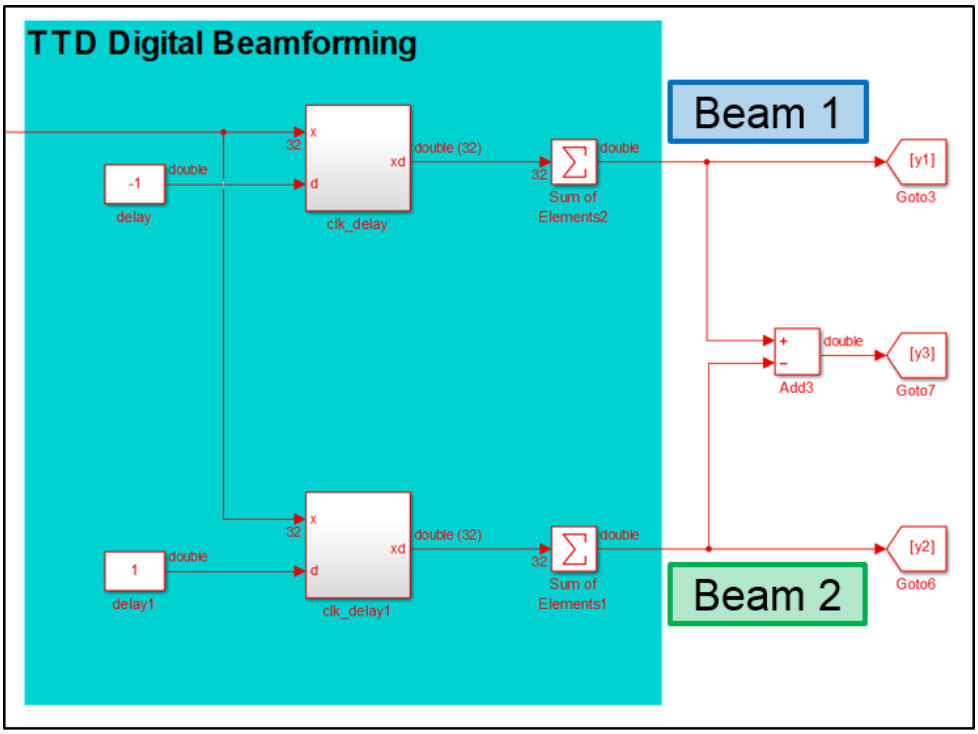

Figure 4.9: Two wideband beamformer and canceller design in high level Simulink.

subtracted from the jammer beam. It is obvious that in this particular case, a lower JSR gives better SNR performance in both the TTD and GSC. It is also evident that native TTD beamforming has performed well, recovering roughly 65 dB of SNR compared to the original SNR of 110 dB. The contribution of the jammer coming in from a different azimuth angle can be identified by the parabolic noise floor shape. This shows that the TTD beamformer can reconstruct the original SOI but does not suppress the jammer well. The GSC on the other hand recovers about 15 dB of SNR. Here, adding additional cancellation channels in the form of a GSC to suppress jammers does not warrant the use of additional processing resources added by the GSC.

Beamspace nulling and frequency based adaptive beamforming [99], [102],

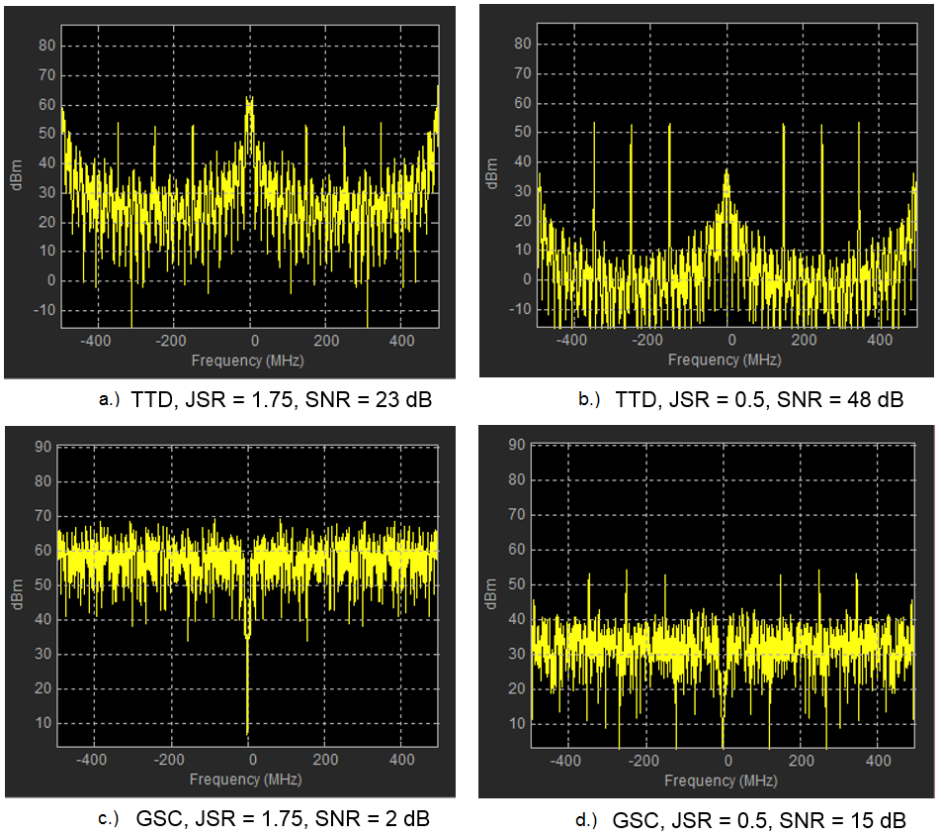

Figure 4.10: In a.) and b.) TTD beam output for JSR = 15 and 20 dB and in c.) and d.) GSC beam output for JSR = 15 and 20 dB.

[104], [105] may require many more resources than time and frequency domain implementations in some circumstances. However, the technique described in this chapter is not affected by array size too much. Using the combined output of 2 beams, Fig. 4.11 shows the adapted response with a single temporal sample (e.g. $N = 1$).

The simulated array factor across a wide operation frequency is shown in Fig. 4.12 for $M > 1$. Furthermore, the beamspace covariance matrix is constructed so that the input vector is created by 2 beams rather than 16 elements. In this way, the algorithm scales with B and not M. The main beam is steered to +45 degrees and a deep null is shown at -45 degrees. Neither an

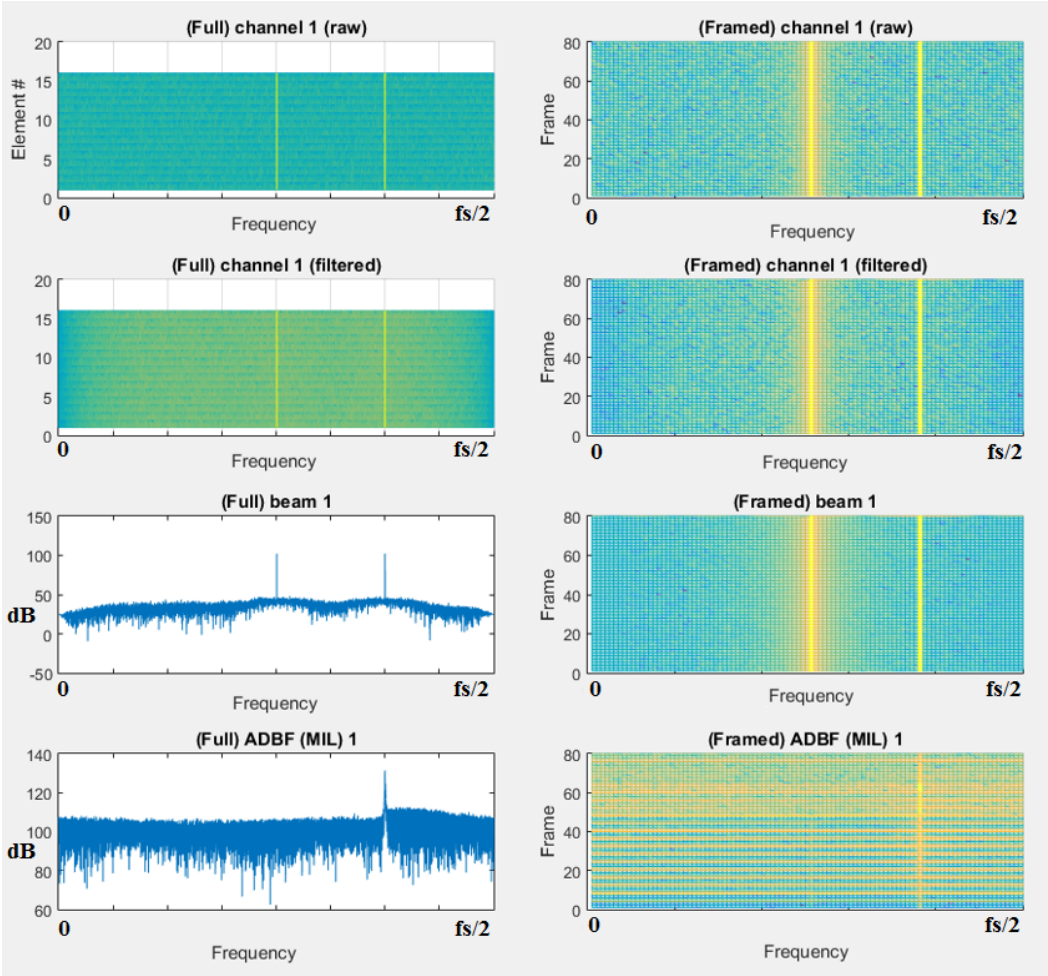

Figure 4.11: Frequency responses of the 2-channel null canceller. A strong interferer in the middle of the band has been removed the by adaptive processor.

interference signal nor it's statistics were injected into the test case as in [132], [133]. The null was solely steered using knowledge of the nulling direction which can limit the practicality of other ADBF technique. A quadratically constrained (LCMP-QC) algorithm [114] was chosen to combine the data from the sum and sidelobe canceller channels to mitigate the interference.

4.3 Chapter Summary

This chapter has shown a scalable hardware approach to implement digital, wideband ADBF techniques in real time. A significant bottleneck in ADBF systems is inverting the covariance matrix, the size of which depends on the size of the array and instantaneous bandwidth requirements. In the beamspace, the size of the covariance matrix is minimized so that the DBF performance and latency across wide bands is improved. Hardware implementations of the GSC and beamnulling techniques have been shown to form beams and nulls in real time efficiently. Feed-forward methods of computing an estimate of the covariance are compared to traditional matrix inversion algorithms.

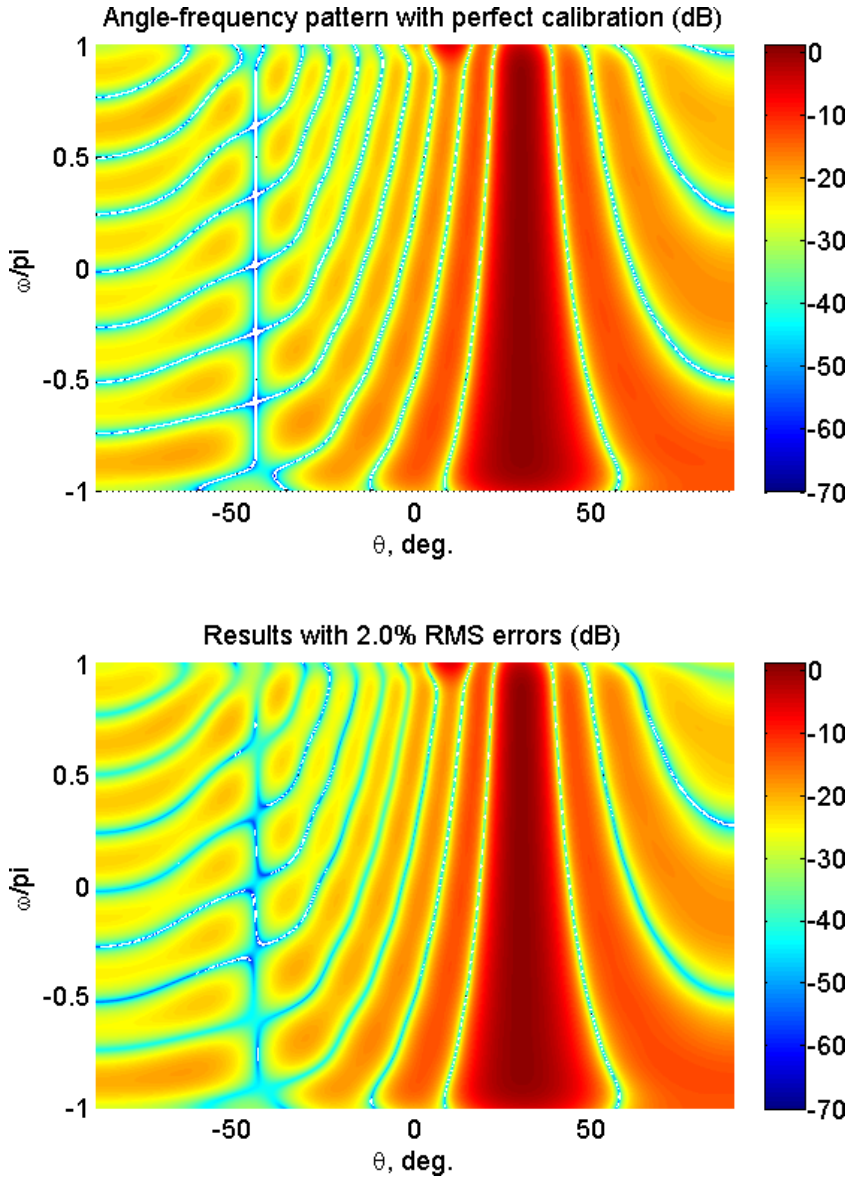

Figure 4.12: Angle-frequency AF pattern of a 16 element array with a.) perfect calibration, and b.) 2% RMS phase errors.

Chapter 5

Benchmarking Digital Beamforming Processors

While the first several chapters focused on the narrowband and wideband beamforming, which are the primary foci of this dissertation, this chapter presents highly relevant implementation topics. A number of benchmarking studies have been done previously which directly compare FPGAs to GPUs vs CPUs such as in [134]. This study and many like it however, did not consider the practicalities of a radar application, such as the time to make a data transfer (i.e. latency) and may be unsuitable to analyze real-time streaming digital beamforming (DBF) radar processors. The IEEE HPEC conference hosted by Lincoln Labs has a high performance embedded computing (HPEC) set of benchmarks, but these kernels fail to address a direct comparison of fixed vs floating point precision using the same algorithm on the FPGA and GPU. The PERFECT program set of benchmarks studies the energy efficiency, but does not directly address the floating point FPGA analysis provided here. We believe the study provided in this paper is unique in that it is the first to provide a direct comparison of an algorithm written in the same precision and run in 14nm/16nm 3D transistors on both GPU and FPGA hardware.

5.1 Digital Processor Precision

In the past, high speed FPGA processing was limited to fixed point calculations. It is possible to build floating point ALUs in the FPGA fabric, however, this is not an ideal solution due to the extraneous amount of resources required for many bits (32 or 64 bits). Such structures also fail to utilize the fast and highly efficient routing fabric offered on the FPGA, considerably limiting the clock speed of the structures. Recently, with the introduction of the 20 nm Arria 10 devices from Altera, hard floating point ALU's can be constructed that overcome these issues, offering ASIC-like performance at or near the performance of the traditional fixed point units. This new architecture enables stream processing of traditional mathematical operations that require the precision of floating point operations (i.e. matrix inversion) which was previously unavailable.

One of the key differences between processing on FPGAs versus CPUs and GPUs is the ability to very finely tune bit width and data type, which we will show are critical to processing throughput and defining algorithmic requirements. The two most commonly used formats are IEEE floating point standards and fixed point (also called integer). Table 1 below shows the dynamic range difference in I/Q signal level (not power) between the different standards. Two terms are used: instantaneous dynamic range which is calculated as the ratio of the largest value to the smallest value during a computation, e.g. with no exponent change. This measures the dynamic range limitation experienced during the middle of a multiply operation such as in an fast Fourier transform (FFT), for example; when this limit is reached, saturation and non-linear outputs occur. The tunable dynamic range is the

ratio of the maximum value capable by the format to the minimum value capable by the format, e.g. allowing for exponent updates. To compare silicon resources used for each approach, a parallel multiplier (both inputs are changing) was implemented in an FPGA using logic only and only DSP multiplier units (max DSP usage). Because the floating point standard is the same bit width on the input and output, we also matched the input bit width and output bit width to compute the resources for fixed point in the table.

Looking at Table 5.1, an interesting result emerges that is often overlooked: fixed point has more instantaneous dynamic range for a given operation and number of bits than floating point. Using 16-bits, fixed point gives 90 dB dynamic range vs float which allows 66.2 dB of dynamic range. However, over the course of many iterative computations, such as performing a multiply and accumulation of the results thousands of times where the floating point exponent can be updated in between operations, the total algorithmic dynamic range is more for floating point. Note that fixed point can in some cases be renormalized to achieve the same effect, but this is often a manual step whereas exponent updating is often automatically done in floating point IP.

Table 5.1: Dynamic range difference between fixed point and floating point in the Xilinx Zynq Ultrascale+.

Precision:	Instantaneous Dynamic Range (dB)	Tunable Dynamic Range (dB)	Resource Usage (LUTs)	Resource Usage (DSPs)
16-Bit Integer	90	N/A	280	1
32-Bit Integer	186	N/A	1088	4
64-Bit Integer	379	N/A	4256	16
16-Bit Float	66	90	192	2
32-Bit Float	144	765	682	3
64-Bit Float	319	6,159	2,444	11

5.2 Beamforming Implementations in Hardware

The DBF algorithms discussed in this dissertation were designed and tested in the Matlab/Simulink programming environment using the DSPBuilder library provided by Altera. This tool provides bit accurate simulations that compile graphical Simulink code into behavioral hardware description language (HDL). The generated HDL code is compiled into a bit-file using the Altera Quartus II tool after synthesis and place and route. A functional block diagram of the

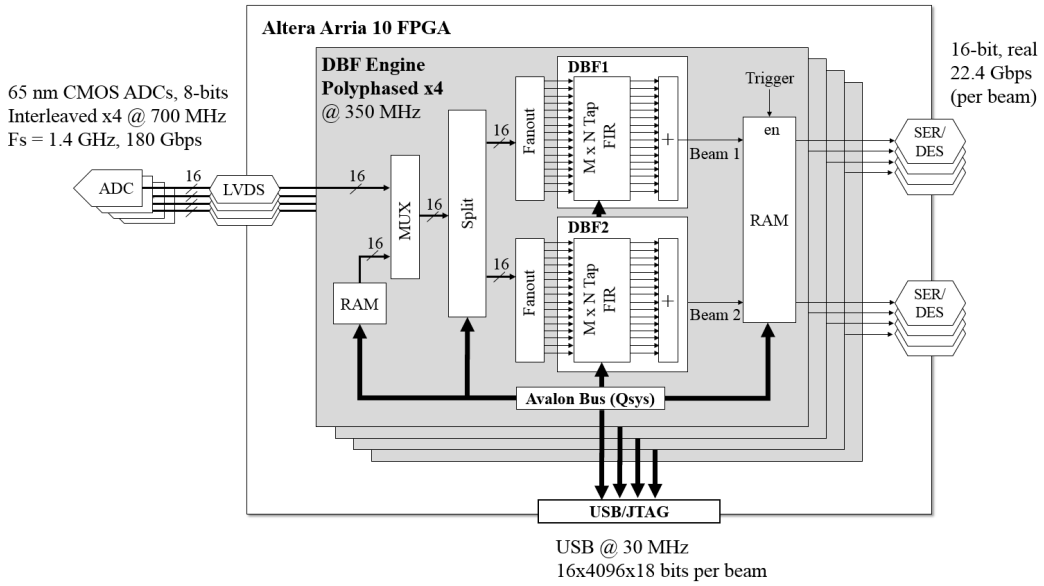

Figure 5.1: High level data flow graph of the FPGA receiver design.

FPGA design is shown in Fig. 5.1. The Arria 10 data clock operates at 350 MHz and supports four polyphased channels forming a 1.4 GHz operational sampling rate for 16 channels. Input to the DBF engine is multiplexed from two sources, the ADC LVDS interface and Matlab generated, 8-bit fixed point data that is stored and recalled to and from the left side RAM in Fig. 5.1 which serves as debugging and emulation port. The output of the MUX is split into 2 data streams which each drive an independent, simultaneous DBF

engine. A single DBF engine is composed of a matrix of chained multiply and accumulator resources that make up a MxP tap filterbank. The outputs of the filter rows are then summed to form a beam.

The streaming output of each of the beamformers is captured in the FPGA BRAMs once triggered by software. A Matlab GUI controls and manages the IMPACT module during runtime. Panels within the GUI configure the SiGe, ADC, and FPGA components over the USB/JTAG interface from a personal computer (PC) in a system-in-the-loop capability. Test and verification waveforms, filter responses, and captured data are displayed in the GUI. Once the hardware is configured, the data capture can be triggered via TCL and the data will be offloaded into Matlab to be processed and displayed within the GUI. The capture RAM depth supports 2 output beam channels of 65 μs each or 16 element-level channels of 12 μs each resulting in a depth of 4096 samples per channel which has been found to be a suitable training length required for the EQ routine discussed in Section II. Data captured in the RAMs is transferred from the FPGA into MATLAB over the Altera proprietary Avalon Bus. Altera provides drivers integrated into the MATLAB environment known as System Console which includes tool command language (TCL) scripts that provide read and write transactions over the USB/JTAG bus and runs at 30 MHz. The hardware utilization of a two channel adaptive wideband beamformer compiled to the Altera Stratix V FPGA is shown in Fig. 5.2. The figure depicts an area of 40 mm by 40 mm that uses a 28 nm TSMC process technology. Floorplaning was done to minimize the total interconnect lengths and the optimize resource utilization. The Altera Stratix V FPGA used here has heterogeneous resources consisting of columns of configurable logic blocks, RAM blocks, and multiplier blocks,

Table 5.2: FPGA resource utilization of a two wideband beamforming design with null canceller on the Altera Stratix V and Arria 10 FPGAs.

Family	Altera Stratix V	Altera Arria 10
Device	5SGXEA7K2F40C2N	10AX115S2F45I2SG
Clock Speed (MHz)	250	350
Logic (in ALMs)	22,153 / 172,600 (13%)	22,798/427,200 (5%)
DSPs	664 / 1,590 (42%)	1,405/1,518 (93%)
Throughput (GMACs)	332	980
DSP Power	20 W	11 W

as seen in Fig. 5.2. The timing requirements were met for a 250 MHz clock.

The Arria 10 implementation of the DBF filterbank design supports re-loadable coefficients and utilizes 5% of the available LUTs, represented by the medium shade of blue in Fig. 5.3. The darker shade of blue is more scarce and shows LUTs that are more densely packed. Overall, the design was able to access 96% of the DSPs requiring a large but sparse fanout of the logic. The timing was met for 350 MHz clock and digitally supports 1.4 GHz of bandwidth. However, the Arria 10 design was routed to a max clock rate of 375 MHz, enabling a 1.5 GHz of bandwidth. The power of the FPGA portion of DBF engine itself (excluding I/O) is determined by the difference of measuring the power of an idle on power up design and the power after loading the DBF engine design bitfile which was found to be 11 W. Details of the Stratix V and Arria 10 FPGA design implementations are summarized in Table 5.2.

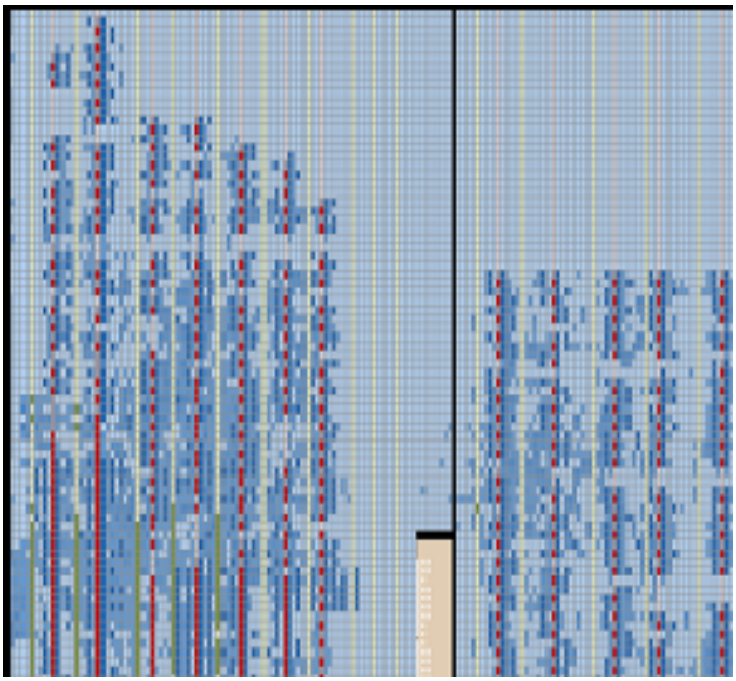

Figure 5.2: Floorplan routing layout for Altera Stratix 5 FPGA of a 2 channel wideband beamformer with GSC.

5.3 The FFT Kernel

In this section, we use a comparable processing node and similar software approach to benchmark the throughput and efficiency of the industry's latest CPU, GPU, and FPGA devices. Kernels useful for DBF, namely multipliers and FFTs are discussed and analyzed. The floating-point and fixed-point approaches to the kernels are compared and contrasted across the unique processing architectures. Theoretical and experimental results are shown for multiple size FFTs and the timing and power is measured accordingly.

In the frequency domain, converting the wideband problem into a narrowband problem can be done by subbanding the wideband channel with the discrete Fourier transform (DFT) (or FFT which is more commonly used in practice due to it's efficient implementation). A sample test variable, taken

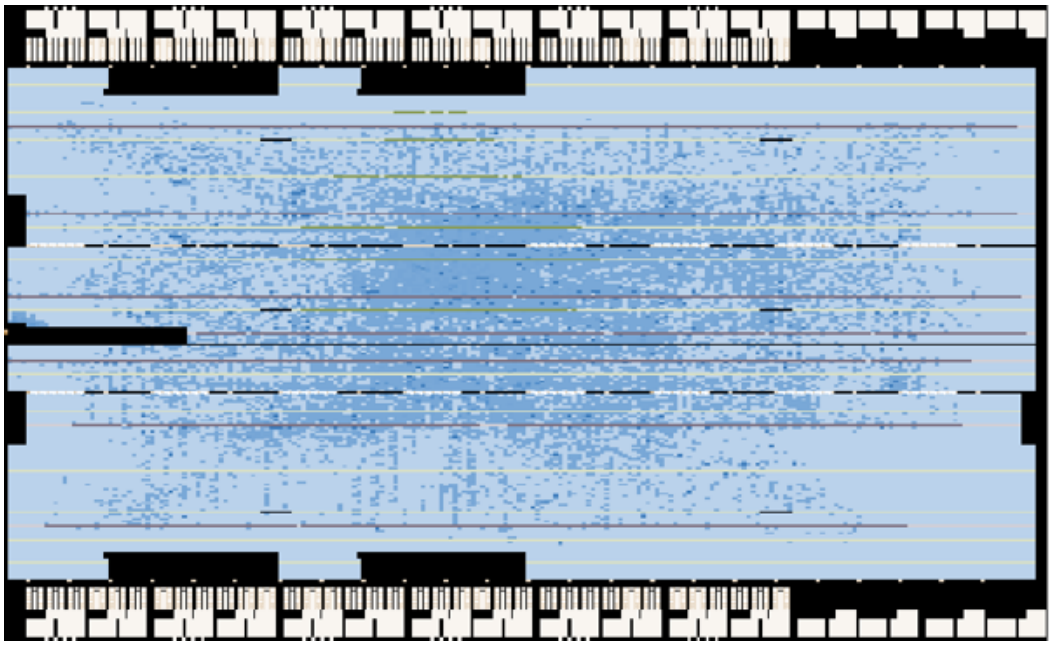

Figure 5.3: Floorplan routing layout for a two channel wideband beamformer compiled to an Altera Arria 10 FPGA.

as a noisy sinusoid, is shown in the time domain in Fig. 5.4 and the frequency domain in Fig. 5.5 after the FFT is applied.

In a beamformer, a vector matrix multiply (VMM) can be used to generate multiple simultaneous beams using phase shift only method with arbitrary independent beam positions. A vector of input signals is digitized at each element, we call this vector of signal inputs, S. We can create multiple simultaneous beams by having a matrix of complex phase shifts, P, which is of size N x B where N is our number of elements and B is our number of desired simultaneous beams. Output beams are then computed using a VMM computation of $S * P$. This algorithm has a computational complexity of $O(N * B)$. However, if equally spaced beams in space are desired the computational complexity can be greatly reduced by using the FFT. The computational complexity of the FFT is $O(N + BlogB)$, greatly reducing

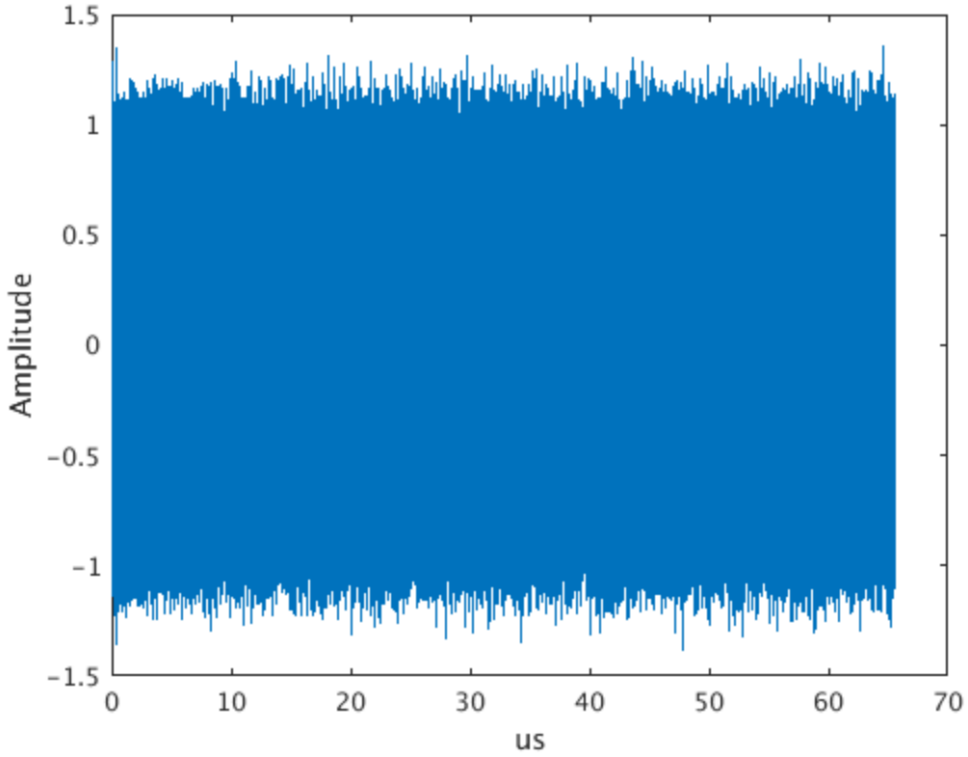

Figure 5.4: Frequency domain representation of a single tone in noise.

the complexity for large N. The output of an 8-element FFT beamformer is shown in Fig. 5.6.

5.4 Benchmarking Heterogeneous Processors

Two test systems were available to benchmark the processors and verify the precision metrics; the Nvidia DGX1 and the Xilinx MPSoC evaluation board. The associated processors are summarized in Table 5.3. For the FPGA, the Xilinx floating point core maximum speed of 544 MHz was found to consume 42 W corresponding to the power consumption from the Xilinx power calculation spreadsheet. We have found that the routable clock speed when using HDL Coder is lower in ES1 devices compared to ES2 due to -1 and -2 speed grades,

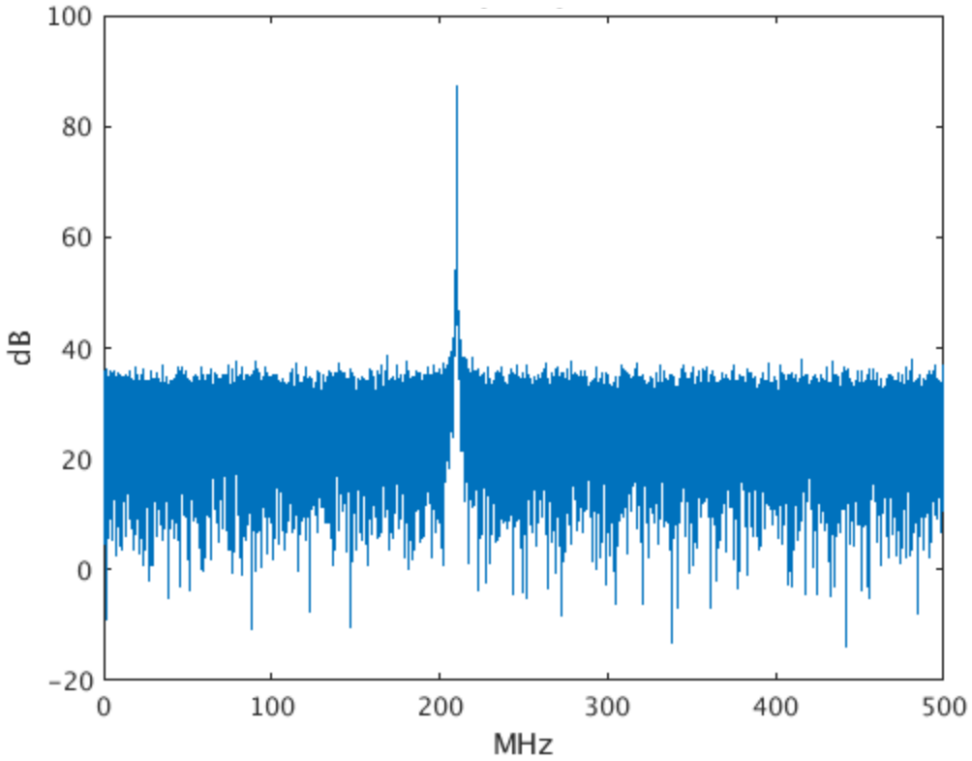

Figure 5.5: Time domain representation of a single tone in noise with low SNR ratio.

respectively. The clock speed of 300 MHz is used in the measured FPGA results and is assumed to be routable in an ES2 device. The number of CPU cores was derived from 20 physical cores capable of executing 32 single precision FLOPs per cycle. The theoretical single and fixed-point throughput shown for the CPU and GPU is the same due to the availability of a fused MAC operation. In the FPGA, 2 DSP48s are used to perform a single precision operation which nearly doubles the fixed point MAC throughput when compared to floating point.

The Matlab FFTW function is the FFT kernel used on the CPU. The FPGA uses Simulink/HDL Coder to compile a streaming version of the Matlab FFT. Generated HDL is inserted into a Xilinx Vivado project where

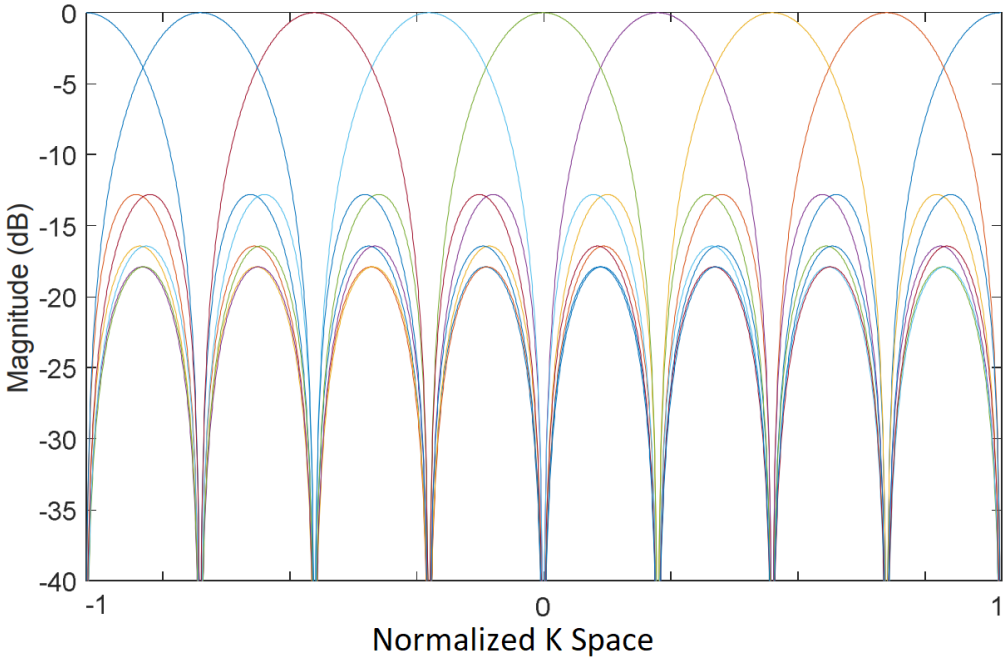

Figure 5.6: Array factor of overlaid multiple simultaneous beams for an 8 point FFT used for DBF in an 8-element array. Reprinted from Thompson et al. (2017) © 2017 IEEE.

a bitstream is generated and tested on hardware through the JTAG/AXI interface. The GPU is programmed in CUDA and compiled with the Nvidia CUDA compiler: *nvcc*. Latency is calculated using the tic and toc Matlab timers in the CPU and by using the CUDA timers in the GPU. For the GPU, the latency does not include any I/O latency to send data to and from the CPU. Clock speed and number of cycles taken to run the FFT kernel is used to determine the FPGA latency. The number of operations performed by the FFT kernel is calculated from the FFTW definition of FLOPs. For the size of FFT, N, the number of FLOPs is

$$FLOPS = 5/2Nlog_2(N)/Execution\ Time \tag{5.1}$$

Table 5.3: Specification summary of processors under test for benchmarking. The Nvidia DGX1 and Xilinx MPSoC platforms support FPGA, GPU, and CPU hardware from comparable CMOS process nodes.

Device:	Intel Xeon E5-2698 v4 (DGX1)	Nvidia Tesla GP100 (DGX1)	Xilinx Zynq US+ ZU9EG (MPSoC)
Type	CPU	GPU	FPGA
Clock Speed	2.2 GHz	1.3 GHz	544 MHz
Cores	640	3584	2520
Process	14 nm Intel	16 nm TSMC	16 nm TSMC
Memory	(50 MB L2, 256 GB DDR4)	(4 MB L2,	(4 MB BRAM,
Max Power	135 W	300 W	42 W
Theoretical Double Throughput (GFLOPS)	1500	5300	415
Theoretical Single Throughput (GFLOPS)	3000	10600	830
Theoretical Fixed-Point Throughput (GMACS)	3000	10600	1797

for real FFTs and

$$FLOPS = 5Nlog_2(N)/Execution\ Time \tag{5.2}$$

for complex FFTs. The throughput is taken as an average of many trials (typically more than 10K). Throughput of the FFT kernel is shown in Fig. 5.7. We assume that the FPGA DSP resources are loaded to 80% and run at 300 MHz. Processing efficiency (peak and measured) is shown in Fig. 5.8. When

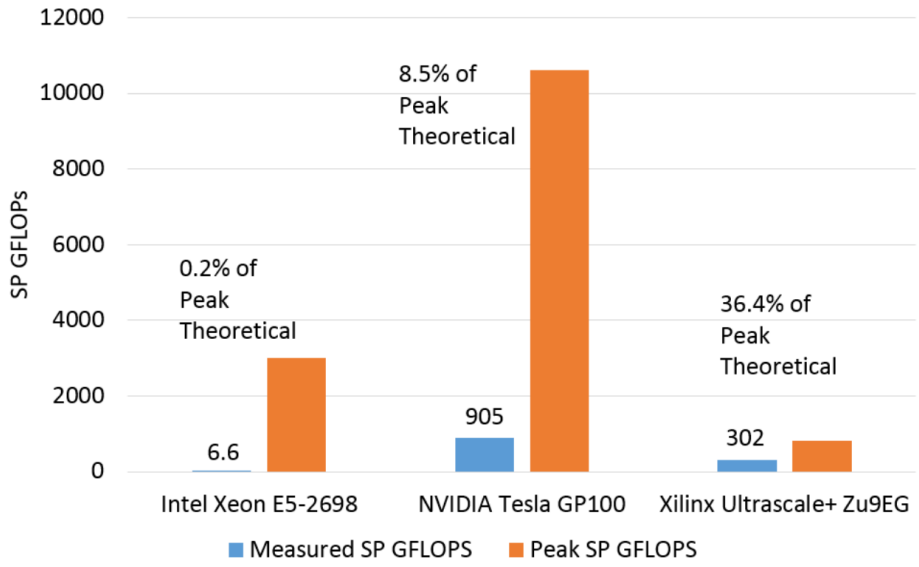

Figure 5.7: Measured vs. theoretical single precision (SP) FLOP throughput. Reprinted from Thompson et al. (2017) © 2017 IEEE.

only counting the active power of the multiply units of the GPU and FPGA, they were 238 GFLOPS/W and 245 GMACS/W respectively. The FPGA DSPs were 8x more power efficient than the logic instantiated multipliers at 245 GMACs/W vs. 34 GMACs/W. The GPU was found to be 46% efficient (compared to peak theoretical) for VMMs and only 8% efficient for FFTs, whereas the FPGA was found to be nearly equally efficient for both approaches. Nvidia has developed an architecture to alleviate this issue in which a direct GPU to FPGA interface is enabled. GPUDirect intends to improve GPU IO performance by streaming data directly to the GPU over PCIe, bypassing the CPU IO bottleneck. Unfortunately, a test system supporting GPUDirect was not available during this analysis but results have been extrapolated to contrast the architectures. The result of using GPUDirect is shown in Fig. 5.9 whereby the CPU power is not included when GPUDirect is available. A 4x energy efficiency improvement is estimated when using this feature.

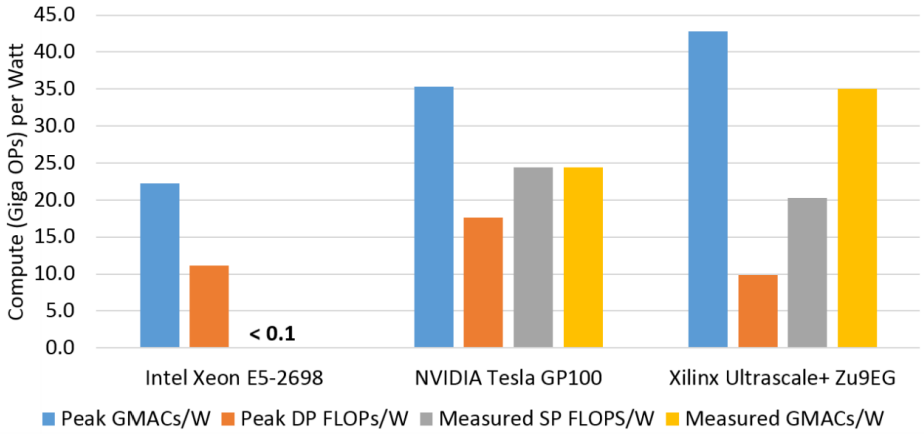

Figure 5.8: Total peak theoretical compute energy efficiency for the FFT kernel. Reprinted from Thompson et al. (2017) © 2017 IEEE.

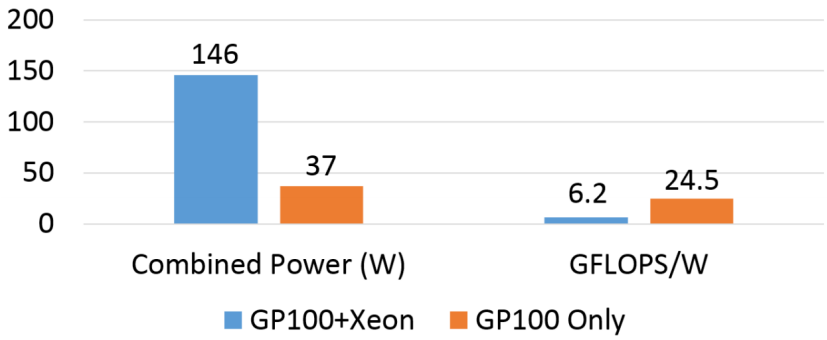

Figure 5.9: Efficiency of GPUDirect vs non-GPUDirect.

5.5 Chapter Summary

Performance and throughput of a critical DBF techniques including filtering and FFT kernels. We have found that GPU performance is the highest in all floating point cases but not fixed point. Additionally, the percentage of peak efficiency is much higher (4.6x) in the FPGA than in the GPU. Although the GPU is nearly capable of achieving Tbps throughput, it falls short of the claimed peak of 10 TFLOPs and is found to be 8.5% efficient compared to 36% in the FPGA. It should be noted that the floating point energy efficiency in the GPU is better than the FPGA but not by much. If "GMACs"

instead of "GFLOPS" can be used, an FPGA will end up being much more energy efficient in the case that lower precision is tolerable. Assuming the GPU requires some amount of IO when not using GPUDirect, the separation between FPGAs and GPUs in terms of throughput will be even higher than reported here. Finally, as the algorithms increase in complexity (e.g. more data dependencies are required), an FPGA will continue to outperform a GPU because of the ability to customize the hardware.

Chapter 6

Conclusion

Digital beamforming techniques with various applications, parameters, and requirements and were implemented in real time RF hardware. Narrowband experiments where shown to minimize multiplier operations in digital transceiver and to reduce IO by minimizing the phase shifter precision of the RF front-end. Both techniques reduce IO and dynamic power. In wideband applications, a digital TTD based algorithm was shown to beamform with better than 0.1 ps of temporal resolution. Wideband DBF experiments were shown to operate at 4.4 GHz and 200 MHz of instantaneous bandwidth and was demonstrated on the fully-calibrated Rockwell Collins IMPACT module. Finally, DBF in the beamspace is shown to syntheszie wideband nulls in an adapive beamformer, eliminating the processing complexity of inverting the covariance matrix. Each technique presented is well suited for FPGA or ASIC implementation in which digital circuits will eventually replace previously analog functionality.

The results shown here argue for the continued pursuit in the processing improvement of DBF techniques down to the transistor level. Today, most modern DBF systems contain an FPGA with access to thousands of DSPs and millions of logic gates suitable for efficient digital processing for low cost.

Such devices have become the digital "front-end" of today's radar systems and will continue to replace analog circuits until, ultimately, the digital will interface directly to the antenna.

References

[1] A. Natarajan, S. Reynolds, M. Tsai, S. Nicolson, J. Zhan, D. Kam, D. Liu, Y. Huang, A. Valdes-Garcia, and B. Floyd, "A fully-integrated 16-element phased-array receiver in SiGe BiCMOS for 60-GHz communications", *IEEE Journal of Solid-State Circuits*, vol. 46, no. 5, pp. 1059–1075, 2011.

[2] A. Valdes-Garcia, S. Reynolds, A. Natarajan, D. Kam, D. Liu, J. Lai, Y. Huang, P. Chen, M. Tsai, J. Zhan, S. Nicolson, and B. Floyd, "Single-element and phased-array transceiver chipsets for 60-GHz Gb/s communications", *IEEE Communications Magazine*, vol. 49, no. 4, pp. 120–131, 2011.

[3] J. Karjalainen, M. Nekovee, H. Benn, W. Kim, J. Park, and H. Sungsoo, "Challenges and opportunities of mm-wave communication in 5G networks", in *International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, 2014, pp. 372–376.

[4] T. Kim, J. Park, J. Seol, S. Jeong, J. Cho, and W. Roh, "Tens of Gbps support with mmwave beamforming systems for next generation communications", in *IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 3685–3690.

[5] M. Fakharzadeh, M. Nezhad-Ahmadi, B. Biglarbegian, J. Ahmadi-Shokouh, and S. Safavi-Naeini, "CMOS phased array transceiver technology for 60 GHz wireless applications", *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 4, pp. 1093–1104, 2010.

[6] E. Cohen, M. Ruberto, M. Cohen, O. Degani, S. Ravid, and D. Ritter, "A CMOS bidirectional 32-element phased-array transceiver at 60 GHz with LTCC antenna", *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 3, pp. 1359–1375, 2013.

[7] W. Roh, J. Seol, J. Park, B. Lee, J. Lee, Y. Kim, J. Cho, K. Cheun, and F. Aryanfar, "Millimeter-wave beamforming as an enabling technology for 5G cellular communications: Theoretical feasibility and prototype results", *IEEE Communications Magazine*, vol. 52, no. 2, pp. 106–113, 2014.

[8] G. Rebeiz, "Millimeter-wave large-scale phased-arrays for 5G systems", in *IEEE MTT-S International Microwave Symposium*, Phoenix, AZ., 2015, pp. 1–3.

[9] J. Lopez-Sanchez, I. Hajnsek, and J. Ballester-Berman, "First demonstration of agricultural height retrieval with PolInSAR airborne data", *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 2, pp. 242–246, 2012.

[10] R. Wang, W. Wang, Y. Shao, F. Hong, P. Wang, Y. Deng, Z. Zhang, and O. Loffeld, "First bistatic demonstration of digital beamforming in elevation with TerraSAR-X as an illuminator", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 2, pp. 842–849, 2016.

[11] R. Rincon, M. Vega, M. Buenfil, A. Geist, L. Hilliard, and P. Racette, "NASA's L-band digital beamforming synthetic aperture radar", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 10, pp. 3622–2628, 2011.

[12] N. Gebert, F. Almeida, and G. Krieger, "Airborne demonstration of multichannel SAR imaging", *IEEE Geoscience Remote Sensing Letters*, vol. 8, no. 5, pp. 963–967, 2011.

[13] W. Yiding, "The maximum phase error of a reflected signal in an active coded transponder", *IEEE Geoscience Remote Senssing Letters*, vol. 3, no. 1, pp. 150–153, 2006.

[14] M. Suess, M. Ludwig, C. Schaefer, and M. Younis, "Technology developments for the next generation of spaceborne SAR instruments based on digital beamforming", in *Proceedings of Geoscience and Remote Sensing Society (IGARSS)*, 2012, pp. 1529–1532.

[15] P. 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.

[16] D. Yu, C. Zhao, and W. Xiang, "Beam position agility in VPRF for spaceborne precipitation radar", *IEEE Geoscience Remote Sensing Letters*, vol. 10, no. 2, pp. 256–259, 2013.

[17] B. Johnson and S. Rani, "A high throughput fully parallel-pipelined FPGA accelerator for dense cloud motion analysis", in *IEEE Region 10 Conference (TENCON)*, Singapore, Singapore, 2016, pp. 2589–2592.

[18] A. George, J. Garcia, K. Kim, and P. Sinha, "Distributed parallel processing techniques for adaptive sonar beamforming", *Journal of Computational Acoustics*, vol. 10, no. 01, pp. 1–23, 2002.

[19] B. Kim and I. Lu, "High resolution broadband beamforming based on the MVDR method", in *IEEE Conference and Exhibition*, vol. 3, 2000, pp. 1673–1676.

[20] J. Chen, A. Yu, and H. So, "Design considerations of real-time adaptive beamformer for medical ultrasound research using FPGA and GPU", in *International Conference on Field-Programmable Technology*, Seoul, South Korea, 2012, pp. 198–205.

[21] W. Weedon and R. Nunes, "Low-cost wideband digital receiver/exciter (DREX) technology enabling next-generation all-digital phased arrays", in *International Phased Array Radar Conference*, Boston, MA, 2016, pp. 1–3.

[22] W. Chappell and C. Fulton, "Digital array radar panel development", in *International Symposium on Phased Array Systems and Technology*, Waltham, MA, 2010, pp. 50–60.

[23] M. Skolnik, *Radar Handbook, Third Edition*. McGraw-Hill Education, 2008.

[24] S. Weiss and I. Proudler, "Comparing efficient broadband beamforming architectures and their performance trade-offs", in *International Conference on Digital Signal Processing Proceedings*, vol. 1, 2002, 417–423 vol.1.

[25] W. Liu and S. Weiss, *Wideband Beamforming: Concepts and Techniques*. West Sussex, UK: Wiley, 2010.

[26] R. Singh and S. Sapre, *Communication Systems, Second Edition*. McGraw-Hill Education, 2009.

[27] L. Pettersson, M. Danestig, and U. Sjostrom, "An experimental S-band digital beamforming antenna", *IEEE Aerospace and Electronic Systems Magazine*, vol. 12, no. 11, pp. 19–29, 1997.

[28] H. Steyskal, "Digital beamforming antennas - an introduction", *Microwave Journal*, vol. 30, p. 107, 1987.

[29] S. Salivahanan, A. Vallavaraj, and C. Gnanapriya, *Digital signal processing*. New Delhi: Tata McGraw-Hill, 2000.

[30] M. Longbrake, "True time-delay beamsteering for radar", in *IEEE National Aerospace and Electronics Conference (NAECON)*, 2012, pp. 246–249.

[31] M. Rosker, C. Bozada, H. Dietrich, H. Hungand, D. Via, S. Binari, E. Vivierios, E. Cohen, and J. Hodiak, "The DARPA wide band gap semiconductors for RF applications (WBGS-RF) program: Phase II results", Tampa, FL, 2009.

[32] M. Jahn, R. Feger, C. Wagner, Z. Tong, and A. Stelzer, "A four-channel 94-GHz SiGe-based digital beamforming FMCW radar", *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 3, pp. 861–869, 2012.

[33] A. Hajimiri, H. Hashemi, A. Natarajan, X. Guan, and A. Komijani, "Integrated phased array systems in silicon", *Proceedings of the IEEE*, vol. 93, no. 9, pp. 1637–1655, 2005.

[34] I. Sever, S. Lo, S. Ma, P. Jang, A. Zou, C. Arnott, K. Ghatak, A. Schwartz, L. Huynh, and T. Nguyen, "A dual-antenna phase-array ultra-wideband CMOS transceiver", *IEEE Communications Magazine*, vol. 44, no. 8, pp. 102–110, 2006.

[35] S. Jeon, Y. Wang, H. Wang, F. Bohn, A. Natarajan, A. Babakhani, and A. Hajimiri, "A scalable 6-to-18 GHz concurrent dual-band quad-beam phased-array receiver in CMOS", *IEEE Journal of Solid-State Circuits*, vol. 43, no. 12, pp. 2660–2673, 2008.

[36] T. Hoffmann, C. Fulton, M. Yeary, A. Saunders, D. Thompson, B. Murmann, B. Chen, and A. Guo, "Measured performance of the IMPACT common module - a building block for next generation phase arrays", in *IEEE International Symposium on Phased Array Systems and Technology (PAST)*, 2016, pp. 1–7.

[37] B. Vaz, A. Lynam, B. Verbruggen, A. Laraba, C. Mesadri, A. Boumaalif, J. Mcgrath, U. Kamath, R. Torre, A. Manlapat, D. Breathnach, C. Erdmann, and B. Farley, "A 13b 4GS/s digitally assisted dynamic 3-stage asynchronous pipelined-SAR ADC", in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 276–277.

[38] Y. Voronenko and M. Puschel, "Mechanical derivation of fused multiply-add algorithms for linear transforms", *IEEE Transactions on Signal Processing*, vol. 55, no. 9, pp. 4458–4473, 2007.

[39] A. Dempster and M. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, no. 9, pp. 569–577, 1995.

[40] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Multiple tunable constant multiplications: Algorithms and applications", in *IEEE Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 473–479.

[41] ——, "Optimization of area and delay at gate-level in multiple constant multiplications", in *Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, 2010, pp. 3–10.

[42] ——, "Design of low-complexity digital finite impulse response filters on FPGAs", in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2012, pp. 1197–1202.

[43] M. Potkonjak, M. Srivastava, and A. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 2, pp. 151–165, 1996.

[44] ——, "Efficient substitution of multiple constant multiplications by shifts and additions using iterative pairwise matching", in *Conference on Design Automation*, 1994, pp. 189–194.

[45] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.

[46] C. Fulton, M. Yeary, D. Thompson, J. Lake, and A. Mitchell, "Digital phased arrays: Challenges and opportunities", *Proceedings of the IEEE*, vol. 104, no. 3, pp. 487–503, 2016.

[47] H. Hashemi and H. Krishnaswamy, "Challenges and opportunities in ultra-wideband antenna-array transceivers for imaging", in *IEEE International Conference on Ultra-Wideband*, 2009, pp. 586–591.

[48] J. Herd, S. Duffy, D. Carlson, M. Weber, G. Brigham, C. Weigand, and D. Cursio, "Low cost multifunction phased array radar concept", in *IEEE International Symposium on Phased Array Systems and Technology*, 2010, pp. 457–460.

[49] C. Lai, K. Tan, Y. Chen, and T. Chu, "A UWB impulse-radio timed-array radar with time-shifted direct-sampling architecture in 0.18-um CMOS", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 7, pp. 2074–2087, 2014.

[50] T. Kanar, S. Zihir, and G. Rebeiz, "A 2-15 GHz accurate built-in-self-test system for wideband phased arrays using self-correcting eight-state I/Q mixers", *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 12, pp. 4250–4261, 2016.

[51] M. Yeary, W. Zhang, and J. Trelewicz, "A computationally efficient decimation filter design for embedded systems", in *IEEE Instrumentation and Measurement Technology Conference (IMTC)*, vol. 2, 2004, pp. 913–916.

[52] D. Thompson, M. Yeary, C. Fulton, and B. McGuire, "Optimized beam steering approach for improved sidelobes in phased array radars using a minimal number of control bits", *IEEE Transactions on Antennas and Propagation*, vol. 63, no. 1, pp. 106–112, 2015.

[53] M. Yeary, W. Zhang, J. Trelewicz, Y. Zhai, and B. McGuire, "Theory and implementation of a computationally efficient decimation filter for power-aware embedded systems", *IEEE Transactions Instrumentation and Measurement*, vol. 55, no. 5, pp. 1839–1849, 2006.

[54] M. Gately, M. Yeary, and C. Tang, "Multiple real-constant multiplication with improved cost model and greedy and optimal searches", in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2012, pp. 588–591.

[55] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[56] T. Ciloglu and Z. Unver, "A new approach to discrete coefficient FIR digital filter design by simulated annealing", in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, 1993, pp. 101–104.

[57] C. Baird and G. Rassweiler, "Adaptive sidelobe nulling using digitally controlled phase-shifters", *IEEE Transactions Antennas and Propagation*, vol. 24, no. 5, pp. 638–649, 1976.

[58] S. Smith, "Optimum phase-only adaptive nulling", *IEEE Transactions Signal Processing*, vol. 47, no. 7, pp. 1835–1843, 1999.

[59] A. Khzmalyan and A. Kondratiev, "The phase-only shaping and adaptive nulling of an amplitude pattern", *IEEE Transactions Antennas and Propagation*, vol. 51, no. 2, pp. 264–272, 2003.

[60] C. Hsu, W., and C. Chen, "Adaptive pattern nulling design of linear array antenna by phase-only perturbations using memetic algorithms", in *International Conference on Innovative Computing, Information, and Control (ICICIC)*, vol. 3, 2006, pp. 308–311.

[61] R. Guinvarc'h, R. Gillard, B. Uguen, and J. El-Khoury, "Improving the azimuthal resolution of HFSWR with multiplicative beamforming", *IEEE Geoscience Remote Sensing Letters*, vol. 9, no. 5, pp. 925–927, 2012.

[62] C. Miller, "Minimizing the effects of phase quantization errors in an electronically scanned array", *Proc. Symp. on Electronically Scanned Array Techniques and Applications*, vol. 1, pp. 17–39, 1964.

[63] M. Clenet and G. Morin, "Graphical investigation of quantisation effects of phase shifters on array patterns", Defense Research Establishment Ottawa, Tech. Rep. DREO TR 2000-092, 2000.

[64] S. Taheri and F. Farzaneh, "New methods of reducing the phase quantization error effects on beam pointing and parasitic side lobe level of the phased array antennas", in *Proceedings of Asia-Pacific Microwave Conference (APMC)*, 2006, pp. 2114–2117.

[65] H. Kamoda, J. Tsumochi, and F. Suginoshita, "Reduction in quantization lobes due to digital phase shifters for phased array radars", in *Proceedings of Asia-Pacific Microwave Conference (APMC)*, 2011, pp. 1618–1621.

[66] H. Kamoda, J. Tsumochi, T. Kuki, and F. Suginoshita, "A study on antenna gain degradation due to digital phase shifter in phased array antennas", *Microwave and Optical Technology Letters*, vol. 53, no. 8, pp. 1743–1746, 2011.

[67] B. Veen and K. Buckley, "Beamforming: A versatile approach to spatial filtering", *IEEE ASSP Magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[68] J. Piper, *Sonar Systems: Beamforming Narrowband and Broadband Signals*. InTech, 2011.

[69] Y. Gao, D. Jiang, and M. Liu, "Wideband transmit beamforming using integer-time-delayed and phase-shifted waveforms", *Electronics Letters*, vol. 53, no. 6, pp. 376–378, 2017.

[70] K. Wagner, S. Weaver, S. Kraut, L. Griffiths, and T. Weverka, "Broadband efficient adaptive method for true-time-delay array processing", in *Proceeding of the IEEE Aerospace Conference*, vol. 5, 1998, pp. 289–298.

[71] M. Burla *et al.*, "Integrated photonic Ku-band beamformer chip with continuous amplitude and delay control", *IEEE Photonics Technology Letters*, vol. 25, no. 12, pp. 1145–1148, 2015.

[72] X. Ye, Y. Zhang, and S. Pan, "Performance evaluation of RF beamforming based on a wideband antenna array and photonic true time delay", in *International Conference on Optical Communications and Networks (ICOCN)*, Nanjing, China, 2015, pp. 1–3.

[73] R. Minasian, "Ultra-wideband and adaptive photonic signal processing of microwave signals", *IEEE Journal of Quantum Electronics*, vol. 52, no. 1, pp. 1–13, 2016.

[74] R. Rotman, M. Tur, and L. Yaron, "True time delay in phased arrays", *Proceedings of the IEEE*, vol. 104, no. 3, pp. 504–518, 2016.

[75] R. Rotman and M. Tur, "Calibration of pulsed phased arrays with wide instantaneous bandwidths", in *IEEE Antennas and Propagation Society International Symposium*, 2007, pp. 121–124.

[76] H. Neoh, "Efficient broadband multibeam beamformer architecture", in *International Symposium on Phased Array Systems and Technology*, Waltham, MA, 2016.

[77] H. Hashemi, T. Chu, and J. Roderick, "Integrated true-time-delay-based ultra-wideband array processing", *IEEE Communications Magazine*, vol. 46, no. 9, pp. 162–172, 2008.

[78] Y. Yao, X. Huang, G. Wu, and K. Wei, "Joint equalization and fractional delay filter design for wideband digital beamforming", in *IEEE Radar Conference*, Arlington, VA, 2015, pp. 0823–0827.

[79] C. Cheung, R. Shah, and M. Parker, "Time delay digital beamforming for wideband pulsed radar implementation", in *IEEE International Symposium on Phased Array Systems and Technology*, Waltham, MA, 2013, pp. 448–455.

[80] H. Johansson, O. Gustafsson, K. Johansson, and L. Wanhammar, "Adjustable fractional-delay FIR filters using the Farrow structure and multirate techniques", in *IEEE Asia Pacific Conference on Circuits and Systems*, Singapore, 2006, pp. 1055–1058.

[81] C. Farrow, "A continuously variable digital delay element", Espoo, Finland, 1988, pp. 2641–2645.

[82] Mathworks. (2011). Fractional delay filters using farrow structures, [Online]. Available: `https://www.mathworks.com/help/dsp/examples/fractional-delay-filters-using-farrow-structures.html`.

[83] A. Madanayake, C. Wijenayake, S. Wijayaratna, R. Acosta, and S. Hariharan, "2-D-IIR time-delay-sum linear aperture arrays", *IEEE Antennas and Wireless Propagation Letters*, vol. 13, pp. 591–594, 2014.

[84] A. Madanayake and L. Bruton, "A speed-optimized systolic-array processor architecture for spatio-temporal 2-D IIR broadband beam filters", *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 55, no. 7, pp. 1953–1966, 2008.

[85] R. Olsson, "Reconfigurable technologies and architectures for phased array antennas: An overview of the DARPA ACT program", in *Proceedings SPIE 9479*, 2015.

[86] M. Parker. (2011). Radar basics - part 3: Beamforming and radar digital processing, [Online]. Available: `http://www.eetimes.com/document.asp?doc\_id=1278838`.

[87] V. Valimaki and T. Laakso, "Principles of fractional delay filters", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, 2000, pp. 3870–3873.

[88] T. Laakso, V. Valimaki, M. Karjalainen, and U. Laine, "Splitting the unit delay: Tools for fractional delay filter design", *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, 1996.

[89] A. Madanayake, N. Udayanga, and V. Ariyarathna, "Wideband delay-sum digital aperture using Thiran all-pass fractional delay filters", in *IEEE Radar Conference (RadarConf)*, 2016, pp. 1–5.

[90] D. Horvat, J. Bird, and M. Goulding, "True time-delay bandpass beamforming", *IEEE Journal of Oceanic Engineering*, vol. 17, no. 2, pp. 185–192, 1992.

[91] T. Chu and H. Hashemi, "A true time-delay-based bandpass multi-beam array at mm-waves supporting instantaneously wide bandwidths", in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2010, pp. 38–39.

[92] H. Aumann, A. Fenn, and F. Willwerth, "Phased array antennas calibration and pattern prediction using mutual coupling measurements", *IEEE Transactions on Antennas and Propagation*, vol. 37, no. 7, pp. 844–850, 1989.

[93] A. Mitchell, "Coupling-based wideband digital phased array calibration techniques", *M.S. Thesis, Department of Electrical and Computer Engineering, University of Oklahoma, Norman, OK*, 2014.

[94] M. Longbrake, L. Liou, D. Lin, P. Buxa, J. McCann, T. Pemberton, T. Dalrymple, and S. Hary, "Wideband phased array calibration method for digital beamforming", in *IEEE National Aerospace and Electronics Conference (NAECON)*, 2012, pp. 11–17.

[95] L. Paulsen, T. Hoffmann, C. Fulton, M. Yeary, A. Saunders, D. Thompson, B. Chen, A. Guo, and B. Murmann, "IMPACT: A low cost, reconfigurable, digital beamforming common module building block for next generation phased arrays", in *Proceedings SPIE 9479*, 2015.

[96] M. Elsallal and J. Mather, "An ultra-thin, decade (10:1) bandwidth, modular "BAVA" array with low cross-polarization", in *IEEE International Symposium on Antennas and Propagation (APSURSI)*, 2011, pp. 1980–1983.

[97] O. Frost, "An algorithm for linearly constrained adaptive array processing", *Proceedings of the IEEE*, vol. 60, no. 8, pp. 926–935, 1972.

[98] P. Vouras and T. Tran, "Wideband adaptive beamforming using linear phase filterbanks", in *Asilomar Conference on Signals, Systems and Computers*, 2006, pp. 2295–2299.

[99] S. Kalia, S. Patnaik, B. Sadhu, M. Sturm, M. Elbadry, and R. Harjani, "Multi-beam spatio-spectral beamforming receiver for wideband phased arrays", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 8, pp. 2018–2029, 2013.

[100] H. Steyskal, R. Shore, and R. Haupt, "Methods for null control and their effects on the radiation pattern", *IEEE Transactions on Antennas and Propagation*, vol. AP-34, no. 3, pp. 404–409, 1986.

[101] J. Simon and W. Cummings, "An adaptive nulling algorithm based on signal subspace concepts", in *MILCOM*, vol. 1, 1991, pp. 107–112.

[102] Y. Bresler, V. Reddy, and T. Kailath, "Optimum beamforming for coherent signal and interferences", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 6, pp. 833–843, 1988.

[103] S. Vorobyov, "Principles of minimum variance robust adaptive beamforming design", *Signal Processing*, vol. 93, no. 12, pp. 3264–3277, 2013.

[104] J. Lynch, "Low latency digital beamforming radar using aperture coding", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 2, pp. 918–927, 2016.

[105] S. Kalia, S. Patnaik, B. Sadhu, M. Sturm, M. Elbadry, and R. Harjani, "Multi-beam spatio-spectal beamforming receiver for wideband phased arrays", *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 60, no. 8, pp. 2018–2029, 2013.

[106] K. Yu and M. Fernandez, "Methods to combine deterministic nulling and adaptive nulling", in *IEEE Radar Conference (RadarConf)*, 2017, pp. 0123–0128.

[107] J. Baldwinson and I. Antipov, "Prediction of electronic attack effectiveness against maritime patrol radars", in *International Conference on Radar*, vol. 5, 2008, pp. 259–264.

[108] D. Yang, G. Peterson, H. Li, and J. Sun, "An FPGA implementation for solving least square problem", in *IEEE Symposium on Field Programmable Custom Computing Machines*, Napa, CA, 2009, pp. 303–306.

[109] M. Mueller, "Least-squares algorithms for adaptive equalizers", *The Bell System Technical Journal*, vol. 60, no. 8, pp. 1905–1925, 1981.

[110] L. Resende, J. Romano, and M. Bellanger, "A fast least-squares algorithm for linearly constrained adaptive filtering", *IEEE Transactions on Signal Processing*, vol. 44, no. 5, pp. 1168–1174, 1996.

[111] H. Brandenstein and R. Unbehauen, "Weighted least-squares approximation of FIR by IIR digital filters", *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 558–568, 2001.

[112] M. Lang, "Least-squares design of IIR filters with prescribed magnitude and phase responses and a pole radius constraint", *IEEE Transactions on Signal Processing*, vol. 48, no. 11, pp. 3109–3121, 2000.

[113] C. Burrus, "Iterative reweighted least-squares design of FIR filters", *IEEE Transactions on Signal Processing*, vol. 42, no. 11, pp. 2926–2936, 1994.

[114] P. Sinha, A. George, and K. Kim, "Parallel algorithms for robust broadband MVDR beamforming", *Journal of Computational Acoustics*, vol. 10, no. 1, pp. 69–96, 2002.

[115] A. Maltsev, V. Pestretsov, R. Maslennikov, and A. Khoryaev, "Triangular systolic array with reduced latency for QR-decomposition of complex matrices", in *IEEE International Symposium on Circuits and Systems*, 2006, pp. 385–388.

[116] F. Echman and V. Owall, "A scalable pipelined complex valued matrix inversion architecture", in *IEEE International Symposium on Circuits and Systems*, vol. 5, 2005, pp. 4489–4492.

[117] D. Boppana, K. Dhanoa, and J. Kempa, "FPGA based embedded processing architecture for the QRD-RLS algorithm", in *IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, 2004, pp. 330–331.

[118] G. Lightbody, R. Walke, R. Woods, and J. McCanny, "Novel mapping of a linear QR architecture", in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, 1999, pp. 1933–1936.

[119] S. Aslan, E. Oruklu, and J. Saniie, "Realization of area efficient QR factorization using unified division, square root, and inverse square root hardware", in *IEEE Conference on Electro/Information Technology*, Windsor, ON, 2009, pp. 245–250.

[120] Z. Liu, J. McCanny, G. Lightbody, and R. Walke, "Generic SoC QR array processor for adaptive beamforming", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 50, no. 4, pp. 169–175, 2003.

[121] G. Prabhu and S. Rani, "Fixed point pipelined architecture for QR decomposition", in *IEEE International Conference on Advanced Communications, Control and Computing Technologies*, Ramanathapuram, India, 2014, pp. 468–472.

[122] A. Irturk, "Implementation of QR decomposition algorithm using FPGAs", *M.S. Thesis, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA*, 2007.

[123] D. Kim and S. Rajopadhye, "An improved systolic architecture for LU decomposition", in *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2006, pp. 231–238.

[124] G. Govindu, V. Prasanna, V. Daga, S. Gangadharpalli, and V. Sridhar, "Efficient floating-point based block LU decomposition on FPGAs", in *ERSA*, 2004.

[125] A. Ahmedsaid, A. Amira, and A. Bouridane, "Improved SVD systolic array and implementation on FPGA", in *IEEE International Conference on Field-Programmable Technology (FPT)*, 2003, pp. 35–42.

[126] A. Irturk, B. Benson, S. Mirzaei, and R. Kastner, "An FPGA design space exploration tool for matrix inversion architectures", in *Symposium on Application Specific Processors*, Anaheim, CA, 2008, pp. 42–47.

[127] R. Schreiber, "Implementation of adaptive array algorithms", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1038–1045, 1986.

[128] K. Buckley and L. Griffiths, "An adaptive generalized sidelobe canceller with derivative constraints", *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 311–319, 1986.

[129] K. Gerlach and F. Kretschmer, "Convergence properties of Gram-Schmidt and SMI adaptive algorithms", *IEEE Trans. Aerospace and Electronic Systems*, vol. 26, no. 13, pp. 44–56, 1990.

[130] I. Cohen, "Analysis of two-channel generalized sidelobe canceller (GSC) with post-filtering", *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, pp. 684–699, 2003.

[131] L. Griffiths and C. Jim, "An alternative approach to linearly constrained adaptive beamforming", *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 1, pp. 27–34, 1982.

[132] S. Applebaum and D. Chapman, "Adaptive arrays with main beam constraints", *IEEE Transactions on Antennas and Propagation*, vol. 24, no. 5, pp. 650–662, 1976.

[133] L. Castedo and A. Figueiras-Vidal, "An adaptive beamforming technique based on cyclostationary signal properties", *IEEE Transactions on Signal Processing*, vol. 43, no. 7, pp. 1637–1650, 1995.

[134] W. Liu and V. Prasanna, "Utilizing the power of high-performance computing", *IEEE Signal Processing Magazine*, vol. 15, no. 5, pp. 85–100, 1998.

Appendix A

Calibrating and Beamforming a Wideband Array in MATLAB

The following MATLAB code computes and applies a wideband calibration and beamforming routine. Data was sampled from the Rockwell Collins common module array described in Chapter 3.

```
clc
clear all
close all

%%
fs = 1.4e9;
fylim = [-20 50];

%% Data format is "adc_data_freq_siggenpower_steeringangle"
load 'adc_data_all.mat';

adcdata = adc_data_435_n15_0; offset = 0; BS = 1;
% adcdata = adc_data_435_n15_15; offset = 185; BS = 0;
```

```
% adcdata = adc_data_435_n15_30; offset = 180; BS = 0;
% adcdata = adc_data_435_n15_45; offset = 180; BS = 0;
% adcdata = adc_data_435_n15_60; offset = 170; BS = 0;
doublespacing = 0;
ttd = 1;
w = ones(16,1);

frf = 4.35e9;
% dx = 0.0172;
dx = .0167; ...
%lambda/4 at 4.5 GHz if array is lambda/2 at 18 GHz

%% Adjust for ADC buffers offsets
DD = [-13 -13 0 0 0 -7 0 0 0 0 0 0 0 0 0 0];
for k = 1:16
    adcdata(k,:) = circshift(adcdata(k,:), [0 DD(k)]);
end
adcdataf = fftshift(fft(adcdata,[],2),2);

L = size(adcdata,2);
t = (0:(L-1))/fs;
f = (0:(L-1))/L*fs-fs/2;

%%
twindow = offset + (120:3755);
```

```
L = length(twindow);
Lp = L/4;

twin = (0:(L-1))/fs;
fwin = (0:(L-1))/L*fs-fs/2;

adcdatatwin = adcdata(:,twindow);
adcdatatwinf = fftshift(fft(...
adcdata(:,twindow),[],2),2);

%% Calibrate polyphased amplitude offsets
oddoffset = 4;
Lz = length(twindow)+oddoffset; %zeropad
Lpz = Lz/4;

% Add zeros to make it a factor of 4
adcdatatwin = [adcdatatwin ...
repmat(zeros(1,oddoffset),16,1)];

adcpoly0 = adcdatatwin(:,1:4:end);
adcpoly1 = adcdatatwin(:,2:4:end);
adcpoly2 = adcdatatwin(:,3:4:end);
adcpoly3 = adcdatatwin(:,4:4:end);

adcpolyrmsavgs = [sum(adcpoly0,2)/Lpz ...
sum(adcpoly1,2)/Lpz sum(adcpoly2,2)/Lpz ...
```

```
sum(adcpoly3,2)/Lpz];

adcpolyrmsavgsq = repmat(adcpolyrmsavgs,1,Lpz);

adcdatacal = adcdatatwin - adcpolyrmsavgsq;
adcdatacal = adcdatacal(:,1:(end-oddoffset));
adcdatacalf = fftshift(fft(adcdatacal,[],2),2);

%% Generate reference waveform from channel 8
refwave = squeeze(adcdatacal(8,:));
refwavef = fftshift(fft(refwave));

%% Build steering filterbank
nfir = 22; % the number of FIR taps
heq = zeros(1,nfir); heq(1) = 1;
clear R;
for i = 1:nfir
    R(:,i) = freqz((circshift(heq, [0 (i-1)])), ...
[1], fwin, fs).';
end

%% Delay reference channel to center impulse responses
D = nfir/2;
refwavefd = refwavef.*...
freqz([zeros(1,D-1) 1 zeros(1,nfir-D)], ...
[1], fwin,fs);
```

```
%%
N = numel(refwave);
B1 = 666;
B2 = .125;
boffset = 525;
ns = 1e-8;

%% Create window
W = tukeywin(N/2+1-2*B1,B2)';
W = [ns*ones(1,B1-boffset) W ns*ones(1,B1+boffset)]; ...
    % include loading to stablize matrix inverse)
W = [fliplr(W(1:(end-1))) W(1:(end-1))];
W = diag(W);


%% Match channels to reference waveform
clear yf yfref Hideal HiDD a Hact;
Nchan = 16;

if doublespacing == 1
    dx = dx*2;
end

load('HidealBS.mat');
th = -80:1:80;
```

```
S = (R'*W*R)^-1*R'*W;
for k = 1:numel(th)
    for i = 1:Nchan
        yf(i,:) = adcdatacalf(i,:);

        if BS == 1
            Hideal(i,:) = refwavefd./yf(i,:);
        else
            Hideal(i,:) = HidealOld(i,:);
        end

        dm = (i-1)*dx*sind(-th(k))/3e8;
        if ttd == 1
            HiDD(i,:) = squeeze(Hideal(i,:)).*...
     exp(-1i*2*pi*dm*(fwin-(-frf+fs/4)*...
sign(fwin))); % Phase shift + TTD
        else
            HiDD(i,:) = squeeze(Hideal(i,:)).*...
                exp(-1i*2*pi*dm*frf*sign(fwin)); ...
% Phase shift
        end

        if doublespacing == 1 % For wider element spacing
            adcdatacalm(i,:) = ifft(ifftshift(...
fftshift(fft(squeeze(adcdatacal(i,:)))).*...
(0+1*squeeze(exp(1i*angle(Hideal(i,:).*...
```

```
yf(i,:)./(Hideal(1,:).*yf(1,:)))))));
        else
            adcdatacalm(i,:) = adcdatacal(i,:);
        end

        a(i,k,:) = w(i)*S*(HiDD(i,:).');
        realimagdetect = rms(imag(a(i,k,:))./...
rms(real(a(i,k,:)));
        a(i,k,:) = real(a(i,k,:));

        sigact(i,k,:) = filter(squeeze(a(i,k,:)), [1], ...
squeeze(adcdatacalm(i,:)));
        Hact(i,k,:) = freqz(squeeze(a(i,k,:)), ...
[1], fwin, fs);
    end
end

sigtot = squeeze(sum(sigact, 1));
sigtotf = fftshift(fft(sigtot, [], 2),2);

i1 = find(fwin >= 0,1);
i2 = find(fwin >= 300e6,1);
mf = ifft(ifftshift(conj(refwavef(i1:i2))));

clear sigtotmf
for k = 1:numel(th)
```

```
      sigtotmf(k,:) = conv(mf, hamming(i2-i1+1)'.*...
ifft(ifftshift(squeeze(sigtotf(k,i1:i2)))));
end

%%
ycal = adcdatacalf.*Hact;
yuncal = adcdatacalf;
ybeam_uncal = sum(adcdatacalf,1);
ybeam_cal = sum(ycal,1);

%% Calibration offsets
beemsteer = [.00009 .000052 .000027 -.00000006 ...
-.000027 -.000052 -.000072 -.000090];
beemsteer2 = -.00005*[1 1 1 1 1 1 1 1];

Httd = exp(-1i*2*pi*beemsteer(k+1)*(1:16)'*(0:(N-1)));
Httdn = .0005*exp(1i*2*pi*beemsteer(k+1)*(1:16)'*...
(0:(N-1))).*randn(Nchan,N);
Hfttd = fftshift(fft(Httd,[],2),2);
Httdsteered30 = exp(-1i*2*pi*beemsteer2(k+1)*...
(1:16)'*(0:(N-1)));

ycalttd = ycal.*Httd;
ycalttdphase = ycal.*Httd;
yuncalttd = yuncal.*Httd;
ycalttdn = ycalttd.*Httdn;
```

```
ycalttdphasesteered30 = ycalttd.*Httdsteered30;

%%
nchan = size(adcdatacalf,1);
nfreqs = size(adcdatacalf,2);
t = (0:1/1400e6:(nfreqs-1)/1400e6);

theta = 0:1:180;
z = zeros(length(theta),nfreqs);
zttd = zeros(length(theta),nfreqs);
zttdn = zeros(length(theta),nfreqs);
zuncal = zeros(length(theta),nfreqs);
zuncalttd = zeros(length(theta),nfreqs);
zttdphase = zeros(length(theta),nfreqs);
zttdphasesteered30 = zeros(length(theta),nfreqs);

%% Do the beamforming in azimuth
phioff = zeros(length(theta),nchan,nfreqs);
p = zeros(length(theta),nchan);
for i = 1:length(theta)
    p(i,:) = (exp(1i*pi/2*cosd(theta(i))*(0:(nchan-1))))';
    phioff(i,:,:) = repmat(p(i,:)',1,nfreqs);
    z(i,:) = sum(ycal.*squeeze(phioff(i,:,:)),1);
    zttd(i,:) = sum(ycalttd.*squeeze(phioff(i,:,:)),1);
    zuncal(i,:) = sum(yuncal.*squeeze(phioff(i,:,:)),1);
    zuncalttd(i,:) = sum(yuncalttd.*....
```

```
squeeze(phioff(i,:,:)),1);
    zttdphase(i,:) = sum(ycalttdphase.*...
squeeze(phioff(i,:,:)),1);
    zttdphasesteered30(i,:) = sum(...
ycalttdphasesteered30.*squeeze(phioff(i,:,:)),1);
    zttdn(i,:) = sum(ycalttdn.*squeeze(phioff(i,:,:)),1);
end

%% Normalize responses
findex = 2200;
chanindex = 3;

flow = 2141;
fmiddle = 2335;
fhigh = 2564;
fhigh = 2525;
fout = 3000;

Nftemp=fhigh-flow;
fsp = linspace(-100e6,100e6,Nftemp+1)+4.4e9;

znormed = max(max(abs(z(:,flow:fhigh))));
zttdnormed = max(max(abs(zttd(:,flow:fhigh))));
zttdphasesteered30normed = ...
    max(max(abs(zttdphasesteered30(:,flow:fhigh))));
zttdnnormed = max(max(abs(zttdn(:,flow:fhigh))));
```

```
zttdnnormed = max(max(abs(zuncal(:,flow:fhigh))));
zttdnnormed = max(max(abs(z(:,flow:fhigh))));
zuncalnormed = max(max(abs(zuncal(:,flow:fhigh))));

Nftemp=fhigh-flow;
fsp = linspace(-100e6,100e6,Nftemp+1)+4.4e9;

%% True Time Delay Filterbank Analysis
nfilts = 20;
ntaps = 46;
downsamplefactor = 500;
Nt = ntaps*downsamplefactor;

t = (0:1/1400e6:(Nt-1)/1400e6);
fbank = sinc(22.9*(repmat((((0:(length(t)-1))/...
length(t)-.5)-1/nfilts/ntaps/2),nfilts,1)-...
    .001075*repmat((0:(nfilts-1))',1,Nt)));

% Shift by fs/4 in frequency
mixfsover4 = (repmat(exp(1i*pi*((0:(size(fbank,2)-1))/...
    size(fbank,2)-.5)*Nt/2),nfilts,1));
fbankmixed = real(fbank.*mixfsover4);
fbankf = fftshift(fft(fbankmixed,[],2),2);

% Apply a Blackman window to the filterbank
w_indow = repmat(blackman(size(fbankmixed,2))',nfilts,1);
```

```
fbankwindow = fbankmixed.*w_indow;
fbankfwindow = fftshift(fft(fbankwindow,[],2),2);

fbankdown = downsample(fbank',downsamplefactor)';
mixfsover4down = (repmat(exp(1i*pi*(...
(1:(size(fbankdown,2)))/size(fbankdown,2)-.5)*...
Nt/downsamplefactor/2),nfilts,1));
fbankdownmixed = real(fbankdown.*mixfsover4down);
fbankfdown = fftshift(fft(fbankdownmixed,[],2),2);

% Compensate for amplitude mismatches
ampoff = sqrt(mean(abs(fbankdownmixed).^2,2));
fbankwindowdownampoff = fbankdownmixed./...
    (repmat((ampoff).^2,1,size(fbankdownmixed,2)));

% Apply Window
w_indowdown = repmat(blackman(size(fbankdownmixed,2))',...
nfilts,1);
fbankwindowdown = fbankwindowdownampoff.*w_indowdown;
fbankfwindowdown = fftshift(fft(fbankwindowdown,[],2),2);
```

Appendix B

Matrix Inversion with the Matrix Inversion Lemma in MATLAB/HDL Coder

The following Simulink/HDL Coder block diagram code implements the MIL algorithm described in Chapter 4.

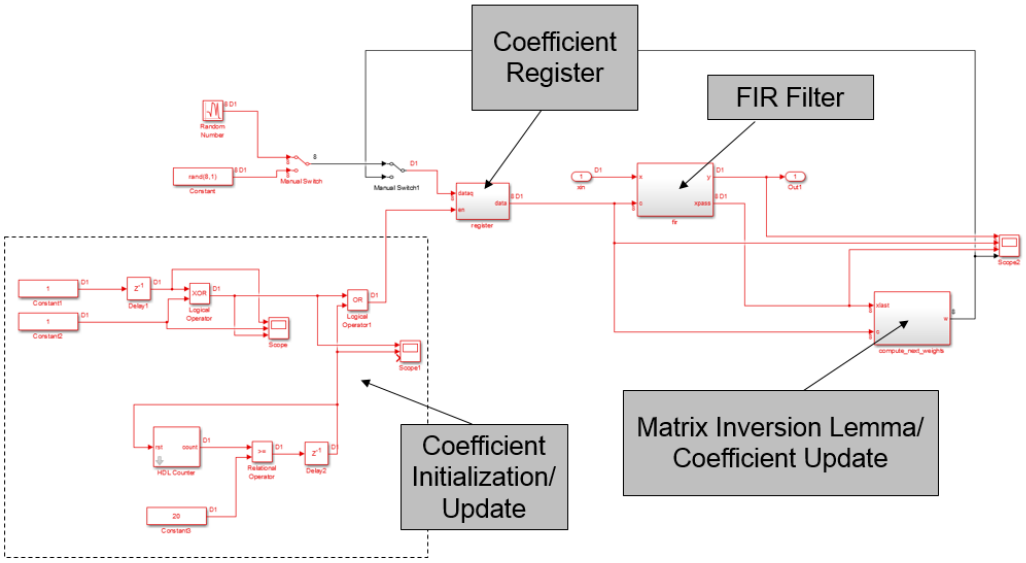

Figure B.1: Top level dataflow diagram of the MIL.

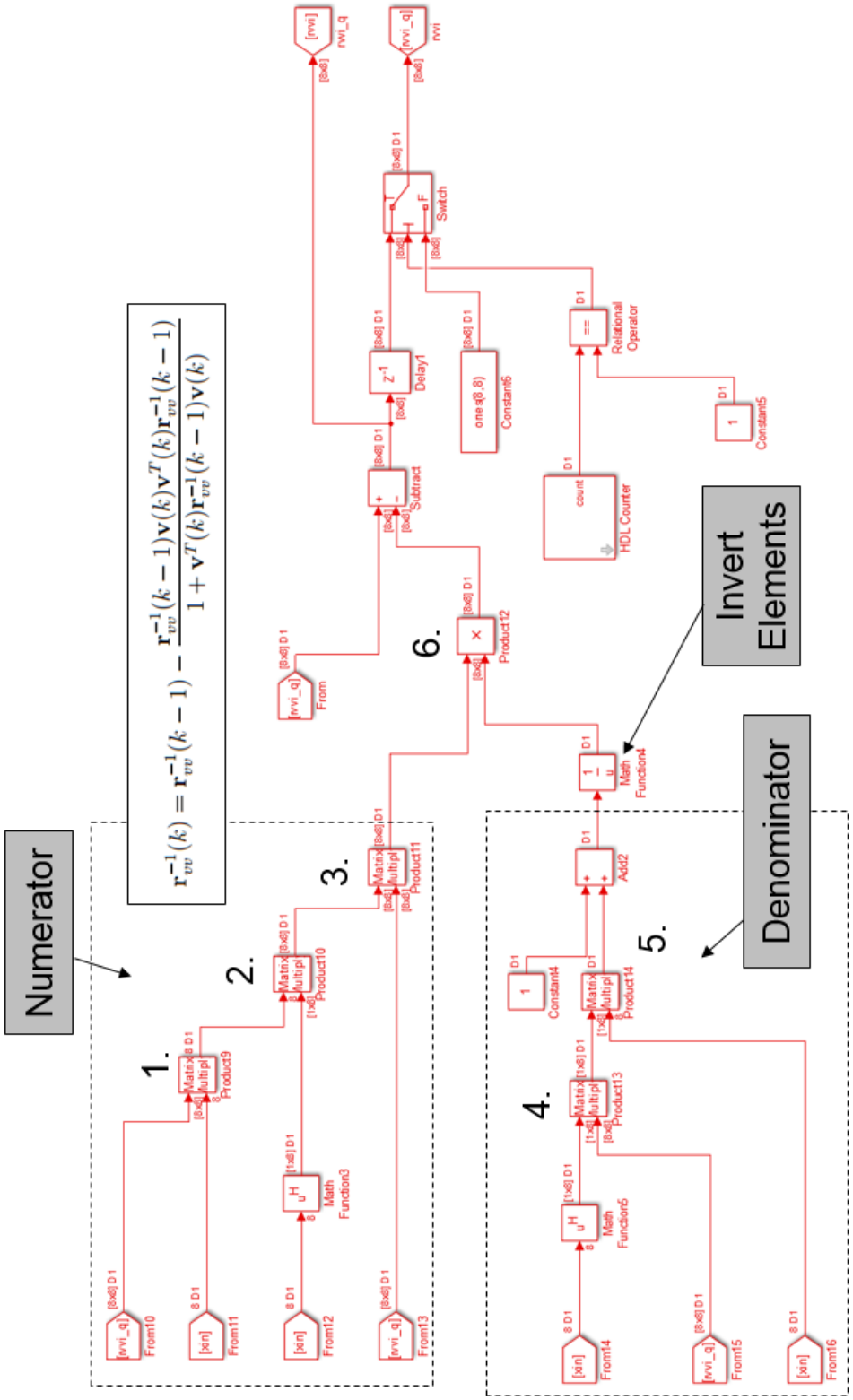

Figure B.2: Computing the inverse covariance matrix with the MIL.

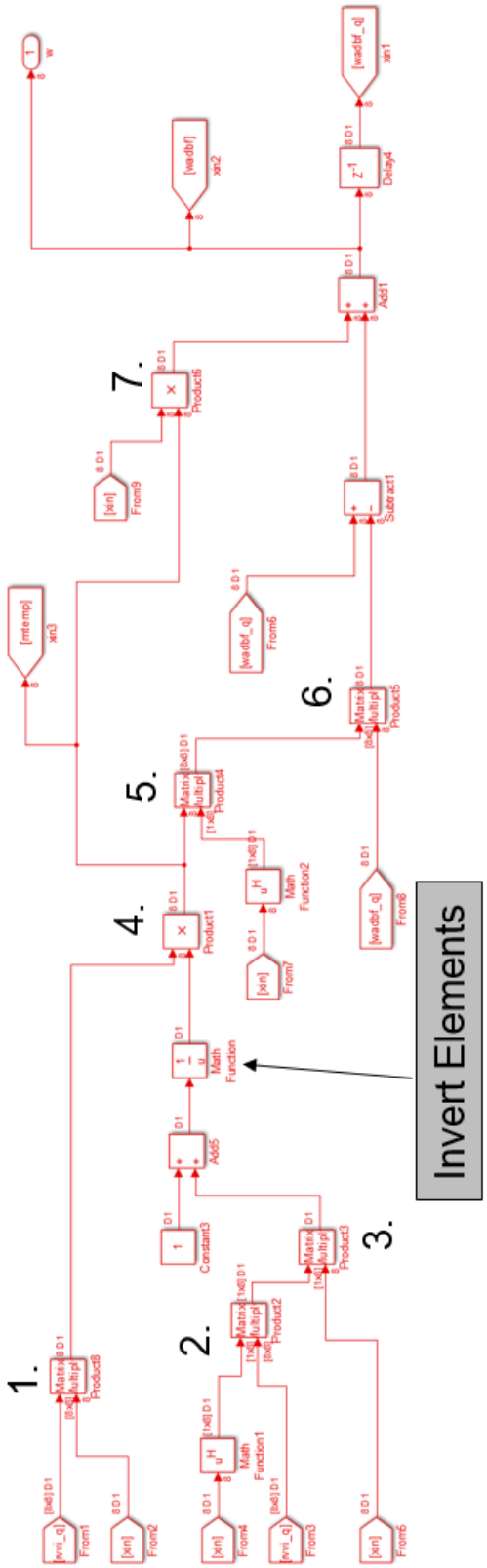

Figure B.3: Updating adaptive weights with the MIL.