DESIGN AND VALIDATION OF A SCALABLE DIGITAL WIRELESS CHANNEL EMULATOR USING AN FPGA COMPUTING CLUSTER

by

Anthony Scott Buscemi

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical and Computer Engineering

Charlotte

2013

Approved by:

Dr. Ronald R. Sass

Dr. James M. Conrad

Dr. Daniel Noneaker

Dr. Jiang (Linda) Xie

Dr. Aidong Lu

ABSTRACT

ANTHONY SCOTT BUSCEMI. Design and validation of a scalable digital wireless channel emulator using an FPGA computing cluster.
(Under the direction of DR. RONALD R. SASS)

A Digital Wireless Channel Emulator (DWCE) is a system that is capable of emulating the RF environment for a group of wireless devices. The use of digital wireless channel emulators with networking radios is hampered by the inability to efficiently scale a DWCE to a large number of nodes. If such a large scale digital wireless channel emulator were to exist, a significant amount of time and money could be saved by testing networking radios in a laboratory before running lengthy and costly field tests. By utilizing the repeatability of a laboratory environment it will be possible to investigate and solve issues more quickly and efficiently. This will enable the performance of the radios to be known with a high degree of certainty before they are brought to the field. This dissertation investigates the use of an FPGA cluster configured as a distributed system to provide the computational and network structure to scale a DWCE to support 1250 or more wireless devices. This number of wireless devices is approximately two orders of magnitude larger than any other documented system. In this dissertation, the term "scale" used for a DWCE is defined as an increase of three key factors: number of wireless devices, signal bandwidth emulated, and the fidelity of the emulation. It is possible to make tradeoffs and reduce any one of these to increase the other two. This dissertation shows a DWCE that can increase all of these factors in an efficient manner and thoroughly investigates the fidelity of the emulation it produces.

ACKNOWLEDGMENTS

My pursuit of a PhD has taken ten years. Over those years there have been many people who have helped me down the path I chose but none more important to me than my wife Lisa and my parents, Sandi and Phil. You all have given me the inspiration, motivation, and encouragement I needed to be successful. To Lisa specifically, thank you for being my rock when I was drowning and giving my work purpose. To my little sister: Jackie, thanks for putting up with me and letting me mercilessly beat the crap out of you. You are the best stress reliever any brother could ask for.

I had a large group of people help me put the DBES together and make this project a success. My IPT lead Rich Anderson has had faith in me and this project, even when schedules and budgets were overrun. Thanks to the DBES team in Charleston, Brent Anderson, Terry May, Greg Sheets, Jude Seeber, and David Hansen and everyone in the RCS lab at UNCC but especially Will Kritikos. All of you have contributed directly and significantly to this project writing code and helping me design the full system. I wish I could have spent more time writing about the amazing and innovative work each of you has done for this project.

This dissertation would not have happened were it not for the support of the SMART scholarship and SPAWAR Systems Center Atlantic. These two organizations have provided the funding for me to go to school and complete this project.

Lastly, thank you Ron. It has been a rocky road getting my doctorate. You have helped me navigate these academic streets time and time again by supporting my efforts and guiding me when I did not know where to turn. I know that while my doctorate journey may soon be over, we'll continue to work together over the years to come.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC	analog to digital converter
AFE	analog front end
AGC	automatic gain control
BB	baseband
BEE	berkley emulation engine
BRAM	block RAM
CN	compute node
COTS	commercial off the shelf
DAC	digital to analog converter
DBES	digital battlespace environment simulator
DWCE	digital wireless channel emulator
FDTD	finite difference time domain
FMC	FPGA mezzanine card
FPGA	field programmable gate array
FSPS	free space path loss
GUI	graphical user interface
GTX	gigabit transceiver
IF	intermediate frequency
IQ	in-phase and quadrature
LO	local oscillator
MANET	mobile ad hoc network
MGT	multi-gigabit transceiver
MIMO	multiple input multiple output
PC	propagation computations
PCB	printed circuit board
PP	propagation path

RCC	reconfigurable computing cluster
RF	radio frequency
RGC	receive gain control
RN	radio node
RTM	rear transition module
SBW	signal bandwidth
SC	scenario compiler
SER	symbol error rate
SNR	signal to noise ratio
SP	signal power
SQ	signal quality
SSD	solid state drive
TDL	tapped delay line
TGC	transmit gain control
VGA	variable gain attenuator
VHDL	VHSIC hardware description language
VHSIC	very high speed integrated circuit
WCE	wireless channel emulator
Wi-Fi	IEEE 802.11 specification

CHAPTER 1: INTRODUCTION

Wireless devices have become and will remain ubiquitous in modern life. Many different technologies are available from cell phones and wi-fi devices to bluetooth and ad hoc radio communication networks. New waveforms and new wireless technologies are constantly being invented. However, testing these wireless devices is becoming increasingly challenging as the size and scope of the networks that can be formed continue to grow. To combat this complexity, developers, industry standards groups, and others would like to be able to examine new radios in the lab using an emulated environment. The alternative is field testing, which is time consuming and resource intensive in terms of equipment and manpower. Large scale testing in the lab requires a large scale Wireless Channel Emulator (WCE).

A WCE is a system capable of emulating the signal propagation environment between wireless Radio Frequency (RF) devices. WCEs can be classified as either analog or digital. Analog WCEs use programmable attenuators and/or amplifiers to dynamically change the power level between wireless devices. Analog WCEs have been scaled allowing for as many as 64 wireless devices to use the system at once; however, analog WCEs cannot take into account channel effects such as signal propagation delay, Doppler, and multipath effects. A Digital Wireless Channel Emulator (DWCE) digitizes RF signals from a transmitting wireless device, digitally manipulates the signal, and then converts it back to an analog RF signal to be sent to the receiving wireless device. Because the signal is digitized, any number of computations can be performed on that signal to manipulate it in the same way as channel effects would manipulate an actual RF signal in a real environment. Commercial DWCEs are available from a few companies such as Spirent and Electrobit and custom DWCEs have been built in academia. Both have seen use in different wireless device communities to test their devices in an accurate and customizable environment.

A major challenge with custom or commercial DWCEs is that they only support a limited number of channels. A channel on a DWCE refers to a connection from the DWCE to a wireless device. Each channel has a single Analog Front End (AFE) that converts an RF signal to a digital signal and vice versa. Typically, one wireless device will need one available channel on a DWCE to connect to the system; however, some wireless devices use multiple antennas and/or multiple RF tuners, which could require multiple available channels on a DWCE for each wireless device connected to the system. Having a DWCE with a large number of channels is not necessary for many point-to-point wireless devices such as traditional analog radios, cell phones, and IEEE 802.11 (Wi-Fi) devices as just a few links are sufficient to determine its efficacy. However, wireless devices such as MIMO Radios, RF Electronic Warfare Devices (i.e. Radio Jammers), and Ad-Hoc Networking Radios are designed to work in an environment with many wireless devices and may require multiple DWCE channels per device. These types of wireless devices cannot truly be evaluated by a DWCE with only a few AFEs. What is desired is a wireless channel emulator that can scale to large numbers of channels and can produce the channel effects to accurately emulate the RF environment. This precludes the use of analog WCEs and the rest of this dissertation only discusses digital wireless channel emulators unless specifically mentioned otherwise.

The challenge and reason why no large-scale DWCE exists, is that the amount of computation required to build a DWCE scales on the order $O(N^2)$ where N is the number of channels. This is due to the fact that each channel must send and compute information to every other channel, resulting in $N(N-1)$ possible combinations in a system. Additionally, the signal bandwidth and the fidelity of the emulation also impact the amount of computation required, but the computation requirements increase only linearly with respect to these. The signal bandwidth defines the frequency difference between the highest frequency and lowest frequency emulated signals. For more information on the process of converting an RF signal to baseband, please see Section 3.1.3. The signal bandwidth defines how many samples must be processed each second, which creates the proportional relationship with computation. The fidelity of a given DWCE system could involve any number of different factors depending on models used by the emulation and the hardware that makes up the system. While it may be possible to increase the fidelity of a given system by improving the hardware in the system, the model used for manipulating a signal has a large impact on the fidelity as well. The system described in this dissertation uses a Tapped Delay Line (TDL) model described in Section 3.2 where the fidelity of the model is largely defined by the number of propagation paths emulated. Other systems may use a statistical model which would have a different metric comparing fidelity to computations. Regardless, it is important to note that having a higher fidelity model will necessitate a higher computational load. This results in a rule that is shown in Equation 1.1.

$$\text{Computations} \propto (\text{RF Devices})^2 \times \text{Signal Bandwidth} \times \text{Fidelity} \tag{1.1}$$

This rule cannot be broken without performing optimizations that significantly limit the capabilities of the DWCE. One such method employed by some commercial DWCEs is to segment the system and not allow certain channels to communicate. A system may be designated as a $A \times B$ system indicating that the A devices can communicate with the B devices and vice versa, but the A devices cannot talk to A devices and B devices cannot talk to B devices. Such optimizations come at a cost to either fidelity or flexibility are therefore not considered in this thesis. It is desirable to analyze as broad a range of devices as possible and these optimizations generally take advantage of a specific characteristic of a specific wireless device that would not be applicable to a different wireless device. The aforementioned optimization may work fine for MIMO wireless devices but would severely limit the DWCE capabilities for single antenna Ad-Hoc Networking radios for example. Many WCE solutions have attempted to reduce the signal bandwidth or the fidelity in order to increase the number of radios and maintain a low enough computational load; however, it is desirable to keep all three of these factors high.

The principle design limitation for all previous attempts at a DWCE is the utilization of a "Hub and Spoke" type system. In this design there is a central "hub" location that performs the computations with connections ("spokes") to all the analog front end components that translate the signal between analog and digital. Most hubs are a single FPGA; however, some organizations have utilized multiple FPGAs on a single Printed Circuit Board (PCB) such as Mitre and Intelligent Automation [33]. These single PCB solutions are still considered to be a hub because there are only a limited amount of FPGAs that will fit onto a single PCB. If someone using a hub and spoke design wanted to expand their system, it would require a complete re-design. This is the main problem with increasing the size of any hub and spoke design: each has a definable upper performance limit based on the amount of computation the hub can perform. A new way to approach this problem is to distribute the digital processing and communication across many computational elements.

Distributing the computation across several computational elements introduces a new constraining factor in addition to the computational requirements: the amount of data passed between channels can result in a large amount of network traffic. As the amount of processing increases beyond the capabilities of a single chip or a single PCB, it is necessary to stream multiple signals at a high sample rate between processors, placing an enormous demand on the chip-to-chip networking. This is a major technical limitation that has not been addressed by academia and is a focus of this dissertation.

This thesis describes the validation, scalability, and feasibility of using a novel computing architecture to implement a digital wireless channel emulator: a distributed cluster composed of FPGA devices augmented with required RF components and connected by a custom high-speed network. This thesis' hypothesis:

Constructing a DWCE with a distributed architecture provides scalability superior than previously documented systems while maintaining high fidelity.

The term "scale" and "scalability" with respect to DWCE in this thesis refers to three different aspects: the number of wireless devices that can be connected to the system, the signal bandwidth that is emulated by the system, and the fidelity of the system.

The hypothesis has been broken down into three questions. The first question is: is it possible to build a DWCE utilizing a distributed architecture? The goal from the beginning of this project was to create a thirty-channel digital wireless channel emulator with 30MHz signal bandwidth and twenty propagation path computations. Each of these parameters were chosen for a reason. Thirty-channel capability was selected because several previous field tests with military Ad-Hoc networking radios used around thirty radios. No DWCE existed that could handle this size and the largest documented DWCE systems at the time only used a one to four FPGAs. The solution envisioned would use thirty-two or more FPGAs, which was greater than an order of magnitude more computational power than any existing channel emulator. A signal bandwidth of 30MHz is selected because that was the greatest amount of bandwidth used by all current and near-future military networking waveforms. Twenty propagation paths was selected based on a study titled "Recommended channel models for WNW 3.1 tests" written by Dr. Dan Noneaker [22]. The study gave channel models for rural, urban, and hilly scenarios and the maximum number of propagation paths listed was twenty. The study claims that after that threshold, the propagation paths typically have little effect because their relative attenuation is significantly less than the dominant path. To answer this question, a feasibility study was performed.

Engineering work was performed to support this study, determine the system hardware and architecture, and develop the VHDL for the core components of the system. Detailed information for these building blocks was needed to identify the networking and computational limits of a system. The answer to the first question will be in the affirmative if a realistic DWCE is demonstrated with thirty channels, 30MHz signal bandwidth, and twenty propagation paths.

Once a single architecture solution is found in the feasibility study the next step is to determine how this architecture can scale. This is what the second question addresses: can this architecture provide a scalable and cost effective solution as compared to other testing alternatives? As shown in Equation 1.1 the amount of computation for the system is largely determined by three parameters: the number of channels (RF Devices), the signal bandwidth, and the number of propagation paths (fidelity). For any given DWCE system, there is a limit on each of these parameters that is determined by the amount of hardware the system uses. In a given system, perhaps the signal bandwidth is limited by the network throughput which then defines the number of propagation paths that can be emulated. In a different system, the amount of FPGA logic resources available may be the limiting factor for both the fidelity and signal bandwidth. Thus, to answer this question, a scalability study will be performed.

The goal of the scalability study is to find the limit for a given subset of systems and then determine how each system must be changed in order to handle even larger parameters. The study attempts to find the trade space between each of the three parameters using a developed hardware platform and architecture. Because of the distributed nature of the system, an infinite number of systems could be connected and analyzed. In order to limit the scope of the work to a manageable size and focus on the most promising systems only two different network configurations were analyzed which will be thoroughly explained in Section 4.2. This will not only analyze the current system but will also provide the framework for an engineer to analyze future DWCE systems. The scalability study will employ many differently sized DWCE systems; however, this alone does not answer the cost effectiveness part of the question. A system will be considered cost effective if it can be built and used for less than the cost of the field test it would replace. In this case, one can suppose that it would be preferable to invest in this technology over going to the field, especially considering that any subsequent lab testing will be at a significantly reduced cost. Based on performed lab and real-world tests, a lab test is ten to twenty percent of the cost of an identical field test. Therefore, if the cost of building a DWCE is less than eighty percent the cost of the field test, then this condition will be met.

Just being able to build a large-scale cost-effective system is not sufficient reason to warrant the use of a DWCE. The final question: is the fidelity of the system good enough to be a substitute for the alternative field test? This question is much more subjective than the previous two and it will not be possible to give a complete affirmative to this question. Nonetheless, this is an important question to ask because if the DWCE cannot accurately reproduce an RF environment then the system is practically useless. Without complete documented knowledge of the capabilities of the system, it is possible for skeptics to argue its effectiveness. Even worse, when proponents of a DWCE use it to test their radios, and the radios do not behave as anticipated, a study is necessary to show that the the system is not interfering with normal radio operation. While it can easily be shown that a given DWCE system is very accurate under certain circumstances, the problem is the multitude of different scenarios and situations that a DWCE must reproduce. In this large space there are some scenarios that are not possible to reproduce due to limitations of the hardware that makes up the DWCE and many of these limitations will be discussed. For example: there is a limited amplitude dynamic range of the system due to isolation and power constraints that is analyzed and quantified. In addition, the wireless device under test will determine many of the required performance metrics, making the set of potential corner cases different with every device connected to the system. Clearly, this study cannot analyze every wireless device or protocol that may be used with the DWCE.

To approach this question, a fidelity study is proposed. This study is comprised of a series of theoretical, experimental, and observational analyses which are thoroughly explained in Section 4.3. The principle idea is to theoretically determine expected performance based on signal processing theory and simulations, experimentally confirm those theoretical predictions, and then observe behavior of radios connected to the DWCE. The requirements for any given wireless device can then be compared to the DWCE theoretical and experimental studies results to determine that wireless devices computability with the DWCE. For the observational analysis a specific wireless device will be selected and the comparison will form a basis for expected observational behavior. The selected wireless devices will be put under a specific set of scenarios designed to show the strengths and limitations of the DWCE. The goal is to explore a few selected scenarios which cover the most anticipated scenarios to give better understanding to how the DWCE performs. While this study is not the final answer to the fidelity question, it will largely address it for a given wireless device and it will provide a framework for people to investigate this question for future wireless devices.

The remainder of this dissertation is organized as follows. Chapter 2 presents background information on wireless channel emulation. The Ad-Hoc networking radios are described first. This is the technology that was the motivation behind the creation of this work and these radios will greatly benefit from the utilization of a large-scale DWCE. Next, alternative wireless channel emulator systems are presented. It will be shown that all these alternative designs utilize a hub-and-spoke system which critically limits the scalability. Lastly, other DWCE fidelity studies are presented. Most of these studies are limited in scope with one exception. This paper will form the basis for the DWCE validation study.

Chapter 3 discusses the detailed design of a DWCE called the Digital Battlespace Environment Simulator (DBES), named after its analog predecessor the Battlespace Environment Simulator (BES). The DBES is the implementation of a DWCE that this dissertation is based on. This chapter is broken up into six sections and begins with the primary factors that limit the scale of DWCEs and discusses solutions that were found to push these boundaries. It describes how solutions were found for the DBES design starting with the system overview and then proceeds to drive down into the design one level at a time until the lowest levels including the signal path component (the central component performing signal manipulation processing) are detailed. At the lowest level, the Tapped Delay Line model is theoretically investigated by deriving the end-to-end mathematical signal processing performed. The third section deals with the hardware that implements the system. No commercial solutions could be found for the majority of the hardware required for the DBES. Initially it was thought that entirely new hardware would have to be developed. Fortunately, commercial solutions were found that were very close to the required hardware and these solutions were modified to produce the boards with all the components required. The fourth section describes how a commercial scenario compiler, REMCOM's Wireless In-Site software, is used with the system. The fifth section describes the DBES control graphical user interface that was designed to handle loading the configuration and parameters to the DBES and controlling the system. The sixth and final section discusses a use-case scenario for the DBES. There are many steps to running the system and while many of them will eventually be automated, there are still several steps that a user will need to go through in order to use the DBES. This section describes that process.

Chapter 4 is the evaluation chapter. It covers the investigations performed to analyze the system and contains the studies performed to answer the questions posed in this chapter. Chapter 4 starts with the initial feasibility study that uses a two-channel prototype and discusses the results and lessons learned from this prototype. The second section is the scalability section where an exact systems solution will be given for an arbitrary amount of nodes. The final section is the fidelity study that theoretically, experimentally, and observationally qualifies the accuracy of the system. Theoretically, it lays out the noise injected by the software and hardware. Next, it experimentally validates the necessary parts of the theoretical analysis. The experimental section will also include a hardware evaluation section where tests are performed to analyze the noise of the analog front end and the propagation delay in the final system. These tests form a baseline of functionality that will be necessary to explain the experimental results. Lastly, the observational section will show actual radios using the DWCE and compare the resulting behavior with predicted outcomes. Three scenarios were run including two two-radio scenarios and one five-radio scenario.

Chapter 5 is the conclusion. Key results are reiterated and the hypothesis questions will be answered based on the presented results in the previous sections.

CHAPTER 2: BACKGROUND

2.1 Radio Technology

Mobile *Ad Hoc* Networks (MANETs) arise when multiple hosts dynamically discover their neighbors and establish a wireless network whose topology changes as hosts move around in the physical world. As long as the network remains connected, any host can communicate with any other host (possibly using intermediate hosts to relay packets). Moreover, the communication links between nodes vary over time based on the environment and movement of hosts. This variability is illustrated in Figure 2.1. In the initial configuration (a), four nodes are connected by the communication links shown. Some hosts are not directly connected; which would be due to something in the physical world, perhaps an obstruction or just distance. As illustrated (b), the hosts may move over time (in this example, D moves southwest and B moves east). As a result, some communication links are lost (D–C) and new communication links are established (C–B). Further complicating the situation, MANET radios have many real world difficulties due to the multiple vendors that create radios for the network and the decentralized nature of algorithms and protocols. The dynamic nature of

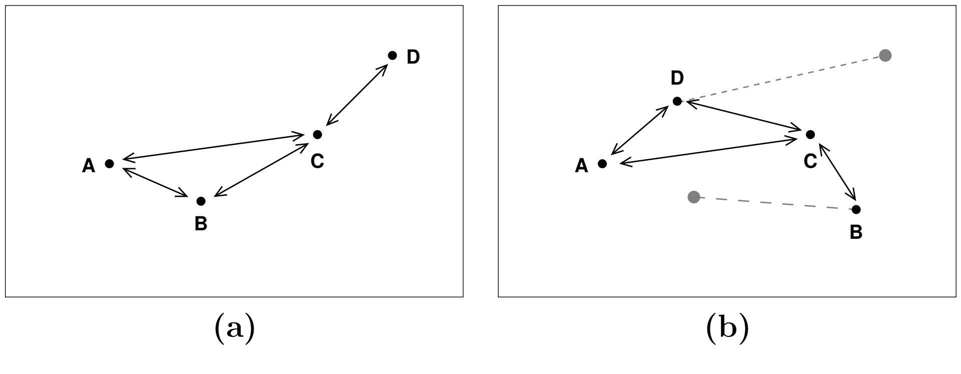


Figure 2.1: MANET network example: (a) time t (b) time $t + \tau$

MANETs has presented challenges throughout the lifetime of the standards — from how to design the next generation protocols to assuring the interoperability of finished products. Commercial MANET radio products available today will typically use multiple processors running a distributed system with several different communication protocols and modes of operation. Moreover, all these components have their own software base that has been developed by multiple vendors. This makes the process of debugging the radios and ensuring capability an extremely challenging task. To make matters worse, many features of these radios cannot be examined without forming large networks of thirty or more radios in controlled situations. Presently, to create these networks, one needs to perform a field test where each radio is loaded onto a mobile platform (on foot, in vehicles, on helicopters, etc.) filled with a plethora of supporting equipment which gather a range of performance data. These events take months to plan and execute, require a large supporting cast of technicians and engineers, and can result with more questions about the wireless device than answers because of the inability to reproduce a single test. This means that network designers have been forced to rely on model assumptions and computer simulations in order to evaluate experimental protocols under various scenarios. Likewise, it is very costly to stage field tests of finished products.

Ideally, one would like to be able to use radios in a laboratory setting and emulate the physical environment as found in general field settings. This would dramatically reduce the amount of personnel and equipment required to run a test and introduce repeatability and fine scenario control for the user. In addition to evaluating network-wide performance, the data gathered would be extremely valuable to scientists and engineers developing new MANET protocols. Unfortunately, no wireless channel emulator exists that can meet the requirements of the Ad-Hoc networking community. Of those that do support a larger number of radios, none can meet the scale, bandwidth, and accuracy required to vet current MANET radios. Current analog WCEs can meet the scaling requirement; however, they do not meet the accuracy requirements because they cannot provide channel effects such as Doppler and multipath. Commercial *digital* wireless channel emulators can create these effects but currently only support a limited number of channels. While some academics have written papers detailing a method of scaling to larger number of channels, the approaches described are typically limited by the capacity of a single device. For example, [4] uses a single FPGA but does not describe how to extend it beyond one chip.

Multiple Input Multiple Output (MIMO) technology can also greatly benefit from a large scale DWCE. The input and output in MIMO describe the antennas used in the system. There are many different reasons to employ these multiple antenna and many different algorithms and antenna patterns that are used. Some MIMO devices can determine, based on the time of arrival of the signal at each antenna, the direction a signal comes from. This enables the radio to tune into a location to boost signal strength and to filter out interfering signals, on the same frequency as the signal of interest, based on location. Other MIMO devices simply use each antenna as a separate channel to communicate with. There are many other potential uses for MIMO radios but the important point for this dissertation, is that a single MIMO radio requires multiple channels on a DWCE. Each signal picked up at each antenna is slightly different, which enables much of the functionality of MIMO devices. This makes testing a MIMO device with an analog WCE not possible. There is no way to simply set up a network of attenuation between pairs of MIMO radios to determine their functionality. In order to test a pair of MIMO radios in the lab, it is necessary to use a large (by today's commercial standards) DWCE which is much too expensive for academics and start up companies to afford. Currently, many MIMO researchers place the antennas on the tops of their cars and drive around a city to gather data [9]. This presents a huge host of problems testing an debugging the radio that could be alleviated by in-lab testing.

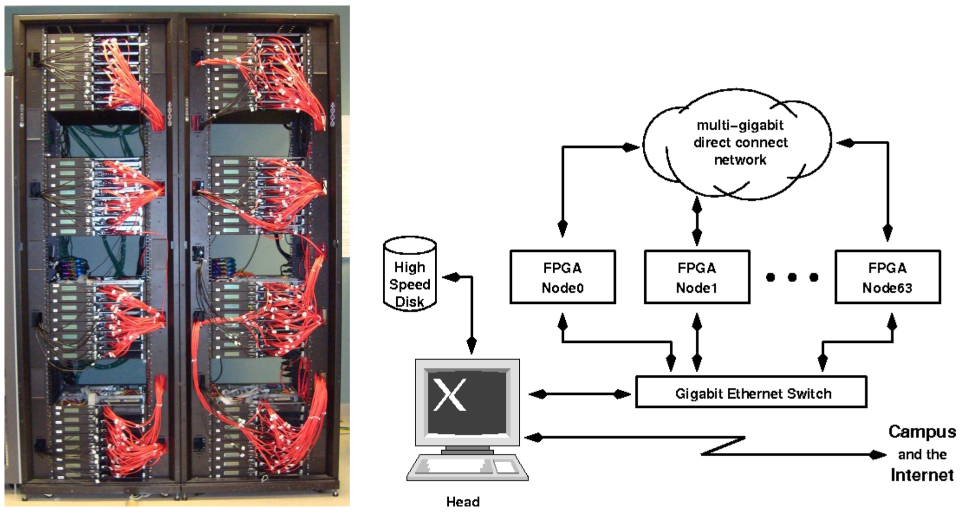


Figure 2.2: SPIRIT rack view and network architecture

2.2 *SPIRIT* FPGA Computational Cluster

A novel computing architecture was required that has not only the capability for large amounts of signal processing, but also for a tremendous amount of network traffic. This capability was found in the *Spirit* cluster. *Spirit* is a 64-FPGA cluster that was built as part of the Reconfigurable Computing Cluster project at UNC-Charlotte [28]. This machine consists of 64 FPGA nodes with no discrete microprocessors. The network switch is integrated into the reconfigurable logic of the FPGA and nodes are directly connected one another in a 3-D torus (six bidirectional channels). Using 2004 CMOS technology, *Spirit* provides 64 Gbps of bandwidth to each node. A picture of the cluster itself and a graphic of the network architecture is shown in Figure 2.2.

After a brief investigation, it was realized that the *Spirit* architecture was ideal for a large-scale DWCE; however, *Spirit* itself did not have sufficient requirements to meet the needs of the DWCE that was envisioned. In part, this is due to the type of FPGA utilized, which emphasized hardcore discrete processors over multiplication cores, and the networking capabilities of the FPGA, which only had 8 transceivers available. Newer FPGAs have over ten times the logic and five times the networking capabilities which would be more than sufficient for the DWCE requirements. This lead to the realization that what is needed is an updated version of the *Spirit* cluster. So, new hardware was constructed that is detailed in this dissertation.

2.3 Wireless Channel Emulation Related Work

The idea to utilize a wireless channel emulator to emulate a field environment for radios has been around for some time. Initially these were analog, utilizing a bank of variable attenuators that could be changed to replicate the environment. SPAWAR Atlantic has a 14 radio WCE that can update the attenuation at millisecond intervals. The fidelity of these analog WCEs is unfortunately limited because they cannot reproduce channel effects such as multipath, Doppler, and frequency selective fading. There are methods for emulating each of these effects in the analog world; however, a more practical solution is the use of a digital wireless channel emulator.

There are many groups that have produced commercial and custom DWCE solutions (SPIRENT [30], Elektrobit [11], MITRE [2], and Intelligent Automation [33]) and the idea for a DWCE has been around since the 1990's. One of the first attempts was published by Olmos and Feminas [23] who developed a single channel solution using ASIC chips. This was a good solution; however, modern FPGAs have more resources available and can now support enough computations to make DWCEs, which has driven researchers to use FPGAs. Several papers have since utilized FPGA technology to develop DWCEs [4, 12, 21, 23, 25]. Some were specifically designed to test multiple input multiple output radios [17, 20]. MIMO radios cannot be accurately tested with analog WCEs as they rely on small differences in the propagation delay between receiving antennas. In general, any multi-channel DWCE can be made to test this technology provided they maintain accurate control of the delay and timing between different signal paths. The problem is that a large number of channels with an all-to-all connection scheme is required for these systems. Most commercial DWCE reduce the number of fading paths in order to claim more channels. The largest commercially available DWCE as of 2013 is a 32-channel system by Elektrobit; however, this system can only emulate 128 fading paths. This means that it could only do a full mesh network between 11 channels and each channel would only have a single propagation path emulated. Only two four-antenna MIMO radios could be connected to this system and many multipath effects, which are a critical issue for MIMO radios, could not accurately be emulated. In comparison, the 32-channel system described in Chapter 3 can support over 14,000 independent fading paths.

MITRE has released a white paper in May 2013 detailing their system [2]. They claim to be able to reach a 64-node system but do not provide any description in how they hope to achieve this. They discuss their current system which is a 16-channel system; however, they sacrificed fidelity in order to increase the number of channels. They describe using two Virtex-5 240sx FPGAs for processing and different ”cells” to do different types of channel emulation. The way they reach the 16-channel full mesh system is by using an attenuation-only cell. This would produce results with no multipath, fading, or Doppler effects.

The equations implemented by these papers generally fall into the sum-of-sinusoids (SOS) approach [1] or the tapped delay line approach [20]. Both these equations are thoroughly discussed in literature [26] and are widely accepted equations used for simulations. One paper that does not exactly fall into these categories attempts to perform computations in the frequency domain in an effort to improve scalability and accuracy [12]. This is an interesting idea; however, there was difficulty with the propagation delay and coherency between frames that made this solution less than ideal for the implementation presented in this paper.

Many papers have attempted to address the scalability issue directly [17]. Specifically, Steenskiste at Carnegie Mellon University has an excellent TDL solution utilizing a Virtex 5 FPGA. His solution is capable of connecting 15 radios, however in this scenario the accuracy is lacking because it “can only create a few multipath channels while maintaining at least a single path between all the nodes" [4]. As mentioned in the introduction, DWCEs are difficult to scale to large number of nodes due to the amount of computation required to implement them. The Steenskiste design utilizes a single FPGA for all the computations and decides to make the tradeoff of increasing the number of radios by reducing the accuracy of the emulation. His design was focused on testing IEEE 802.11 transmitters and receivers and it does exceptionally well at that. In fact they mention using GTX transceivers for networking but determine this approach will not work because of the propagation delay. For IEEE 802.11 the propagation delay must be very small because the waveform is designed to operate over very short distances; however, the wireless devices that this thesis is targeting are designed to operate over much larger distances enabling a longer acceptable propagation delay in the emulation. It was found that by bundling the transceivers in groups of four, the propagation delay was significantly decreased as shown in Section 4.3.2.2.1. This shows that the delay for the transceivers is significantly smaller than the delay added by the analog hardware, and could be potentially used for IEEE 802.11 devices.

while several papers claim to scale, none of them discuss capabilities of expanding to multiple FPGAs which is necessary for making an accurate large scale DWCE. Borries has done an excellent job utilizing hardware that has the capability of scaling to multiple FPGAs, however, their discussion of this was lacking and by examining their hardware it will not be possible to utilize the number of FPGAs required for an accurate, wide-band, 16 or 32 radio system [4]. Lastly, studies have been performed with the Berkley Emulation Engine (BEE), which is an architecture that uses a cluster of FPGAs. These papers discuss the use of this hardware as a radio test bed [19]. They have also discussed the capabilities for performing real-time large-scale hardware emulation of radios [8, 15]; however, they have not discussed implementing a DWCE on the BEE.

2.4 WCE Validation Related Work

Another focal point of the thesis is the validation study. The goal of the validation study is to determine the emulation produced by the DBES as compared to a real-world environment. In almost every paper describing a WCE, the researcher has presented some evidence supporting the fidelity of the emulator. A common graph shows the power level of the received signal (represented as Signal to Noise Ratio (SNR) or attenuation of the system) versus the error rate of the receiving radio [10] [18]. A likely precedent for this approach is due to work seen by simulation papers. These papers do not have hardware for an emulation, but instead simulate a waveform in a simulated channel and show various charts depicting performance of the waveform [31]. Again, charts that compare error rate vs. attenuation or error rate vs. Doppler are common.

The most comprehensive published work done in this area is performed by Steenkiste and his students at Carnegie Mellon [14] [32]. Most of his papers have several charts showing various scenarios vs. radio performance. One of the papers that has the most data is written by the Steenkiste group on the experiences they have had using their emulator [3]. Some of the tests include: performance metrics of IEEE 802.11g under different rates and different path losses, packet reception rate vs. distance, throughput comparisons between the emulator and simulations, and performance of the radios under fading and multipath profiles.

All these different papers show a specific radio's behavior under test for a specific scenario. While this may lend confidence to that particular case, it is not possible to determine how the emulator will perform under different circumstances. None of the papers say anything about the limitations of the system, and only present good working behavior. The goal of the first two parts of this validation study is to not be radio specific and not to judge the fidelity of the emulator by the radio under test. Instead, the goal is to gather performance and noise metrics of the emulator itself, not the RF device. For this there is no precedent. Only after the emulator has been well defined and experimentally confirmed will the observational test be conducted that shows radio behavior.

CHAPTER 3: DBES DESIGN

This chapter describes the design of the DBES. The first section is an overview of the systems and how they are put together. The next section is on the signal path design, which is the central component for the system that performs the computations to transform the signal. The purpose for every other component in the DBES is to either get data to or from this signal path component. The third section is the hardware design, which describes the actual boards, chassis, and cables utilized to build this system. The creation of this hardware was not trivial as there were no commercial solutions for any of the boards we needed. Custom solutions had to be determined, developed, and fabricated for each component. A picture of the final DBES system is shown in Figure 3.1. It is included here to give the reader an idea of the size of the system and provide a reference for the rest of the chapter. The fourth section is on the scenario compiler. The fifth section is on the DBES control graphical user interface. The final section is a use case example to demonstrate how to use the DBES.

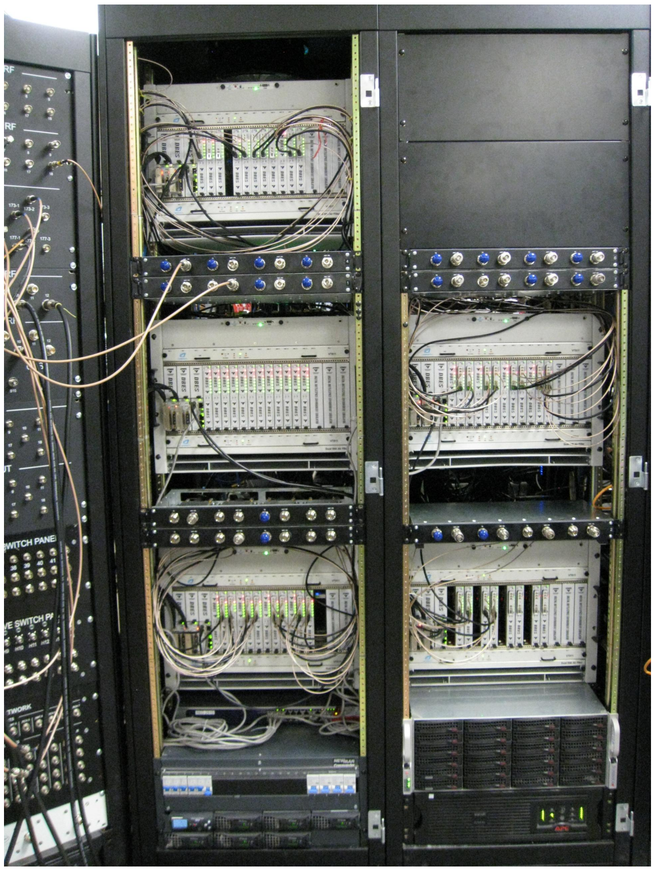


Figure 3.1: Picture of the completed DBES

3.1 DBES System Overview

This section describes a high-level technical view of the DBES. It starts with a detailed description of the factors that influence the scalability of a system then discusses the high level terminology used throughout this thesis. Next it describes the four major components of the DBES: the analog front end, the radio and compute nodes, and the network design.

3.1.1 Factors in Scaling a Digital Wireless Channel Emulator

As stated before the first consideration when scaling a DWCE is the amount of computations the system requires. This amount is primarily dictated by three components, the number of signal paths, the signal bandwidth, and the fidelity of the emulation. Each of these three components will be identified. First, the signal bandwidth simply refers to the amount of spectrum that will be emulated by the system. The RF signal is converted to a baseband signal in order to process the data. The more bandwidth this baseband signal needs, the higher the data rate must be to accurately represent the signal. A higher data rate means more data to process and thus more computations to perform. Therefore, the amount of the signal bandwidth that is to be emulated is directly proportional to the amount of computations that must be performed.

The second consideration is the number of signal paths. *A signal path is defined as the emulation of a signal between a transmitting source and a receiving sink.* This term "signal path" is used frequently throughout this thesis. Note that for a pair of radios there exist two signal paths between them. The amount of signal paths is determined by the number of channels used and the amount of clustering done. In the worst case where every node can talk to every other node, if there are N nodes there will be $N \times (N-1)$ signal paths. Clustering is a means of reducing this relationship by eliminating links that are not utilized. This could be done in any manner of different ways, but usually the goal is to segment groups based on emulated physical location or spectrum utilization. Currently, the DBES will not utilize any clustering; however, thanks to the reconfigurable nature of the FPGAs, it could be added at any point based on a user's needs. For every signal path in the system there must be an equation to perform the computations. Therefore, the amount of signal paths is directly proportional to the amount of computations.

The final consideration is accuracy. In general, the more accurate the model is to reality, the more computations it must perform. Specifically with regard to the tapped delay line model utilized (see Section 3.2), the number of propagation paths are directly proportional to the number of computations that are required. In general, computing more propagation paths results in a more accurately emulated scenario. Adjusting this number is an easy way to directly affect the number of computations.

Additionally, for each propagation path emulated there are several computations that must be done. Some of these, like interpolating the delay parameter and signal for a more precise Doppler, could be removed if such effects were not a concern. Currently the full model performs about fifteen computations per propagation path. The bare minimum would be two, one for a multiplication and another for an addition per propagation path.

By combining the three considerations above, a general relationship can be determined between the number of computations and these factors 3.1.

$$computations \propto N^2 \times SBW \times PP \times PC \tag{3.1}$$

Where N is the number of nodes, SBW is the Signal Bandwidth, PP is the number of Propagation Paths and PC is the amount of Propagation Computations. This relationship gives an excellent insight into the size of a given system. It describes how you can increase the number of nodes for a given DWCE at the cost of signal bandwidth and/or emulation accuracy and maintain the same amount of hardware utilized.

Aside from computations, the other consideration to account for in determining the scale and requirements of a DWCE is the networking resources. As described in the background section, at some scale it will be necessary to network multiple processors together. This requires attention to ensuring sufficient throughput for the real-time system. There are various methods of mitigating the amount of network traffic that are dependant on implementation. This thesis will describe the network that the DBES utilized in Section 3.1.5. The network is not a standard network but ensures it will take two hops to get from the input to the output of the system.

This is how initial sizing estimates were performed for the feasibility study. Later, in the scalability study, the actual size of each component was used in addition to timing and resource overhead metrics to determine the scale of a system.

3.1.2 System Terminology

The full scale system is designed around the SPIRIT architecture [28]. The SPIRIT cluster consists of 64 ML410 boards, each having a Virtex 4 FX60 FPGA. These FPGAs are connected by high-speed Multi-Gigiabit Transceivers (MGTs) giving very large communication bandwidth between the FPGAs. This cluster has been used for studying high performance computing topics and applications such as hardware filesystem, the BLAST bioinformatics application, and resiliency in exascale computing. After a brief investigation it became apparent that this architecture was ideal for a large scale DWCE. This is due to the large number of FPGAs, allowing for the computing resources necessary, and the high network bandwidth between FPGAs allowing for the communication of the massive amount of data required when multiple chips are used. The only missing hardware component is an Analog Front End (AFE) or a board to convert the RF signals into digital signals and vice versa. The high level design and terminology is shown in Figure 3.2.

This overview depicts many of the critical components to the system and then traces out the path that a signal takes through the DBES. The major systems in the DBES are the Reconfigurable Computing Cluster (RCC), the analog front ends and the Scenario Compiler (SC). The reconfigurable computing cluster is the cluster of FPGA nodes that forms the hardware on which the processing takes place. The inputs to the RCC are the digitized RF signals at baseband, the parameters describing how to modify the RF signals, and system configuration and calibration information. The outputs from the RCC are the digitally modified RF signals at baseband and the error reporting signals. The main components of the RCC are the FPGA nodes which are designated as Radio Nodes (RN) or Compute Nodes (CN) based on their functionality. The purpose of the FPGA nodes is to read in signal data, process it with parameters, and output that data. This process is described in detail in Section 3.1.4. Each FPGA node contains Signal Path (SP) components described in Section 3.2. They

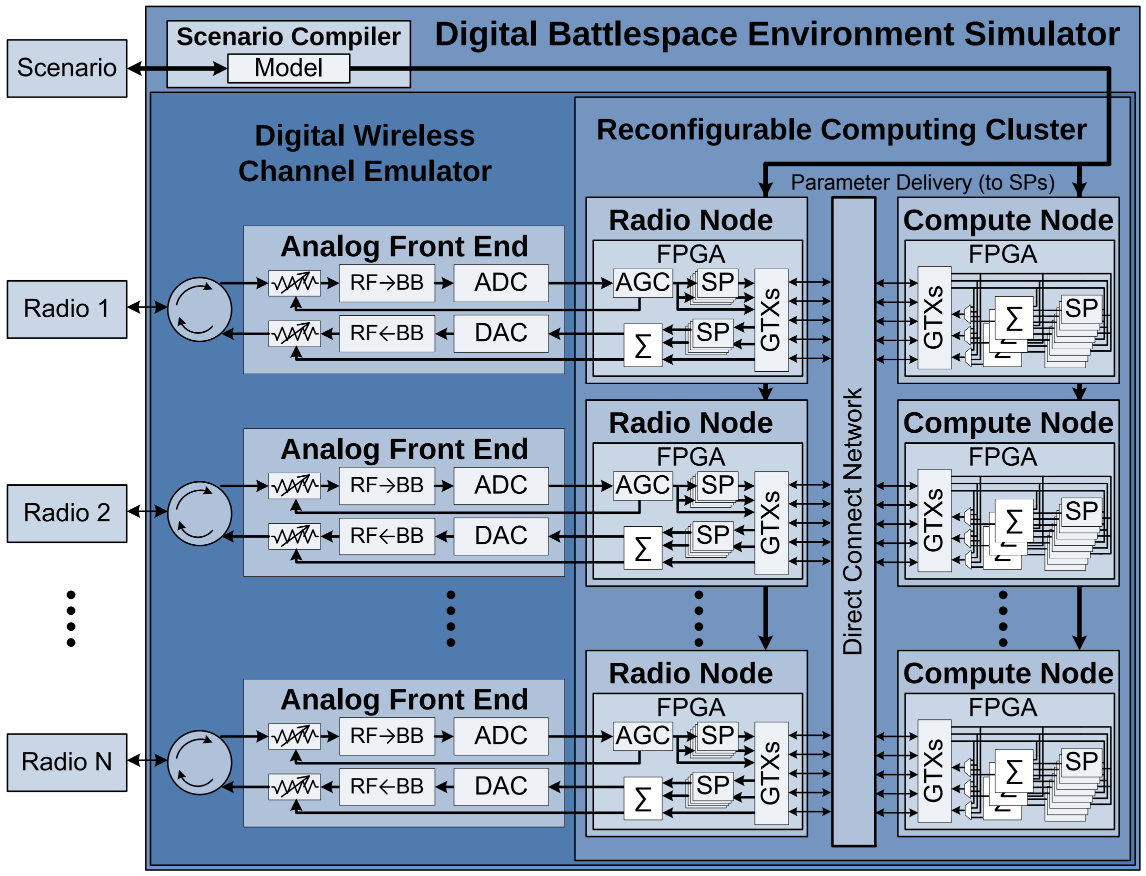


Figure 3.2: High level DBES schematic. Note that duplicated lines and components are meant to signify more than one and not an exact count.

are the heart of the entire system as they perform the computations to modify a single input signal to produce a single output signal. The RNs are connected to the CNs through a direct connect high speed network formed using the Multi-Gigabit Transceivers (MGTs) called GTX transceivers as detailed in Section 3.1.5.

The analog front end is what enables the RCC to be a wireless channel emulator. This handles the digitization and reconstruction of RF signals as described in Section 3.1.3. Critical pieces of this hardware are the Analog to Digital Converter (ADC), the Digital to Analog Converter (DAC), the RF to Baseband (BB) converter, the BB to RF converter, and the variable attenuators on both the transmit and receive paths. Exterior to the AFE are splitter/combiner networks that separate out the transmitted and received signals.

The scenario compiler takes user input in the form of radio locations, terrain data, building information, antenna type, etc. and compiles that information into parameters that can be used by the DWCE. This process is done before the emulation takes place so that when the DWCE runs, the parameters are already computed and can simply be fed to the system. This is largely due to the fact that the SC can take days to compile a scenario. This component is not a principle focus of this thesis. The development was done by a third party who had a base design that was very close to the required specifications. The SC is where the user first starts to implement a test by telling the SC what the environment will be. For purposes of the next few paragraphs it is assumed that a user has input the scenario, the compilation has taken place, and parameter data has been loaded on all the FPGA nodes.

A signal starts as an RF signal transmitted from an RF device. This signal goes through the splitter network and reaches the receive port of the AFE. The AFE has a variable attenuator that is controlled by its RN which attenuates the signal so that it is in the correct amplitude range. Next, the signal is converted to baseband and digitized by the ADC. This data is then sent to the FPGA on the receiving RN. The first signal component in the RN is the gain control which works with the variable attenuator. The signal then is processed by a SP core or not. This decision and all the routing decisions are based on a predetermined layout that is found by the load balancing algorithm described in Section 3.1.5. Note that any given signal path (from transmitting RF source to receiving RF sink) only passes through one SP component. After being processed or not, the signal is passed through a GTX to a compute node.

The compute node receives the signal and again makes a determination on where it will go next. If it has already been processed by a signal path core it will immediately go to the appropriate sync-and-sum (depicted by Σ in Figure 3.2) where it will be synchronized with other signals that have been processed and summed together. If it has not then it will either be processed in a SP core and then summed by the sync-and-sum component or simply forwarded on. The final step in the compute node is to be output over a GTX to the receiving RN.

The signal now has reached the receiving RN and has it's final routing decision. If it has already been processed it will proceed straight to the sync-and-sum component and if not it will be processed in a SP core and then sent to the sync-and-sum component. At this point all signals from all the other radios will have been combined by summation and the transformed signal is ready to be output to the receiving RF device. The last digital stage is to check the amplitude of the signal and set the transmitting attenuator appropriately. The signal is then sent to the AFE where it is reconstructed in the DAC, upconverted to RF, and then output from the DBES.

3.1.3 Analog Front End Design

The AFEs are the intermediary between the RCC and the RF devices and have two paths as shown in 3.3. The receive path takes the transmitted RF signal from the RF device, converts it to a baseband signal, digitizes that signal, and sends the digital signal to the RCC. The transmit path takes the modified digital signal from the RCC and converts it to an anolog signal, then upconverts that analog signal to RF. This setup is known as direct RF conversion because it does not use an Intermediate Frequency (IF) as is common with heterodyne transceivers. There are many advantages and disadvantages to this setup which will largely be discussed in the fidelity chapter 4.3. In general, this approach requires less hardware and filtering which makes it a more inexpensive option; however, there are a lot of implementation difficulties that require precise calibration to eliminate noise and leakage.

The difference between this AFE and that of traditional radios is significant but cannot be seen in the schematic. Because the WCE AFE transmitter is connected to an RF devices receiver and vice versa, the power levels that are dealt with are opposite. The AFE transmitter must be able to send out very small signals just as an

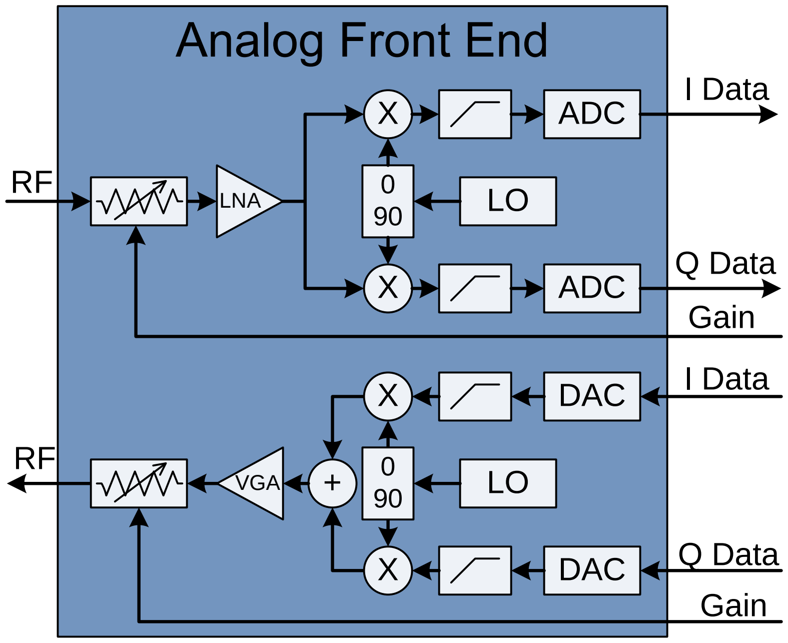


Figure 3.3: Analog front end schematic

RF devices receiver must be able to receive very small signals. A typical RF device will transmit at the highest power level possible, which means that the receive side of the DBES AFE does not have to be as sensitive as traditional RF device receivers. Because of this difference the gain control is a very important aspect of the design. The transmitted output signal can vary over a very large range which is the reason for the Variable Gain Attenuator (VGA) on the transmit path. A normal RF device would have no need for this component but it is critical to the DBES.

3.1.4 Radio and Compute Node Design

The radio nodes and the compute nodes are descriptions of the application running on each FPGA board. The only difference in hardware between a compute node and a radio node is that a radio node has an analog front end attached to it. This means that the compute nodes sole duty is to receive, compute, and transmit signal data whereas the radio nodes must have additional functionality to interface with the AFE and it's components.

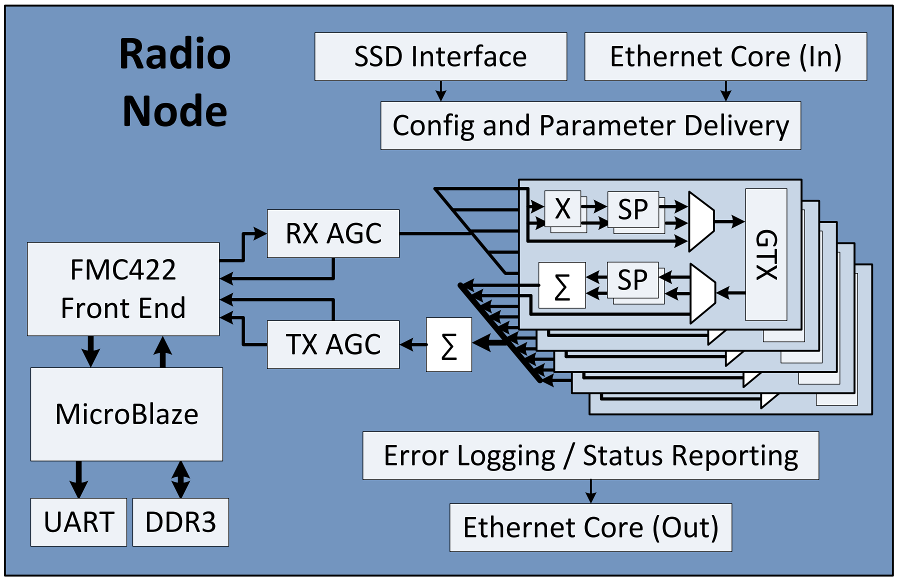


Figure 3.4: Radio node schematic

Figure 3.4 and Figure 3.5 shows the high level design for the radio node and compute node respectively. The primary purpose for both these nodes is to get signal data in, processes it, and route it out. The signal input and output for the RN comes from both the AFE and the GTXs whereas the signal input and output for the CN are solely through the GTXs. This section will describe the high-level functionality of the CN and RN.

The FPGA nodes have several major components: the FMC422 Front End, the RX and TX gain controllers, the parameter delivery system, the error logging system, the routing system, the sync-and-sum components, the GTXs, and the signal path cores. The SP cores and the GTXs were described in Section 3.1.2 and will not be discussed here.

The FMC422 front end includes all the board-level drivers to interface with the AFE as well as a MicroBlaze that controls the initialization and calibration of the AFE. The MicroBlaze runs a stand-alone C program largely written by the manufacturer of the FMC422 and adapted to be included with this application. The program performs various initialization and calibration tasks prior to running the application.

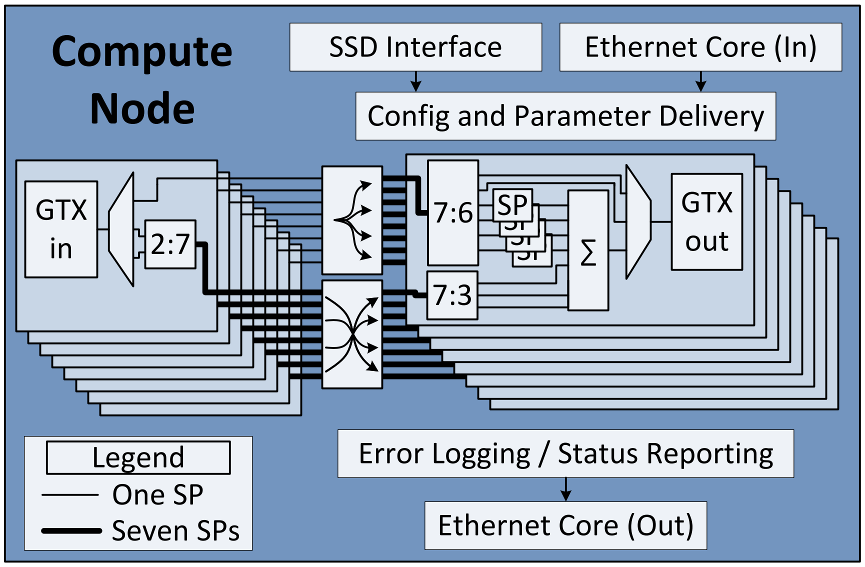


Figure 3.5: Compute node schematic

These tasks include calibrating the gain of the 422, the LO frequency, the IQ balance, and initializing DAC and ADC. The program resides in DDR3 memory so as to not take up valuable BRAMs and communicates status messages over the UART.

The parameters are stored on a Solid State Drive (SSD). These parameters are loaded onto the SSD by another VHDL build and read off the SSD by this application build. Also included on the SSD are initialization parameters such as which SPs are on this FPGA, where to route signals, settings for the gain controllers, settings for the AFE, etc. All the information is located at a known address in memory by the loading build and is sequentially streamed off the SSD by this build. One detail is the throttling of the data is done by each SP core. They dictate when another parameter set is required and the SP Params controller responds.

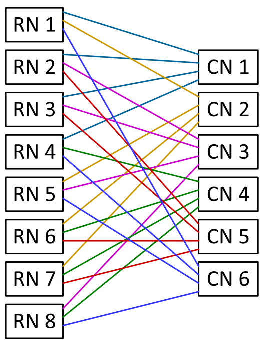
The Error Logging core is a deceptively important piece of this application. This core can report various debugging information such as sync-and-sums not synchronizing or GTX core failures; however, it also has a more critical application role. As will be seen in the fidelity study, many problem can arise due to different scenarios being introduced to the system. Based on accumulators and multipliers overflowing or underflowing or problems with the gain control stages, these corner cases can be inferred. Reporting these corner case occurrences is critical to confidence in the scenarios validity.

3.1.5 Network Design

The network design for this system is a deceptively challenging problem. Every radio connected to the system has the possibility of communicating with every other radio and as such there must be a constant flow of data that can reach from every radio node to every other radio node. This means that the network must support a streaming all-to-all broadcast in real time.

The network is formed utilizing the transceivers on the FPGAs. There are eight bi-directional network connections per FPGA and each FPGA will be directly connected to another FPGA through one of those connections. Each connection will bundle four GTX transceivers together to allow for the necessary throughput. This has the added benefit of reducing the propagation delay from the transmission to the reception of the data. The goal is to allow every radio node to communicate with every other radio node in as few hops as possible. The scheme used to do this is to have each compute node connected to eight different radio nodes such that no two radio nodes are connected and no two compute nodes are connected. The challenge is to determine how the connections should be made.

This turns out to be an intractable problem with many solutions. Initially, several different algorithms were used and after many failed and some successful attempts completing non-optimal solutions, we found that this problem could be solved optimally using a (v, k, t)-Covering Design where v is the number of radio nodes, k is the number of connections, and t is two indicating the number of groups (RN and CN). Solving a traditional (v, k, t)-covering yields many sets of k elements. There is a website dedicated to providing known solutions called the La Jolla Covering Repository [13]. In our case the number of sets is the number of compute nodes needed,



(a)

Solution to (8,4,2)-covering problem

CN	RN_A	RN_B	RN_C	RN_D
1	1	2	3	4
2	1	5	6	7
3	2	3	5	8
4	4	6	7	8
5	2	3	6	7
6	1	4	5	8

(b)

Figure 3.6: (V,k,t)-network example: (a) Shows the application solution for 8 radio nodes and 4 connections which utilizes 6 compute nodes. (b) The numerical solution to (8,4,2)-covering. Each row designates the four radio nodes that the enumerated compute node is connected to.

and the elements of a given set enumerate the radio nodes that the compute node is connected to. There are different methods for solving a given set of v, k, and t parameters and no single method can solve arbitrary values of these parameters. Additionally, for many combinations of parameters it is not possible to create the covering design. Fortunately for our design it is possible to solve the covering problem. Using the repository, a solution to the (32,8,2)-covering problem is available online which has 21 sets each containing eight numbers. This result means that 21 compute nodes are needed to complete the network and each of those compute nodes will be connected to eight different radio nodes. This is an optimal solution resulting in 53 total FPGAs needed to realize the application.

For brevity, an example is shown for an eight radio node system with only four connections per FPGA board. Figure 3.6(a) shows a (8,4,2)-covering solution as it applies to this application. The repository was used to find that six compute nodes are needed for this solution. While it appears to be random connections, in fact this optimally solves the problem of connecting every radio node to each other with two network hops. The table of data in Figure 3.6(b) shows the set representation of the solution. Figure 3.7 is a picture of the back of the DBES that shows the miniSAS-HD cables being routed. It is included here to visually show the complicated and routing that the (32,8,2)-network results in.

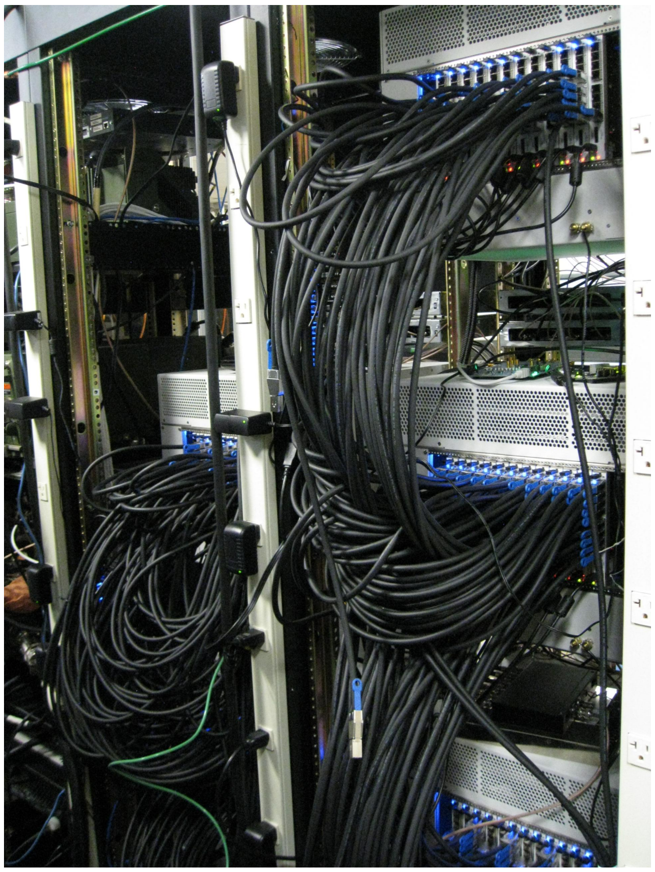


Figure 3.7: Picture of back of the completed DBES showing the routing of the (32,8,2)-network

The last component of the network design is how the signals are routed on each FPGA. How these signals are routed have a major impact on the load required for each GTX. To illustrate this point lets assume that all signal path components are located on the compute nodes. For the 32 channel system with 64 FPGA nodes and 32 compute nodes, this would mean that there needed to be 31 signal paths per compute node assuming an ideally balanced system. In this case each compute node would be reading in data from eight different radio nodes, running this data through various signal path components, and then outputting the resulting data back out. In the worst case there are seven signals destined for a single radio node. Fortunately, even in this case all those signals can be summed together to produce only a single signal that needs to be sent on the transceiver. This means that in this ideal situation there will always be one signal sent and one signal received from each transceiver at the sample rate. The issue with this is the computations are not evenly balanced between the radio and compute nodes. While the radio nodes have some overhead in dealing with the AFE, they have plenty of space for additional SP cores.

Balancing the Signal Path Core (SPC) load then requires moving SPCs from the CN to the RN. The signal path core is the VHDL component that performs the processing for a given signal path. Let's examine the load balancing case for a single SPC. On a radio node there are two distinct paths the signal takes. The beginning path is defined as the path where signals are received from the AFE and sent out to the GTXs. The ending path is defined as the path where signals are received from a CN and sent to the AFE. When moving a SPC from a CN to an RN it can either be put in the beginning path or the ending path. If the SPC is put on a beginning path then there is an additional signal that is being produced which must be sent to a CN. Therefore, in this case the amount of network traffic increases from a RN to a CN by one signal path. If instead the SPC is moved to the ending path, an unmodified signal must be sent to the RN that has not gone through a SPC and therefore cannot be summed at the CN. This results in additional traffic through the network from the CN to the RN of one signal path. Therefore, it is possible to balance the SPC load by moving SPCs from CNs to RNs at the tradeoff of adding load to the network. Moreover, the network load from RN to CN and from CN to RN can be balanced by choosing to place a SPC in either the beginning or ending paths of the RN.

Balancing both the network and SPC load could be done by hand; however this would take a long time so an algorithm was designed to automate this process. This algorithm is depicted in Figure 3.8. The load distribution algorithm starts with an

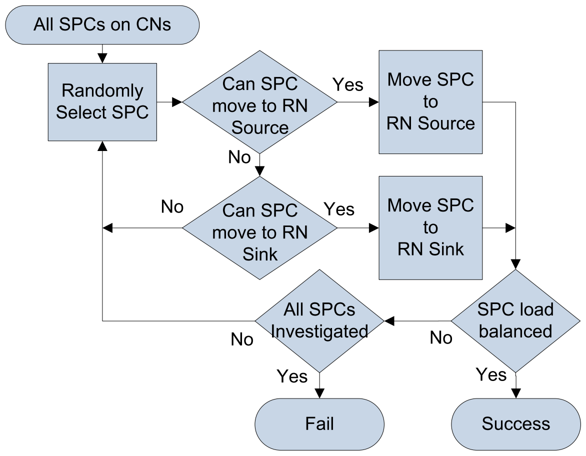


Figure 3.8: Signal path and networking load balancing algorithm

initial state where all SPCs are on the CNs and then proceeds to randomly select an SPC and either move it to it's beginning RN path, it's ending RN path, or leaving it on the CN. It proceeds through each SPC and determines a possible location until the loads are balanced. Once a SPC has been moved to a RN, it is considered to be locked in place and it is not moved again. In order to assure the algorithm will finish, it is looped a set number of times through every SPC. The SPCs are randomly ordered to help distribute the load. Also, the first time through, only one SPC can be on each beginning and ending RN path and this increases by one for every iteration. Again, this has the effect of load balancing. This is good because it ensures a specific number of steps to be taken, even if a solution is not found. The algorithm is not guaranteed to find any solution, or an optimal solution for that matter. In fact it can succeed or fail based on the random seed that is given to the ordering of the SPCs. This was an easy solution that works in many cases for a complicated ordering challenge and research on it was halted after finding a solution that worked.

After the network is formed by the (v,k,t)-network, and the location of the signal paths have been finalized, the signals in each FPGA must be routed to and from their respective GTX connections. To complicate matters a different configuration can occur on each FPGA. It is not feasible to build a new FPGA image for each

FPGA, so a configurable routing system was developed that attempted to use as few resources as possible. The RN and CN designs shown in Figures 3.4 and 3.5 shows this routing. This system includes several components including the signal path to GTX mux, the GTX to signal path mux, the crossbar switch, and the sync-and-sum component. Each is a critical piece of the routing.

Fortunately, not all of these components have to be configured at runtime as the SP-to-GTX mux and GTX-to-SP mux can be identical from node to node. They are designed with predetermined signal path connections. If a connection has a ready signal at it's input then the path is used, otherwise nothing is transmitted. The ready signal is a single bit signal that is sent along with the data that when high indicates that there is data ready to be processed. These two systems behave just as the name suggests, they multiplex signals over the GTX connection on the transmitting FPGA, and then multiplex them back out on the receiving FPGA. GTX is a serial transmission line and as signals come in to the SP-to-GTX they are queued to be sent over the line. Each signal path has a pre-determined location in the queue so that on the GTX-to-SP side, no control messages are needed to know which signal goes where. The biggest challenge with this component was dealing with the clock correction that happens on the GTX and ensuring that a steady stream of data leaves the GTX-to-SP component. The clock correction event occurs so that the GTX cores can synchronize their clocks. The result is that user data is blocked for several clock cycles. Normally this is not an issue, but with this application the data must arrive every Nth clock cycle, where N is the multiple of the FPGA clock rate as compared to the ADC clock rate. To ensure this steady stream even when data is stopped, a buffer was used on the output of each signal path that would only send out a signal every Nth sample. During the clock correction the buffer's data would be consumed until the clock correction period is over. At this point, there is data buffered in the SP-to-GTX that gets sent over the GTX connection quickly. This fills the buffers on the GTX-to-SP side again because the data arrives faster than it is sent out. Interestingly, the amount of buffering to be done is calibrated by doing nothing. On the first clock correction event, the GTX-to-SP buffers are completely emptied, and data is missed. After that, the SP-to-GTX sends over it's buffered data and fills the GTX-to-SP buffer with the right amount of excess data. This only happens once, just after the DBES is started so it does not affect a scenario being run.

The sync-and-sum and the crossbar switch need to be uniquely configured on each FPGA. The sync-and-sum has been defined in passing throughout this section, but to reiterate here: it synchronizes the signals coming to the component and then sums them together to be output. The synchronization occurs at startup and is determined by when the ready signals first arrive to the component. The component has buffers on each of the input lines and will pull from all the buffers at the same time: when all of the valid signal paths have a ready signal. Because all the ready signals from each input signal are started at the same time, simply waiting until each valid signal path has a ready signal arrive at the sync-and-sum component calibrates for any differences in propagation delay between each of the signal path signals. This is where the configuration comes into play; each sync-and-sum component must be told which input paths it will receive ready signals from. The last component is the crossbar switch, which performs exactly as it's name suggests. Any input can be mapped to any output based on configuration data sent to the switch. The only major challenge with this component is timing because it could be routing signals from one side of the chip to the other side of the chip. Both these components are configured by configuration parameters that are read in from the solid state drive at startup. They can also be programmed by ethernet at run-time, but that is typically only for debugging operations.

3.2 Signal Path Design

Any DWCE is built around implementing an equation to modify the input signals to produce one or more output signals. The equation implemented in this design is a tapped delay line equation and is shown in Equation 3.2. Other methods utilized for channel emulation are Sum of Sinusoids (SoS) equation and filtered white Gaussian noise. The SoS equations are frequently utilized in traditional processors because they can be implemented very efficiently to reduce run time or increase simulation capability. The aim of both these models is to produce a signal with the same autocorrelation and Doppler power spectrum as a Rician or Rayleigh fading channel. The issue with these fading channels is that they are statistical approximations of the signal path environment. What is desired in this implementation is to reproduce the environment as accurately as possible. This precludes the use of any statistical models.

This section first describes the theory behind this signal path design and then discusses the core that was created to implement the theoretical model. Finally, the Doppler Shift implementation is discussed as special care is needed to ensure a clean signal is generated.

3.2.1 Tapped Delay Line Theory of Operation and Derivation

This TDL equation (and the hardware and software used to implement it) maps every input signal to one output signal.

$$y_\nu(i,d,\alpha,\omega,t) = \sum_{m=1}^{M} \alpha_m \left(x_i(i - d_m(i))\right) e^{\mathrm{j} d_m(i)\omega_{rf}} \tag{3.2}$$

The α parameter is the attenuation and the d parameter is the delay. These are summed over m multipath elements. Each multipath describes a single propagation path the transmitted signal takes from its source to the receiver. In urban models, because of reflections and refractions from buildings, there can be many multipath elements per signal path. The exponential term performs a frequency shift make an adjustment for the Doppler effect that will be described in more detail in the next paragraph. An excellent description of the equation resides on page 183 of [27] and the derivation of this equation follows in the remainder of this section.

A Signal Path refers to the RF environment between a transmitting and receiving radio. The Signal Path Equation describes the mathematics behind altering a transmitted RF signal to account for the real world environment. Equation 3.3 is a Tapped Delay Line Model and has been described in many academic journals and textbooks.

$$y(t) = \sum_{m=1}^{M} \alpha_m(t)x(t - d_m) \tag{3.3}$$

The goal of this section is to mathematically show that the DBES implements equation 3.3. Because of hardware and real-world limitations the equation directly implemented by the DBES (equation 3.2) is not identical to the desired result. Several operations are done that can be shown to be mathematically equivalent to the original equation within a specified bandwidth. First, the input signal is brought down from the RF frequency of interest to baseband. This is done in the analog domain and then digitized into In-Phase and Quadrature components. The attenuation for this step is set for a given input in order to maximize the range of the Analog to Digital Converters (ADCs). This inherently limits the RF frequency range of interest around a bandwidth that is predetermined and fixed for the duration of a experiment. This conversion is described in equation 3.4.

$$x_s[n] = x(nt_s)e^{-\mathrm{j}\omega_{RF}nt_s} \tag{3.4}$$

Note that the signal is now in the digital domain. Also, filtering is done on these signals but is omitted here for clarity sake. It is assumed here that appropriate filtering has been performed to ensure no aliasing has been introduced. After the signal has been digitized it can proceed through the signal path modifications as shown previously in equation 3.2. The final step is to send the modified signal through a DAC and upconvert it to RF which can be described by equation 3.5.

$$y(t) = y_s[t]e^{j\omega_{RF}t} \qquad \text{where: } nt_s < t < (n+1)t_s \tag{3.5}$$

The remainder of the proof is substituting each of the equations 3.4, 3.2, and 3.5 together and simplifying to show that it is identical to equation 3.3. The first step is substituting the downconversion equation 3.4 into the digital signal path equation 3.2. The result is shown in equation 3.6 with a further simplification shown in equation 3.7.

$$y_s[nt_s] = \sum_{m=1}^{M} \alpha_m[nt_s]x(nt_s - d(nt_s)) \times e^{-j\omega_{rf}(nt_s - d(nt_s))} \times e^{-j\omega_{rf}d(nt_s)} \tag{3.6}$$

$$y_s[nt_s] = \sum_{m=1}^{M} \alpha_m[nt_s]x(nt_s - d(nt_s)) \times e^{-j\omega_{rf}nt_s} \tag{3.7}$$

The result of equation 3.7 is $y_s[n]$, which is a representation of the signal after it has passed through the signal path. Note here that the signal is not at the correct frequency at this point. Equation 3.8 shows the substitution of the upconversion equation 3.5 into equation 3.7. Equation 3.9 further simplifies equation 3.8.

$$y(t) = \sum_{m=1}^{M} \alpha_m(t)x(t - d(t)) \times e^{-j\omega_{rf}t} \times e^{j\omega_{rf}t} \qquad \text{where: } nt_s < t < (n+1)t_s \tag{3.8}$$

$$y(t) = \sum_{m=1}^{M} \alpha_m(t)x(t - d(t)) \qquad \text{where: } nt_s < t < (n+1)t_s \tag{3.9}$$

The differences between the ideal equation 3.3 and the derived equation 3.9 are due to sampling, but this shows that they are indeed theoretically equivalent. There are many implementation factors and corner cases that will result in errors. This is one primary focus of the validation study, to identify and explore these factors. The point made here is that in an ideal system the equation used produces a TDL model in the analog world.

3.2.2 Signal Path FPGA Implementation

The biggest obstacle to the creation of a large scale DWCE is the number of computations that must be performed by the system in real time. As addressed in the introduction the amount of computations depends on the number of radios and the signal bandwidth. Specific to this equation, it is also proportional to the number of propagation paths emulated per signal path. By examining this relationship it can be seen that the same basic components are repeated many times. This fundamentally means that by designing this component it becomes possible to get an accurate estimate on the size of any DWCE you might build with this equation. For this reason, a large effort was done to implement this block in VHDL as efficiently as possible. It took several iterations to find the best design and ultimately one design was found that can optimally pipeline and parallelize the signal path to use a few resources as possible. A major challenge is that the optimization is dependent on the signal bandwidth, the number of multipath computations, and the FPGA processing clock rate. First the system was designed as a completely pipelined system. Assuming a sufficiently fast FPGA clock rate, all the computations could be pipelined such that all the multipath calculations including the attenuation, the delay look up, and the frequency shift can utilize the same hardware.

There are two issues with an entirely pipelined implementation. First, the adder used to accumulate all these signals must have a delay of one. Unfortunately, running at even a moderate clock rate, a typical one clock cycle delay adder does not exist for the bit-width required. A solution to this problem is based on the approach in [24] that utilized 3:2 compressors. The hardware for this component is identical to a normal adder having N full adders where N is the number of bits, with the exception that the carry logic is not connected and is instead pulled out as an output. This alleviates the timing constraints of the carry logic and allows for a sum to occur for any bit-length of number in one clock cycle. The result is such that the sum of the

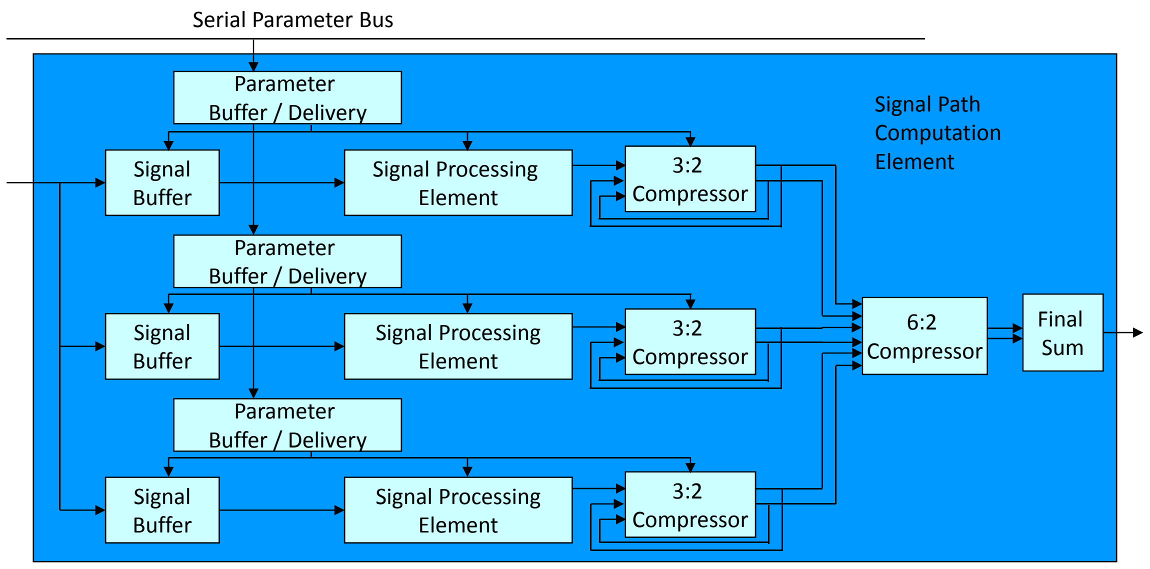

Serial Parameter Bus
Parameter Buffer / Delivery
Signal Buffer
Signal Processing Element
3:2 Compressor
Signal Path Computation Element
6:2 Compressor
Final Sum

Figure 3.9: Signal path design

output two numbers is identical to the sum of the input three numbers. This allows for an accumulation to occur with every clock cycle and once the accumulation is completed a normal sum can occur on the final two numbers to get the result. This utilizes much less hardware than other alternatives such as an adder tree.

The second issue with the fully-pipelined implementation is that the FPGA clock speed must be greater than the signal bandwidth multiplied by the number of multipath computations. Utilizing in-phase and quadrature components allows for a sample rate equivalent to the signal bandwidth. With modern FPGAs a clock rate of 300 MHz or higher is possible for this design; however, this is not fast enough for many scenarios. A typical scenario consisting of a signal bandwidth of 30 MHz and 30 multipath calculations would result in a clock rate requiring 900 MHz. Also, future wide-band waveforms will support bandwidths over 100 MHz. To support these concerns it must be possible to parallelize the computations.

The most straightforward solution to parallelization turns out to be the best. By simply replicating everything in the signal path including the signal buffer, parameter buffer, signal processing element and the 3:2 compressor, parallelization is achieved.

An alternative design was considered that used only one signal buffer. By utilizing the taps of multiple BRAM elements in a singular signal buffer it is possible to only store the signal once and still have the throughput necessary. This design was created; however, the size of the logic on the FPGA turned out to be too large because of the necessity of dealing with memory contention where all the indices of interest for a given sample cycle were in the same BRAM. The most efficient method to sum the resulting signals was to utilize an N:2 compressor where N was the number of paths multiplied by two. The N:2 compressor was formed by making a tree utilizing $N-1$ 3:2 compressors. The final design for this component given a parallelization factor of three is shown in Figure 3.9.

The parameter delivery system is a critical sub-component and was designed to allow for rapid parameter changes while a signal is being processed. A specialized parameter memory system and delay chain was utilized to address timing concerns with the delivery of parameters to each component. This system allows for parameter changes well in excess of 1 MHz with no change to the processing components. This was considered to be more than sufficient as an update rate of 1 kHz should be sufficient to replicate real-world scenarios. The VHDL was written utilizing generics so that any configuration of pipelining and parallelization could be utilized. In this way it can be tailored to any given situation.

3.2.2.1 Doppler Shift Implementation

The Doppler Shift is one of the important channel effects this equation emulates. A Doppler shift is a spreading or contacting of the frequency space. To accomplish this we use the indexing to the signal buffer. A Doppler shift can be thought of as samples arriving consecutively earlier or later to the receiver as compared to a static scenario. Therefore the effect of changing the delay from sample to sample is a Doppler shift. Just as in the real-world, as the delay changes this compresses or expands the signal produced by this equation. By simply taking the real-world delay and using that to index the signal buffer a Doppler shift is inferred. As mentioned in Section 3.2, the parameters are delivered with a certain frequency and the delay is interpolated between parameter periods so that there could be a different delay parameter for every sample period. This is key to producing an accurate spreading and contracting of the entire frequency space emulated. Using the delay in this manner works out very well because a concern with this is that you can run out of Buffer space (either addressing too early or too late) by doing a Doppler Shift. This is actually not a problem because if we constrain the parameter sets to model the real world then the delay will never go below zero and therefore we will never run out of Buffer space.

There are several issues with this implementation that must be accounted for: the delay interpolation, the signal interpolation, and the RF frequency correction. These factors make this equation much more difficult to implement in terms of coding and the amount of computations performed which has a dramatic impact on the size of the system as shown in the results section. Additionally, there are a few VHDL features that are utilized to ideally realize this implementation that are also described in this section.

The first issue is the delay interpolation. The parameter update rate is variable but is typically around one millisecond which means that the parameters update much more slowly than the sample rate. If the delay were not interpolated, a rapid phase jump would occur between the two samples that surround a parameter update. While over a long period of time, this statistically produces a Doppler effect, the time series of the signal output does not look correct at all as it would appear to be on the original frequency with sudden phase jumps. To solve this issue the delay parameter can be interpolated between parameter periods. This results in a smooth change in the delay at the signal's sample rate and a continuous signal with no quick phase jumps is the result. The question now is what type of interpolation should be performed. The two most basic interpolation schemes that could work in this situation are linear and cubic. Other schemes like cosine interpolation, while more basic than cubic in terms of ease of implementation, would not be appropriate as the delay is not periodic. The advantages of a cubic interpolation is the error is reduced and the resulting delay is differentiable whereas the delay produced by a linear interpolation would only be continuous. Ultimately, the linear interpolation scheme was used because the differentiability of the delay parameter is not an issue and the linear interpolation utilizes many times less resources than the cubic.

The second issue is the signal interpolation. Changes in delays on the order of 100's of picoseconds will have a noticeable effect on the Doppler Spectrum of a signal. The problem is the sample period of the signals is usually around 15 nanoseconds. This means that it is necessary to know what the signal level is in between samples. The only way of doing this is interpolation. The question of what type of interpolation is much more difficult for the signal and the delay; however, the resulting answer is still the same. Linear interpolation, while it produces the highest error, was much easier to implement and only takes up a fraction the amount of resources on the FPGA as compared to higher order interpolation schemes. This made the benefits for going to higher order interpolation schemes not justifiable. However, this is a point of investigation for the validation study and if the linear interpolation is not good enough, higher order schemes could be implemented. Both the noise generated by the Doppler Effect and the accuracy of the frequency selective fading can be improved by higher order schemes. Additionally, simulation of the system showed that digitally upsampling the signal to a higher sampling rate can also improve the noise seen but this comes at the cost of increased computation and memory requirements and increased propagation delay.

The last factor is the exponential multiplication. The term $e^{\mathrm{j}d_m(i)\omega_{rf}}$ is utilized to correct for the fact that the signal is at baseband. If the signal were digitized at the RF frequency then the term $x(i - d_m(i))$ would produce the correct Doppler shift.

However, the signal is not at RF, it is at baseband. In order to produce the correct shift when the signal is output and upconverted, the baseband signal must be shifted in frequency. The amount of shift is dependant on the change in delay and the RF frequency. Note that if a delay is constant this exponential multiply will only perform a phase shift. It is not until the delay changes (causing a changing phase shift) that the frequency shift will be seen.

There are two features that are implemented to realize this scheme in an efficient manner on an FPGA. The first is the delivery of ”time” to the delay interpolation and the second is the implementation of a buffer that can produce one write and two reads every clock cycle. The Delay Parameter Interpolation Counter component is incremented every sample period and is reset every new parameter period. The output (called interp out) is a fixed point number that represents a value between 0 and 1 inclusive. The idea is that beginning with a new parameter period interp out starts at 0.0 and counts up linearly every sample period until the next parameter period when interp out equals one before cycling back to 0.0. There are two key challenges with this implementation. First, the value of '1.0' must be represented by a power of 2. This is done by using a regular counter and then multiplying by the parameter period divided by the sample period. Second, you must insure that as soon as interp out equals 1 the next parameter period starts with the next signal period. Logic is built into this component to suggest to the parameter delivery component when it is time to update parameters. So long as the parameter delivery component has parameters, then it will update when told to do so. If it doesn't have parameters then the parameter delivery is not timed correctly.

The second feature has to do with the signal buffer. It is necessary to read out two addresses every clock cycle and potentially to write one in. This means that a traditional memory component cannot be used as it is not capable of this; instead, two memory components are instantiated. Data is written into the components in a ping-pong fashion where each consecutive sample entering the system is written to the opposite memory component as the previous sample. Because interpolation occurs between two consecutive samples, this ping-pong memory assures that data from both memory components are read every clock cycle. Based upon the read address, the buffer will output data at that read address and at the previous read address. These two data points are then interpolated according to the interpolated delay value.

3.2.3 Gain Control Design

This section discusses theoretical gain control system that can improve the dynamic amplitude range of the input and output of the DBES. These components were designed and implemented but not perfected and as a result they were turned off for the experiments. It is included here because this is a novel concept that will be implemented in the future.

Radios are capable of receiving and transmitting over wide ranges of power. Many radios will vary their transmit strength by 30 dB or more depending on scenario conditions. The Radios receive side is even more dynamic as many can receive a 90 dB range of signals or greater. This large range is difficult for any DWCE to reproduce. The problem is that without gain control, the dynamic amplitude range of the system is defined by the number of bits in the ADC and DAC. For example, if there are 16-bits on the DAC then the maximum amplitude that can be produced is represented by 0x8FFF or a logical zero in the first bit location (indicating a positive value) with the remaining bits all logical one's. The smallest analog value that can be represented is 0x0001 or all zeros except for the least significant bit. This means that the largest value is 2^15 times greater than the smallest value. This is equivalent to saying the output has a range of 90.3 dB. This sounds good; however, several bits are required to create a modulated signal.

The number of bits required is waveform dependant, but you can assume that if a waveform needs X dB of Signal to Noise Ratio (SNR) to operate, then it would need that equivalent number of bits in the DBES for the DBES to emulate that signal without incident. If a waveform needs 30 dB of SNR, the DBES needs at least 5 bits to represent that signal ($30 \lessapprox 20 \log 2^5$). This means that 30 dB is now lost from the dynamic range of the DBES since bit errors will occur in the radio if the DBES cannot digitally represent the signal. The only way to gain that range back is to use gain control. If there is an attenuator on the transmit side of the DBES then it could attenuate the signal to the appropriate power level and then use more bits to represent the signal. This can conceptually be thought of as analog floating point, where the attenuation sets the gain (or exponent) for the DAC signal (the mantissa).

A similar setup can be used on receive side that is called the Receive Gain Control (RGC). The RGC is there to improve the dynamic range on the receive side of the DBES. If a radio transmits a very large signal and then a very small signal, the RGC can compensate for this. The RGC is set up just like a traditional Automatic Gain Control (AGC) for radios, only the algorithm is slightly different to accommodate the fact that the signal is not going over the air and uses knowledge of general waveform behavior instead of specific waveform behavior.

Both the transmit and the receive sections have an gain control component that adjusts the attenuators on the AFE in an effort to increase the amplitude dynamic range of the system, although they were bypassed for the final experiments. In a perfect world both attenuators could be utilized to change the amplitude of the signal to the right level without any loss of information; however, the propagation time of the signal and the switching time of the attenuators eliminate that notion. These factors guarantee that whenever the attenuator is changed, an error in the signal is sure to follow. Because of this, both receive and transmit algorithms are written to change as infrequently as possible.

3.3 Hardware Design

This section covers material that was published at the FPL conference in 2012 [7]. The physical implementation of the system is a custom computing machine using FPGA nodes similar to the *Spirit* cluster at the University of North Carolina at Charlotte [28]. The two main differences are the FPGA technology (*Spirit* is based on the older Virtex 4) and the presence of an interface for the analog front end. Initial investigations revealed that the Virtex 4's networking and computing capabilities were insufficient to meet the high requirements targeted for the DWCE; however the Virtex 6 SX was sufficient. The critical differences are described below. After the reason for expanding the cluster is defined, critical design decisions for the new cluster will be detailed and this solution laid out. In addition to the FPGA Cluster, there are two more boards that had to be fabricated before the cluster could be used as a DWCE: the analog front end board and the transmit receive and calibration board known as the TRCal board. The AFE has the ADC/DAC and RF up and down converters to translate a radio signal into a digital signal and vice versa. The TRCal board is used to interface between the AFE and the radio.

The Virtex 4 FX chips on the *Spirit* cluster have limited resources as compared to Virtex 6 chips. After the initial Signal Path Core was developed; logic, DSP cores, and memory were all found to be limiting factors for different aspects of the application. The *Spirit* FPGA, the V4 FX60 has about 56 thousand Logic Elements, 232 DSP cores, and 4,176 Kbits of memory. At the time of investigation, one signal path core used 42 DSP cores. This would only allow for five signal path cores per FPGA which was not sufficient for the application. The new FPGA designated for the cluster is the V6 475SX and has 476 thousand logic elements, 2016 DSP cores, and 38,304 kbits of memory, which is generally a 10x improvement in all categories and is sufficient for the DWCE application.

In the last two generations of FPGAs Xilinx has focused on increasing the networking capabilities of the chips by increasing the number of their Multi-Gigabit Transceivers (MGT). These MGTs have been given multiple names and the ones on the V6 chips are called GTX transceivers. From the V4 MGTs to the V6 GTXs the speed improvement is moderate going from a measured 4.4 Gbps on *Spirit* [29] to a theoretical 5 Gbps a V6 -1 speed grade. However, the SPIRIT cluster could only utilize 8 MGTs and the new cluster will utilize 32 GTXs for networking. These changes will result in a 4.5$\times$ improvement in networking bandwidth between the two systems. This is critical because in order to meet an application requirement of 30 MHz signal bandwidth for 32 radios, it is necessary for each FPGA to receive 31 different signals at 65 MHz sample rate where each sample is 44 bits. This results in 89 Gbps of data, excluding overhead, which is much more than the theoretical single-direction maximum of the *Spirit* FPGA node at 35.2 Gbps.

The decision for which FPGA to utilize was a difficult one to make. Initially, the V6 240LX chip was targeted because of the ideal price-to-performance range. Unfortunately, after the network was designed we found that in order to meet networking requirements additional transceivers were needed. The network bandwidth was a critical factor to meet the signal bandwidth requirements. Additionally, bundling the transceivers into groups of four was necessary to reduce the number of physical connections to and from the board. This also agrees with the FPGA hardware design as groups of four utilize a single clock on the FPGA. Breaking that grouping results in additional complexity and reduced performance. Using the V6 240LX chip and it's 24 GTXs we could only achieve six connections for each FPGA. A (32,6,2)-network solution would require 38 compute nodes which is 17 more FPGA boards than a (32,8,2)-network (see Section 3.1.5 on (V,k,t)-networks). In other words, 17 less FPGA nodes are needed if eight connections are used versus six connections. This led us to decide on a larger chip with more network capability. The next largest size V6 LX FPGA with more transceivers is the 550LX. It's 36 GTX connections allowed for eight physical network connections and an additional four transceivers for other connections such as SATA SSDs. The eight physical connections were also necessary to create a four dimensional torus which is an ideal connection for many applications. In the end, the 475SX chip was chosen because it had 36 transceivers and the additional DSP elements and memory improved application performance.

Once the FPGA was chosen it was necessary to determine what will house the FPGA. An exhaustive online search revealed that there was nothing that could meet the needs of the system that was commercially available. The principle problem was that they did not provide the connections that were required for the system. It was critical to have at least one connection for the AFE, one connection for a SATA SSD, and 8 connections that utilized 4 transceivers each. These specific requirements could not be met by COTS board. It was determined that a custom board would be needed. Unfortunately, this became a common trend for all the hardware in the system. Principle to what board we would use was how it was to be stored, cooled, and powered. Many different chassis were looked at including AMC, ATCA, and MicroTCA. At first these were not desirable because the backplane on the chassis was over-engineered for networking between the payload slots as compared to our needs. Additionally, we found it difficult to physically fit all the connections necessary on many sized boards. We thought to make a custom solution but realized that determining power constraints and cooling was a major challenge and leveraging the current standards would be ideal. We first looked to the AMC standard but it was very large. While it could meet all the connection requirements the density of the cards in the rack was low necessitating extra racks for the system. In the end we settled with a double-width full-height MicroTCA 0.4 chassis. This utilizes an ideal amount of space and with the utilization of the RTM board can house all the connections necessary.

After the chassis was determined, the next focus was on the AMC FPGA and RTM

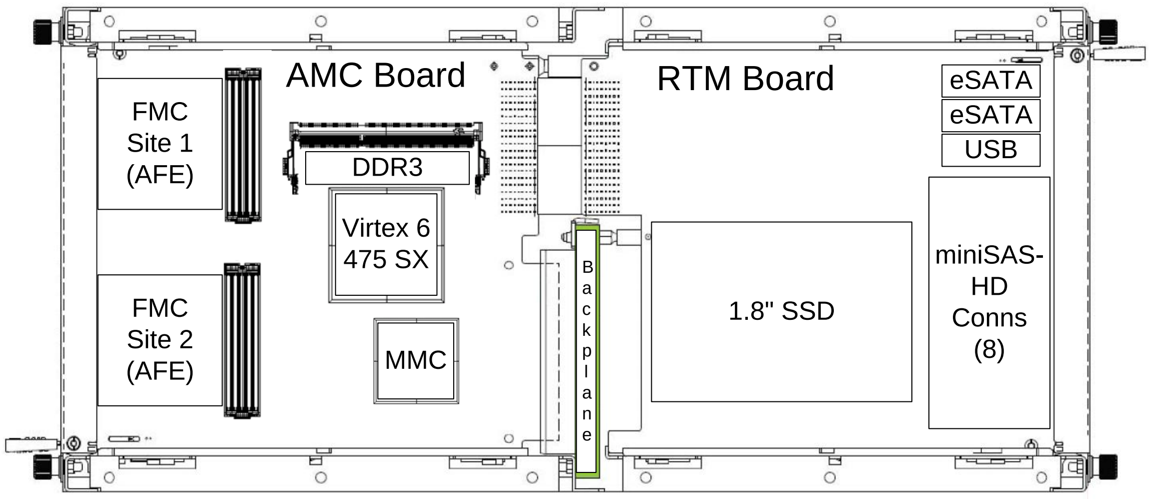
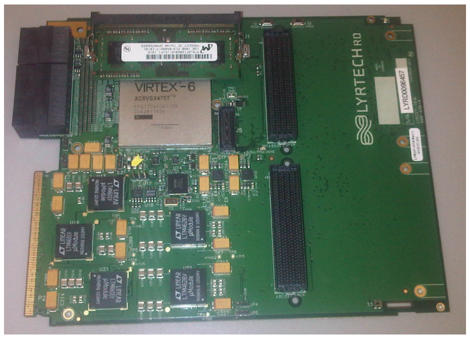


Figure 3.10: μTCA implementation showing AMC, RTM, and FMC boards utilized in the chassis

boards that would support our needs. Because of the 32 GTX network connections and the use of a SATA SSD, no commercial boards were found. Fortunately, there was a board that utilized 90% of the functionality that was needed. This board was leveraged and expanded to get all the connections we required. The Rear Transition Module (RTM) that is used with this specification added sufficient front and rear panel real estate to allow for the necessary connections as shown in Figure 3.10. The AMC board houses the FPGA, memory, and two FMC cards and the RTM contains the SATA drive and all the network connections. The backplane provides power, gigabit Ethernet, clocking, and board control.

The analog front end is another critical component. This component consisted of an ADC, DAC, and IF-RF up and down converters. Initially it was thought that we could find COTS components, utilizing an ADC-DAC FMC card and exterior IF-RF converters. The main issue is that traditional IF solutions are too expensive, so alternative solutions were looked into. Finally, we came across a non-traditional all-in-one design that incorporated all these components onto a single FMC card. This COTS card unfortunately did not meet all the specifications we needed and as


(a)

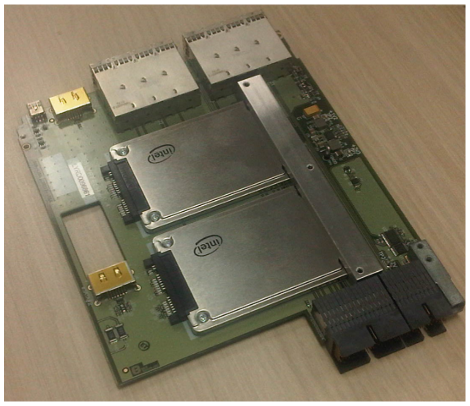

(b)

Figure 3.11: μTCA boards rev. A (a) AMC FPGA board (b) RTM board

such the company was contracted to modify their design to meet all requirements. The result was a cost effective solution tailored to our needs.

The RTM board is used for additional connectivity and physical space to mount necessary components. The principle need for the RTM was for the eight mini-SAS HD cages that are used for the high-speed network. Various form factors were considered including QSFP+; however, mini-SAS HD had the smallest footprint which made it ideal for the application. There were plans to utilize the QSFP+ connectors, however it would have been necessary to use full-height module slots in order to fit eight connections whereas the mini-SAS could fit eight in a mid-height module. The RTM includes mounts for two 2.5" Solid State Drives (SSDs) that will be used for parameter storage and delivery and two eSATA connections that can either be used for additional storage or for an auxiliary ring network. There are only three GTX transcievers for these four connections, so either two SSDs and one eSATA can be used, or one SSD and two eSATA. A soldering change is necessary to convert between the two, which was necessary to support the high-speed lines. Finally, the RTM supports a USB connection that will act as a JTAG port for the FPGA, or as a UART.

The TRCal board is the final board built for the system and is shown in Figure 3.12

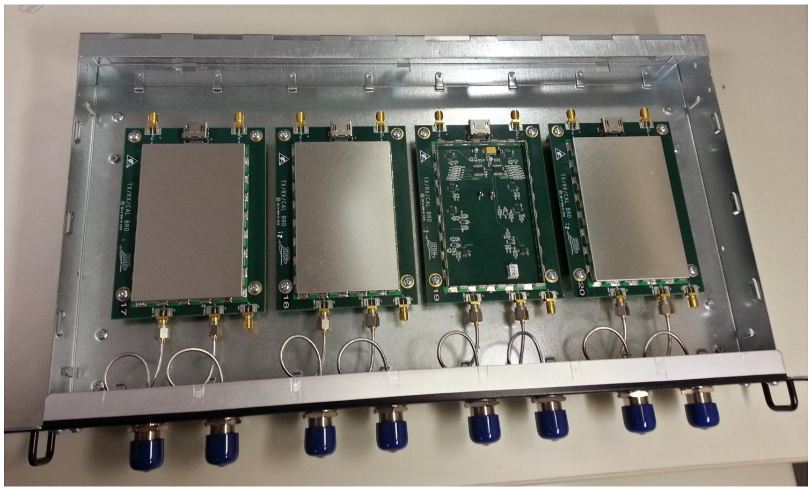
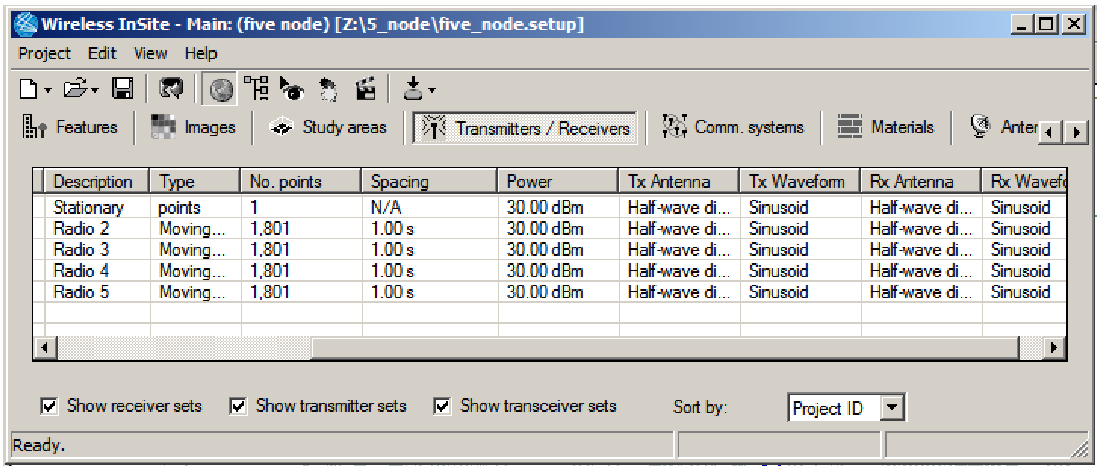


Figure 3.12: TRCal boards and chassis: the RF shield of one board was removed to view the components underneath.

and stands for Transmit, Receive, and Calibration board. This board interfaces between the analog front end and the radio and has switching, splitter/combiner, and attenuation chips to route different types of radios to the analog front end and ensure the power going into the AFE is appropriate. It allows for either an antenna connection that has the transmitted and received signals on the same cable, or a full duplex connection that has transmit and receive signals separated on two different cables. It also has a calibration capabilities with a second input port to allow a known signal to be sent to the receive side and the ability to loop the transmitted signal back onto the receive port. These could allow for automatic calibration of the system without user intervention but this automation process has not yet been implemented. There is a microHDMI connection on the front of each FMC422 board that contains a few digital lines that can be set with by the FPGA on the 6113. A simple component was written in VHDL to interface with the chips on the TRCal board to program the switches and attenuators. Parameters for the component can be set at startup through the solid state drive, or at runtime with the DBES GUI over ethernet.



Description	Type	No. points	Spacing	Power	Tx Antenna	Tx Waveform	Rx Antenna	Rx Wavefo
Stationary	points	1	N/A	30.00 dBm	Half-wave di...	Sinusoid	Half-wave di...	Sinusoid
Radio 2	Moving...	1,801	1.00 s	30.00 dBm	Half-wave di...	Sinusoid	Half-wave di...	Sinusoid
Radio 3	Moving...	1,801	1.00 s	30.00 dBm	Half-wave di...	Sinusoid	Half-wave di...	Sinusoid
Radio 4	Moving...	1,801	1.00 s	30.00 dBm	Half-wave di...	Sinusoid	Half-wave di...	Sinusoid
Radio 5	Moving...	1,801	1.00 s	30.00 dBm	Half-wave di...	Sinusoid	Half-wave di...	Sinusoid

☑ Show receiver sets ☑ Show transmitter sets ☑ Show transceiver sets Sort by: Project ID

Ready.

Figure 3.13: Wireless InSite parameter input

3.4 Scenario Compiler: REMCOM Wireless InSite

This section will describe the REMCOM Wireless InSite software used to generate the parameters, and will also describe the propagation models used. The REMCOM software has some of the highest fidelity RF modeling tools available on the market and they have also done a lot of work to speed up the runtime of their simulations. These two factors lead it to be an ideal software for the DBES.

3.4.1 Wireless InSite GUI

The GUI for Wireless InSite enables the creation and manipulation of different scenarios. REMCOM performed a significant amount of work to achieve the capability presented in this section and much of the theory was published at the ACES conference in 2013 [16]. Figure 3.13 shows the front end software that allows you to change each individual radios characteristics. Figure 3.14 shows the three dimensional view of the scenario and allows you to interact with the different radios routes and edit terrain and buildings. From the point of view of the DBES system the goal for this software is to accurately model the scenario. For a more detailed description on the software please see the REMCOM website (www.remcom.com/wireless-insite).

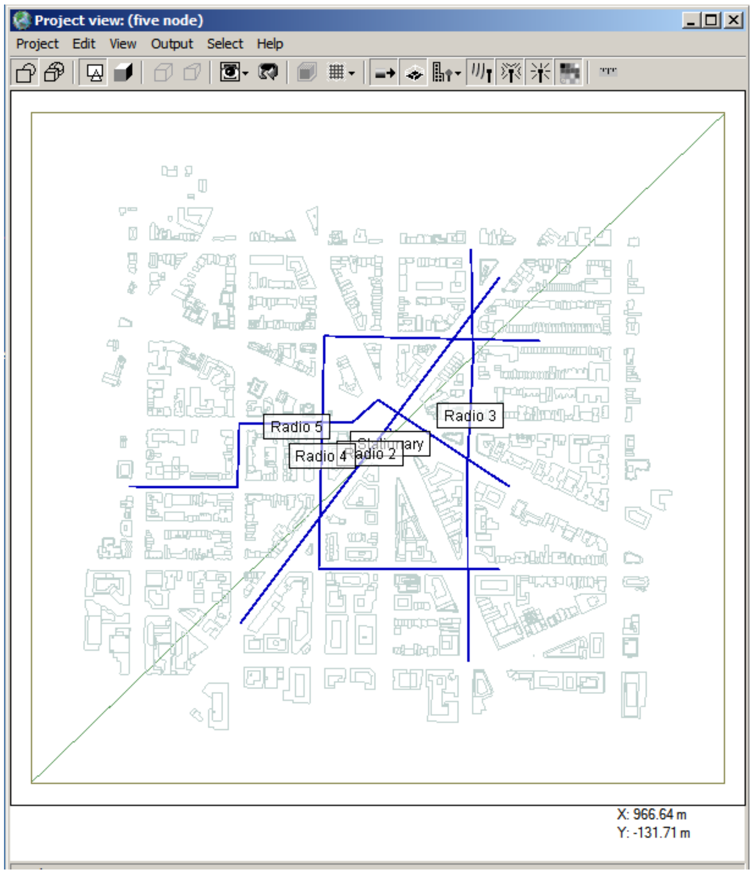

Project view: (five node)
Project Edit View Output Select Help
Radio 5
Radio 3
Radio 4
Radio 2
Stationary
X: 966.64 m
Y: -131.71 m

Figure 3.14: Wireless InSite 3D scenario input

3.4.2 RF Propagation Model

After a scenario has been defined it is necessary to generate the parameters that define the RF environment of that scenario which can be then given to the DBES. There exist many different models that have differing theoretical approaches to modeling the real world in this manner and they can generally be broken into three categories: statistical models, Finite Difference Time Domain (FDTD) models and ray tracing models. Of these the ray tracing models are the best to use with the DBES. The FDTD models are not the best to be used with the DBES because current models do not generally break down each individual wave front that passes over a receiver. This means that while the amplitude results could be used as an input to the DBES, the individual propagation paths are inherently summed at each point in the model which makes it difficult to pull that information back out to give to the DBES. Many different statistical models exist but the most common are Rician and Rayleigh fading models. They define a statistical distribution of received power strength that can be seen in certain scenarios. In general, these models do not break out individual propagation paths and instead attempt to define how the cumulative received power will vary over time. If this variation over time is long enough to be appropriately sampled by the parameter update rate of the DBES then these models are easy to implement by simply changing the attenuation of a given signal path over time. The advantage of statistical models is that they can be easily set up and used; however, they are not accurate to a specific real world scenario unless that scenario falls under very specific conditions. Additionally, the signal delay and Doppler is not directly included in most statistical models which further limits their usefulness.

The DBES was designed for use with ray tracing models. These models break down the RF environment to many different propagation paths between transmitter and receiver. The different propagation paths are caused by the reflection and/or refraction of the RF wave as it travels through the environment. This is the reason that Wireless InSite was chosen. There are many different algorithms used of computing these path and the REMCOM software used by the DBES selects from one of three: full 3D tracing model, vertical plane model, X3D model. The full 3D model is the most computationally expensive model and traces all the rays from each transmitter to each receiver at every time step. REMCOM uses some approximation techniques, but the purpose is to have little impact to the fidelity of the model. The vertical plane model is good for open spaces with hilly terrain. It is a two dimensional ray tracing algorithm which only investigates one plane orthogonal to the surface of the Earth along the line connecting the transmitter and receiver. The rays traced largely focus on ground and atmospheric effects: refractions over the tops of hills and reflections off the ground, mountains, and the upper atmosphere where applicable. This is a less computationally expensive model good for rural terrain over larger distances but can be very inaccurate in the presence of buildings. The final model is another full 3D model that makes large assumptions about an urban environment. It typically attempts to find the four different propagation paths where applicable: line-of-sight and to the left, to the right, and over top of an obstruction. This makes for a significantly lower computational time and generally finds the three to five largest propagation paths. This model is a good middle ground sacrificing a little fidelity for large gains in computation time.

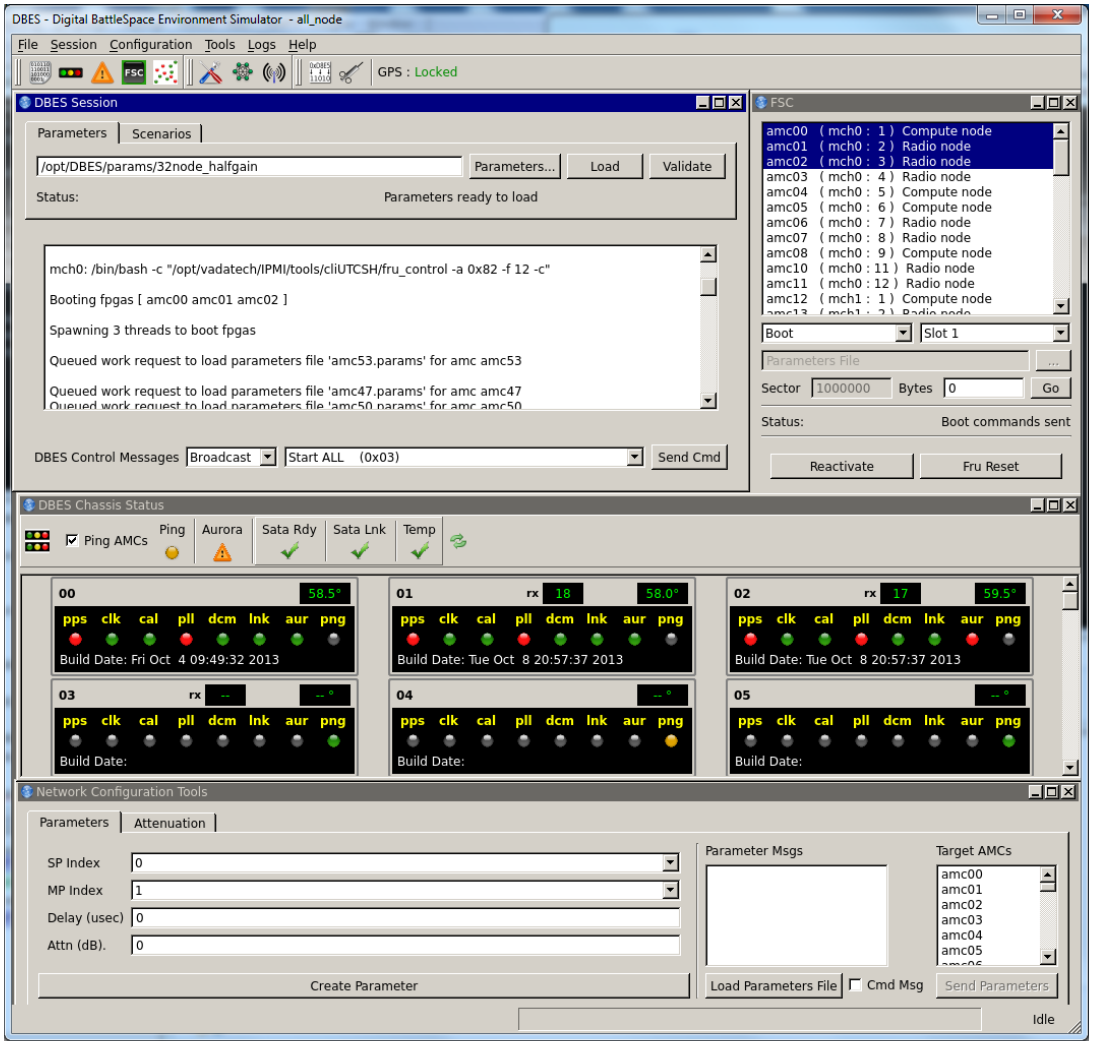
All the ray tracing models produce M different propagation paths for each transmitter and receiver pair at every time step. Each propagation path can be defined by the path loss (A), signal propagation time (d), and the phase shift incurred by reflections (ϕ). These are the parameters sent to the DBES that define the RF environment: one set of A, d, and ϕ for each M propagation path, for each signal path, for each time step. The ϕ parameter has not yet been implemented at the time of writing this dissertation and is not included in any of the evaluation experiments. There is a straightforward method for including this parameter into the signal path core that only uses a single additional adder per parallel path. Unfortunately straightforward does not mean that it can be quickly included because changes would have to be made to many components to deliver the parameter to the signal path core and the timing of the signals going through the core would have to be altered to take into account the delay of the addition.

3.5 DBES Control Graphical User Interface

This section describes the DBES Control Graphical User Interface (GUI) used to control the DBES. This tool acts as the control system for the DBES provides a method for a user to quickly access DBES functionality. This GUI is actually a critical part of the system because without it, getting the system up and running would take too much time to be feasible. It proved invaluable to the development of the VHDL by reducing the amount of time the developers needed to get a design onto the FPGAs and debug the added features. It is really a developers tool in it's current state but the plan for the GUI is to allow anyone with a little training to be able to use the DBES. The author had little to do with the actual programming of this software and it's underlying functionality, but did provide guidance in it's development.

The DBES Control GUI functionality includes: parameter translation from the scenario compiler to the DBES, configuration setup of the DBES, loading parameters and configuration to the SSDs on each FPGA, starting and stoping the signals and the emulation, and 6113 board control including reseting, loading images to, and booting the FPGA. Each of these tasks requires a significant amount of automation to perform on a large scale. For example, booting one FPGA over JTAG is easy, but doing that 52 times is not feasible. To handle this several different systems had to be implemented. The FPGA must be loaded on power-up with a "Golden Boot" image that has a processor which communicates over Ethernet and has interface to the Flash memory. The GUI can then communicate with the FPGA and write bit files over Ethernet to the FPGA, which then stores those images in Flash memory. At that point the GUI can tell the FPGA to reset itself and boot itself with the bitstream loaded into flash memory. Each of the GUI functionalities listed previously have a complicated set of tasks that must be done, which is why automation is critical when using this many FPGAs.

Figure 3.15 shows the GUI and commonly used windows. It was known that the functionality and display of the GUI would evolve with time and the developer of this GUI had the excellent idea to split all the functionality into different windows so that it would be easy to visualize different combinations of information by bringing up and positioning different windows. This also assisted the development by compartmentalizing the functionality and easily allowing pieces of the GUI to be edited without affecting the entire program. This figure shows the DBES Session window where different parameter sets can be selected, the FPGA Session Control (FSC) window which is the interface to the golden boot FPGA image, the DBES Chassis Status that shows each FPGAs status, and the Network configuration that allows the user to send parameter messages to specific FPGAs. There are also windows for setting up the DBES configuration, viewing REMCOM generated scenarios, and compiling Matlab generated parameters.



GPS : Locked

/opt/DBES/params/32node_halfgain

Status: Parameters ready to load

```
mch0: /bin/bash -c "/opt/vadatech/IPMI/tools/cliUTCSH/fru_control -a 0x82 -f 12 -c"

Booting fpgas [ amc00 amc01 amc02 ]

Spawning 3 threads to boot fpgas

Queued work request to load parameters file 'amc53.params' for amc amc53

Queued work request to load parameters file 'amc47.params' for amc amc47
Queued work request to load parameters file 'amc50.params' for amc amc50
```

DBES Control Messages: Broadcast — Start ALL (0x03)

```
amc00  ( mch0 :  1 )  Compute node
amc01  ( mch0 :  2 )  Radio node
amc02  ( mch0 :  3 )  Radio node
amc03  ( mch0 :  4 )  Radio node
amc04  ( mch0 :  5 )  Compute node
amc05  ( mch0 :  6 )  Compute node
amc06  ( mch0 :  7 )  Radio node
amc07  ( mch0 :  8 )  Radio node
amc08  ( mch0 :  9 )  Compute node
amc10  ( mch0 : 11 )  Radio node
amc11  ( mch0 : 12 )  Radio node
amc12  ( mch1 :  1 )  Compute node
amc13  ( mch1 :  2 )  Radio node
```

Boot — Slot 1

Parameters File

Sector 1000000 Bytes 0

Status: Boot commands sent

Ping AMCs ✓

```
00                    58.5°     01        rx 18    58.0°     02        rx 17    59.5°
pps clk cal pll dcm lnk aur png  pps clk cal pll dcm lnk aur png  pps clk cal pll dcm lnk aur png
Build Date: Fri Oct  4 09:49:32 2013  Build Date: Tue Oct  8 20:57:37 2013  Build Date: Tue Oct  8 20:57:37 2013

03        rx --    -- °     04                 -- °     05                 -- °
pps clk cal pll dcm lnk aur png  pps clk cal pll dcm lnk aur png  pps clk cal pll dcm lnk aur png
Build Date:                       Build Date:                       Build Date:
```

SP Index 0

MP Index 1

Delay (usec) 0

Attn (dB). 0

Target AMCs: amc00, amc01, amc02, amc03, amc04, amc05, amc06

Idle

Figure 3.15: DBES control GUI showing commonly used windows

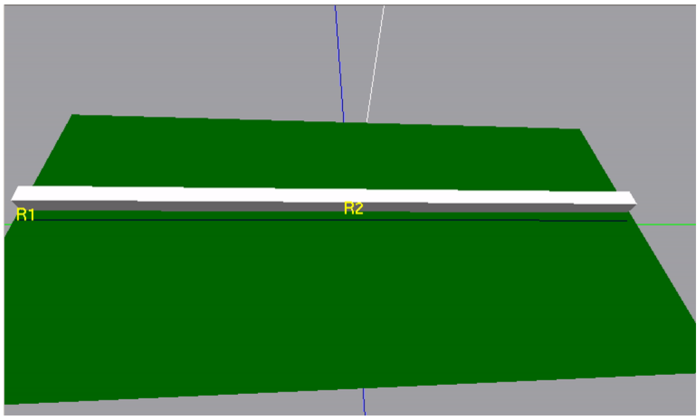


Figure 3.16: Two node run-by scenario described in the use case

3.6 DBES Use Case

This section will describe how to operate the DBES in it's current state at the time of writing this dissertation. The section will detail how to set up a scenario, use an RF Propagation model to generate parameters, input those parameters to the DBES, and run the scenario with actual wireless devices. There are plans to automate much of this process to make the DBES easier to use, but that is future work that will not be included in this dissertation.

3.6.1 Generating Parameters

As mentioned there are many ways to generate parameters and this use case will focus on using Matlab to generate a simple scenario; however, it will also describe how parameters generated from Wireless In-site can be compiled and loaded onto the DBES. The scenario chosen has a stationary radio and another radio moving very quickly in a straight line past that radio and is shown in Figure 3.16. This figure was generated from the Wireless InSite and DBES GUI software and the scenario was first run with the Wireless InSite software. This software found that the reflections created by the wall were not large enough to be used and as such only the line-of-sight path is taken into account.

This single path was described by placing the radios on a cartesian grid where one is at the origin and the other moves on a straight line. The line was placed such that it came to 100 meters from the first radio. Time is established from zero to five minutes with millisecond steps. The distance between the two radios can now be calculated according to the Pythagorean Theorem. From the distance, both the delay and attenuation values can be found. The attenuation was found using a free-space path loss model (Equation 3.10) and the delay is found by simply dividing the distance by the speed of light. This is just one example and it could be made more complicated or changed to suit different needs; however, even this small model was taxing on the memory of the computer I was using. Complicated scenarios would be very difficult to implement in Matlab, which is why dedicated software is used in these cases.

$$FSPL = \left(\frac{4\pi df}{c}\right)^2 \tag{3.10}$$

After the parameters are generated they must be transformed into a form that can be stored on the solid state drive and read in by the DBES. In this use case study, this functionality is done by a Matlab script, but if the REMCOM software is used it is done with the DBES GUI. The Matlab script and the DBES GUI are functionally equivalent in this regard, with the exception that the GUI has the capability to compile data more quickly because it can use multiple CPU cores at the same time and C++ is inherently faster than Matlab. A file is generated for each FPGA that contains a sequence of 64-bit parameters and commands. The FPGA will ultimately read this parameter file from start to finish in order, so the location in the file determines the time the parameter will be read. Each 64-bit parameter contains an SPI, MPI, shift, mantissa, and delay data. The Signal Path Index (SPI) specifies a specific signal path component on the FPGA and is a 6-bit integer value. The Multi-Path Index (MPI) specifies the propagation path of the given signal path to load the parameters onto and is a 4-bit integer value. The attenuation parameter is split into the shift and mantissa values because it was more computationally effective to do this way than have single large multiplier. This is done by equations 3.11, 3.12 and 3.13. The mantissa is in 1.15 fixed point notation and the shift is a 5-bit integer. The X.Y fixed point notation indicates that there are X bits to the right of the decimal and Y bits to the left of the decimal which means the total size of the parameter is X + Y bits. The delay parameter is in 13.19 fixed point notation and has units of sample periods. Equation 3.14 shows how to obtain this number from a delay value in seconds. At the end of each parameter period, a specific 64-bit command sequence is used to tell the DBES to wait until the next parameter period before continuing to read the parameter data. To recap, compiling parameters results in a binary file for each FPGA in the compute cluster. Each binary file has all the parameters for each signal path on that FPGA ordered in time and partitioned by a command parameter.

$$attn_{dec} = 10^{\frac{-attn_{dB}}{20}} \tag{3.11}$$

$$\alpha_{shift} = -\lceil log2\left(attn_{dec}\right)\rceil \tag{3.12}$$

$$\alpha_{man} = \lfloor attn_{dec} * 2^{\alpha_{shift}} * \left(2^{16} - 1\right)\rfloor \tag{3.13}$$

$$d_{param} = \lfloor \frac{d_{sec}}{f_s} * 2^{19} \rfloor \tag{3.14}$$

3.6.2 Determining Configuration

The parameters are not the only piece of information that is loaded onto the solid state drive before a test is run. The configuration data is also loaded. As described toward the end of Section 3.1.5, each FPGA must be configured with a different routing configuration. In addition, it may be desirable to use a subset of nodes which would require a different set of configuration data. To achieve this capability, a second binary file, different from the parameter binary file, is generated that contains the configuration information. This section describes this process.

The number of wireless devices used in the scenario determines how many DBES nodes should be utilized. It is possible to use the entire thirty-two node DBES configuration and only connect two radios but that is a significant waste of resources and power. In addition, while the system was still in development, it was not possible to bring up a full 32 node DBES because of reliability issues. If one of the 52 FPGAs goes down, even if it isn't a node that is directly being utilized by a current scenario, the signal on every node will stop being transmitted. To limit these issues and use less hardware, different configurations can be loaded into the DBES. A Matlab script, named the configuration generation program, was written to determine the parameters that need to be sent to each FPGA for a given set of nodes used. It simply goes through each signal path component in use and determines how the various crossbar switches and sync-and-sum components need to be configured to deliver a signal to and from that signal path. This is iterated for each signal path in use and then the commands are correlated and a file is generated for each FPGA which contains all the configuration data that must be sent to each of the routing components on that FPGA. This Matlab script will eventually be compiled into a dynamically linked library that the DBES GUI will call to automatically determine system configuration based on user input. Also included in the configuration data is a parameter that tells the signal path cores which RF frequency is being used, and the configuration for the TRCal board.

The output of this configuration generation program is a binary configuration file for each FPGA that is used in this configuration. The configuration is split into 64-bit data just as the parameter file is. The order of data in the configuration file does not matter; however, when the data is loaded onto the solid state drive, the configuration data must be loaded into the start address with the parameter data starting after that address.

Lastly, there is a command inserted at the beginning of the parameter data which prevents the FPGA from reading any more from the solid state drive until a command is sent over ethernet. This command is referred to as the "Scenario Start" command and allows the FPGA to boot up into a configured state without loading any parameters into the signal path component. This allows a user to configure the signal paths over ethernet and initialize the system without running a scenario and was very useful in the development of the system.

3.6.3 Initializing the DBES

There are several steps that must be taken to initialize the DBES prior to running the system. At power-on the DBES is automatically loaded with a golden boot image that has a microblaze processor on it which can use the ethernet, solid state drive, and flash memory. This golden boot image and the software used to interface with it was developed by the combined effort of many people in the UNCC RCS Lab. It's capability to control the FPGA cluster is unique and powerful tool for any other project that would use this FPGA cluster. Once the configuration data and the parameter data files have been created they can be loaded onto the sold state drive of each FPGA over ethernet using the golden boot image and the DBES GUI software. The DBES GUI allows a user to select the configuration and parameters they want to use and then has the ability to load those data files onto the correct SSD. Next, the DBES GUI can load the DBES RN or CN image into flash memory on each FPGA node, if that has not already happened. The golden boot images can then reboot themselves into the RN or CN image loaded in flash. Each FPGA will configure themselves, initialize the analog front end, and load the configuration parameters from the solid state drive.

3.6.4 Connecting Radios to the DBES

After generating the parameters, determining the system configuration, and initializing the DBES, the DBES system is powered up and configured but is not yet started. At this time, the radios should be connected to the system to ensure they can communicate prior to starting the scenario. A few more steps must be done to achieve this. First, each analog front end must be tuned to the correct frequency and calibrated at this point using the DBES GUI. Next, the user needs to know what power levels the radios will be outputting. The power input to the front of the TR-Cal board cannot be greater than $20dBm$. This is for two reasons: first, the TRCal board's components can break at higher power levels, and second, this ensures there is enough isolation between the channels of the DBES so that they will not be able to communicate through crosstalk. The receive attenuator on the TRCal board must then be set to the correct value to attenuate the signal further for input into the AFE. If the power was too high, the ADC on the 422 would be overrun and wrapped around on itself causing noise that would ruin the signal. If the power level is too low, then some of the integrity of the signal is lost. This will be automated in the future, but for this dissertation, chipscope was used to view power levels of the input of the radios in the system and the TRCal board was adjusted from there.

One final check is to make sure isolation from transmit to receive on the AFE is good enough. There is only $20dB$ of isolation between the two split ports on the RF splitter on the TRCal board. This means that a transmitted signal will feed back through the splitter into the receive of the AFE. If this is not rejected, then a lot of noise will be added to the system. A simple threshold is used to do this and assumes that the received power from the maximum power output from the AFE transmit is less than the minimum power that will be seen from the radio. If this is not the case, then the transmit attenuator on the TRCal board must be increased until this holds true. At this point the threshold can be set and all the radios can be connected.

In addition to setting up the DBES, it may be desirable to run user data traffic through the radios. The radios used in this dissertation used ethernet based IP traffic. A Spirent Test Center CTL-11U-2 was used to generate and analyze IP traffic sent over the radios. The test center was set up to generate unicast traffic which means that all data generated had a single IP address destination. Broadcast, multicast, or any other scheme could be used; but unicast was chosen for it's simplicity. Each of the radios ethernet ports were connected to a port on the test center and the test center handled the generation of the IP packets and analyzed the IP packets that it received. Many different charts could then be created but the one found useful for this dissertation was the number of packets received from each radio over time. Because transmitting over the air is inherently lossy, seeing when packets were successfully received over the air could be a useful tool to analyze behavior.

3.6.5 Running the DBES

The DBES is now ready to run. There is one last check performed before each of the experiments is to make sure all radios can see each other through the DBES. To perform this check the DBES is started but the scenario is not. In this state the signals are moving through the DBES but the parameters are not being streamed from the SSD. A user can now send parameters to be sent over ethernet to instantly change the state of the system. The DBES GUI lets the user select groups of signal paths to set that are specified by selecting a set of radios. To check if all the radios can see each other, all the signal paths can be set to unity gain and each of the radios can be checked. Next, to make sure that the DBES can keep the radios from talking, turn all the signal paths to $100dB$ or more of attenuation. If both these cases are working then the scenario is ready to be run. Any software that is going to be used to monitor the radio is set to record, and the DBES scenario can be started. If the scenario was compiled with REMCOM's software, a scenario viewer was created to show the progress of the scenario. If Matlab was used then unfortunately the best method of knowing the time in the scenario is to start a stopwatch when the scenario is started. In the future, the FPGAs may report back what time they are in a given scenario but this functionality is not yet present. After the hopefully successful scenario is completed then the scenario can be reset and rerun without powering down the entire system if desired.

CHAPTER 4: EVALUATION

This chapter presents the studies performed for evaluation of the DBES. It starts with the two channel prototype that was initially developed to explore the feasibility of the DBES and discusses some of the findings and lessons of that prototype. Next the scalability study is presented that shows how the architecture can be used to scale to very large numbers of nodes. The last section is the fidelity study and it's theoretical, experimental, and observational components.

4.1 Two Channel Prototype

A two channel prototype was developed as a proof of concept and to explore the fidelity and scalability of a full system. This section was written while the feasibility study was underway and was published at the MILCOM conference in 2011 [6]. There exist some approximations about final system sizings and measurements which are now known. A careful reader will notice some discrepancies between this section and the final two sections due to the nature of development over time. The two channel prototype was built with COTS components that utilized similar hardware to a full scale DWCE. The core of the system was built with two ML605 Xilinx development boards, two 4DSP FMC150 ADC/DAC cards, two PXI-5610 National Instruments (NI) upconverters, and two NI PXI-5600 downconverters. The signal path core was coded in VHDL and incorporated with 4DSPs support software delivered with the FMC150 boards that controls the ADC and DAC. Initialization commands and parameters were sent over an Ethernet cable to the ML605 board.

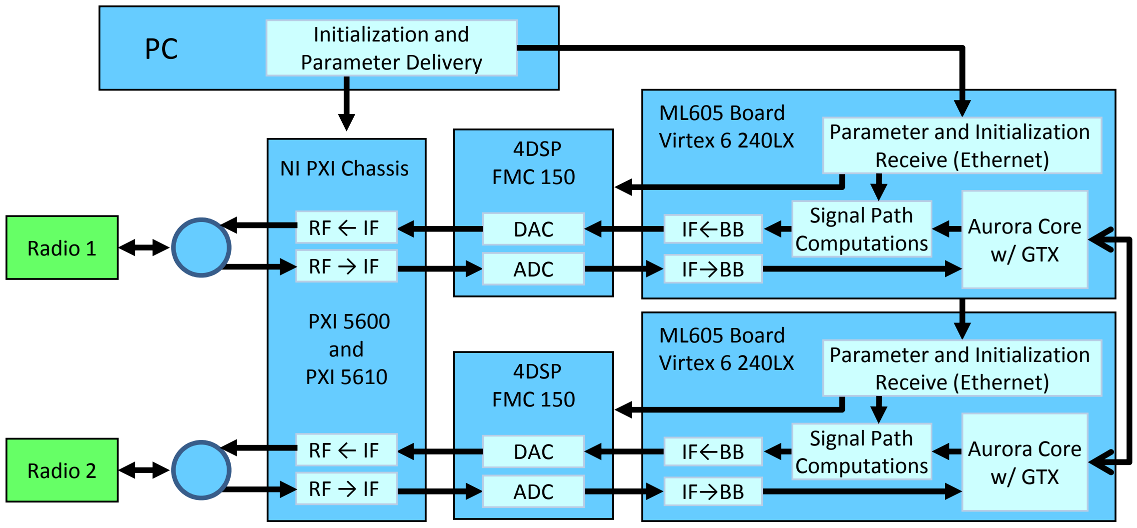


Figure 4.1: Prototype DWCE design

4.1.1 Prototype Validation

In order to examine the fidelity of the implementation three scenarios were examined. First, the attenuation accuracy was tested by utilizing two VHF UHF Line of Sight (VULOS) radios connected through a variable gain attenuator and the prototype. The second setup examined the Doppler and Doppler rate of change capabilities of the system as compared to a SPIRENT SR5500, which is a commercial two-channel DWCE. Third, two pairs of JTRS radios were connected to both the prototype and a SPIRENT SR5500 [30] and the radio performance was compared. Lastly, propagation delay measurements were taken on the prototype to determine if they were in acceptable limits.

Because exact radio performance cannot be discussed in a public setting, the first test is performed without reporting the exact attenuation used. Instead attenuation steps are used. Each step has a corresponding dB value that will not be reported. The same attenuation was entered into a variable gain attenuator and the prototype DWCE. The radios used were JTRS Mobile Tactical Radios (JMTR) running VULOS. These radios were used because they had the capability to utilize the TTC Fireberd

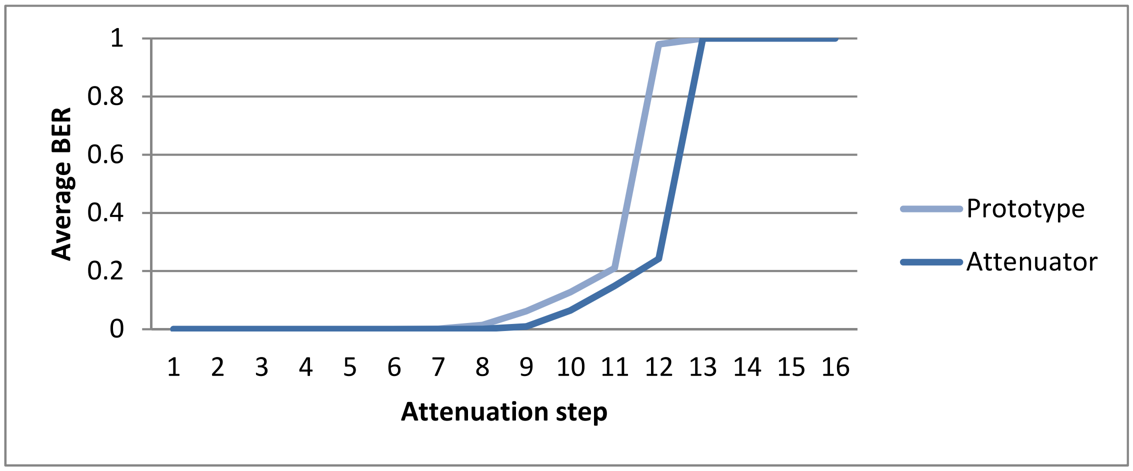


Figure 4.2: Prototype DWCE attenuation validation

6000A Communications Analyzer that could accurately measure the number of bit errors seen by the pair of radios. The bit errors on the VULOS radio were tallied over one minute and a percentage was calculated by the number of Bit Errors divided by the number of Bits transmitted. This is plotted in Figure 4.2.

The second test examined the Doppler capabilities of the wireless channel emulator. A scenario was set up such that the Doppler varied between -100 Hz and 100 Hz in a triangle wave fashion and had a rate of change no greater than $+/-$ 30 Hz per second. The VULOS radios were connected to both the prototype and the SPIRENT SR5500. A visual inspection of the output of both DWCEs on the spectrum analyzer verified the signal was changing as predicted. The VULOS radios produced no bit errors when connected to the prototype. When connected to the SPIRENT SR5500, they experienced 4 to 16 bit errors when the Doppler rate of change went from positive to negative at the peak and trough of the triangle wave.

The final test was conducted with two pairs of radios, the VULOS radios from the first test and the SRW Waveform Development Environment (WDE) platform running the SRW waveform. This platform was used to develop the SRW waveform. In both tests the same parameters were given to both the prototype DWCE and the

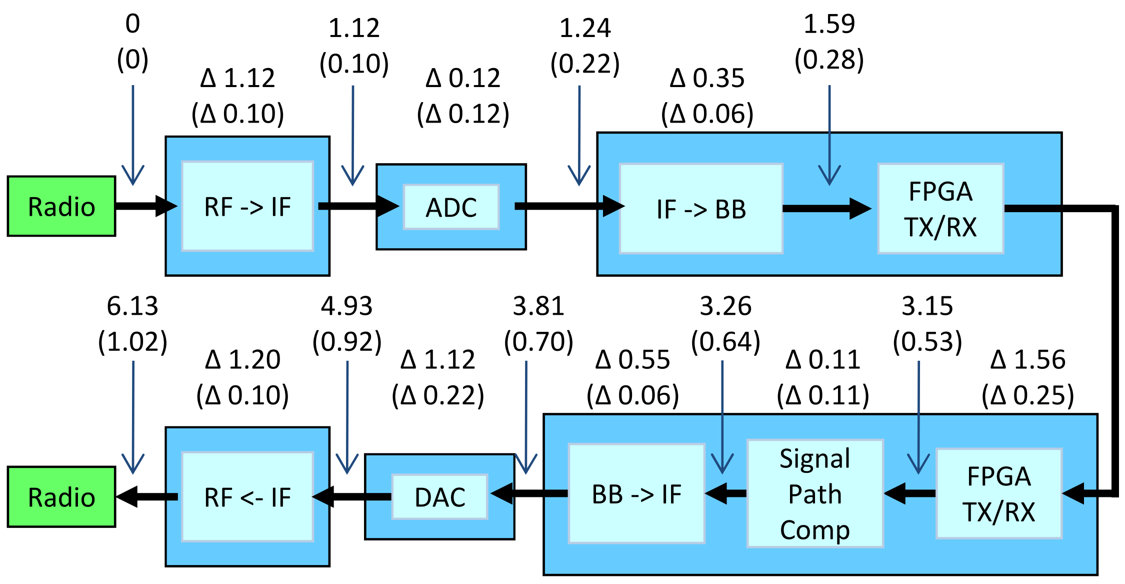


Figure 4.3: Prototype end-to-end propagation delay measurements. All measurements are in microseconds. The time in parentheses is the estimated improved time for a future system.

Spirent SR5500, and the radios performance was examined. Unfortunately, metrics for this test cannot be given as they would reveal radio performance; however the radios behaved identically in every scenario given.

With two FPGAs the total propagation delay for the prototype was approximately 6.13 microseconds, which corresponds to a signal propagation distance of approximately 1.8 kilometers. This is significantly larger than what was anticipated and after the tests were conducted, the PI went back to analyze the system to determine where this could be improved. Figure 4.3 depicts a breakdown of the time through the system as well as predicted times that could be realistically met in future systems.

The delay of almost every component was significantly higher than expected and as a result, each component must be analyzed to determine how to reduce the delay. The GTX tranceivers on the FPGA produced a delay of 1.56 microseconds. This delay was largely due to the Aurora Core utilized as the communication protocol. This core has been removed and a custom streaming protocol was put in its place.

The propagation delay of this new system was measured to be 0.25 microseconds. The DAC had a 1.12 microsecond delay due to the 4 times interpolation done on the DAC chip. The removal of this interpolation will reduce the delay from 290 clock cycles to 59 clock cycles or 0.23 microseconds. The IF/Base Band interface had large overdesigned digital filters that were put in place to maximize filtering performance. Future systems will utilize Quadrature Downconversion in Analog and will bring in in-phase and quadrature signal components as opposed to an IF signal. Additionally, the signal will be oversampled to allow for a future Doppler implementation. This change will allow for a filter with only a few taps reducing the delay below 0.060 microseconds. The RF/IF converters utilized were large boxes with very large filters to ensure out of band rejection was as good as possible. These large filters are what caused the delay. Future large-scale systems will not be able to employ such a large box for cost reasons. The noise will attempt to be controlled in the lab environment and the filtering will be limited. It is impossible to know what this delay will be and it is not something that is measured by manufacturers however, it should be below 0.1 microseconds. If these changes are made the total propagation delay was guessed to be approximately 1.02 microseconds. Section 4.3.2.2.1 shows the propagation delay measured in the final DBES system.

4.1.2 Prototype Scalability Study

The reason for this study was to examine capability to scale the emulator to a large number of nodes. At the time of this study, it was not possible to obtain enough hardware to actually build the system; however, it is possible to extrapolate the number of FPGAs required to build the system given the number of radio nodes desired. This study served as a precursor to the full scalability study presented in Section 4.2. At the time of this study, it helped to estimate that enough resources were available on the planned FPGA cluster, and that there were not enough resources available on the current SPIRIT cluster. The signal path component used for this study was

Table 4.1: Size of various Signal Path Configurations on a Virtex 6 Device

Bandwidth	Number Multipath	parallelism	Slices	DSP48E1	Num Fit on V6 240LX
20 MHz	20	1	928	8	32
40 MHz	10	1	813	8	36
80 MHz	5	1	469	8	63
80 MHz	10	2	950	14	31
80 MHz	15	3	1296	20	23
80 MHz	20	4	1816	26	16

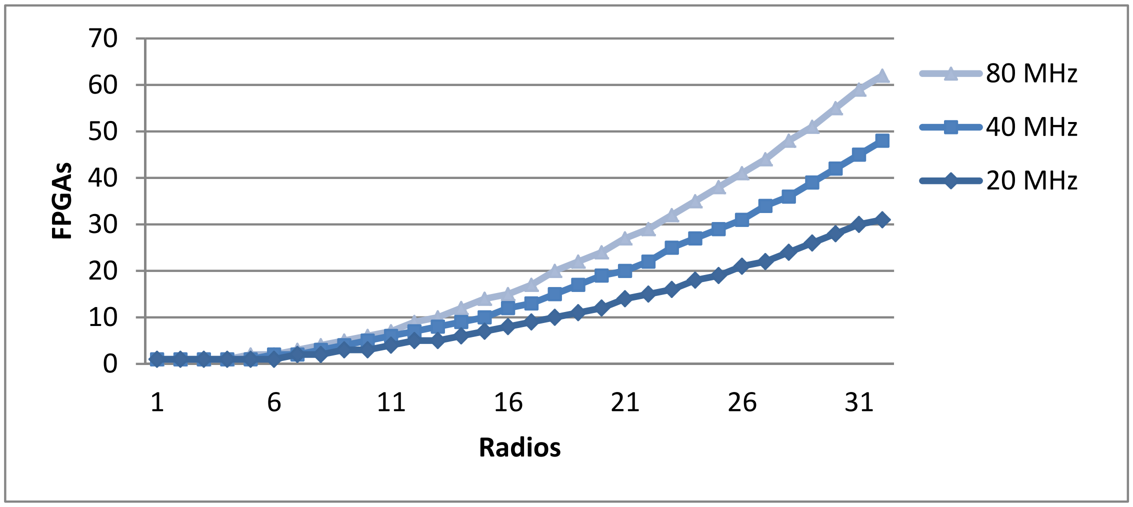


Figure 4.4: Number of FPGA nodes required for a variable number of Radio Nodes and amount of Signal Bandwidth given 20 multipaths emulated.

a the version used in the prototype. This version is different than the final version created for the full scalability study, which accounts for the differences in results. Table 4.1 shows the resources required both for the various overhead components and for different configurations of the signal path computation component. By using these numbers the size of various DWCE configurations was extrapolated and is presented in Figure 4.4 and Figure 4.5. These figures show the Order N^2 relationship between radio nodes and the number of FPGAs required.

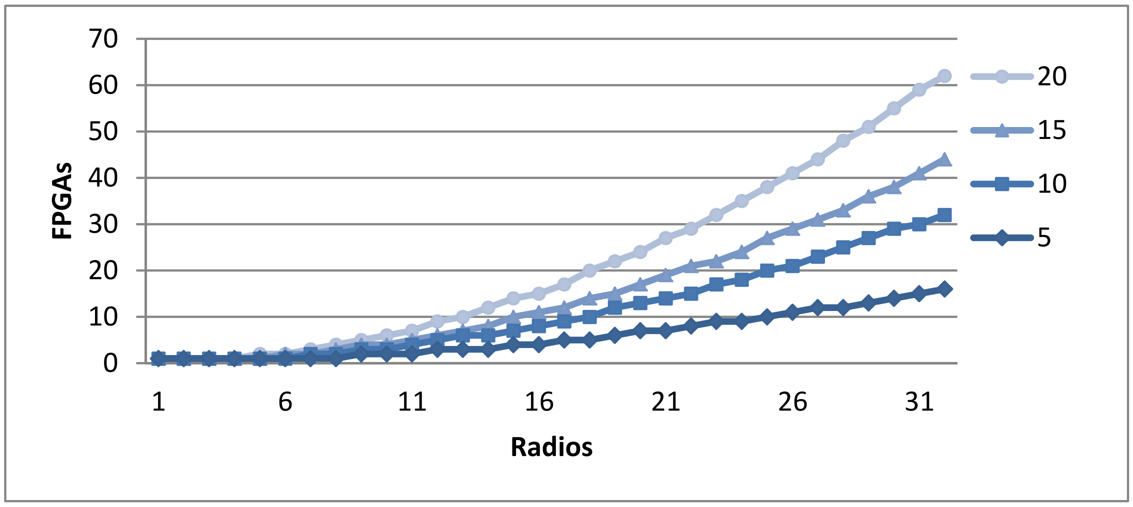


Figure 4.5: Number of FPGA nodes and number of multipaths emulated given a constant signal bandwidth of 80MHz

4.1.3 Prototype Conclusion

The results show how a reasonable sized cluster can be utilized to create a large-scale DWCE. While it is not possible to eliminate the order $O(N^2)$ hardware trend as shown by Figure 4.4 and Figure 4.5 without making optimizations that limit the signal path capabilities, it is possible to adjust the system for a specific desired testing scenario. This can tailor the size of the final system by changing the signal bandwidth and number of multipaths emulated. For example, if a 32 FPGA node DWCE was built, it might only be possible to emulate 16 radios where they utilize a large bandwidth in an urban environment. However, the same size system setting a narrow-band waveform could possibly connect 64 radios to the 32 FPGA system provided sufficient analog hardware.

This prototype was built to demonstrate the capabilities of the system with military radios. While the propagation delay was not ideal, the radios still operated correctly. The discoveries made in this study were critical in the design of the components that made up the final system.

4.2 Scaleability Study

This section will define the scalability of the architecture described in this dissertation and much of this work was published at the FCCM conference in 2013 [5]. The study will assume that the hardware remains unchanged and will show how by simply adding FPGA nodes and analog front ends it is possible to change the size of the system in terms of the number of radios. This can have an impact on the fidelity of the emulation in terms of the number of propagation paths emulated, and the signal bandwidth that is emulated. As stated previously and shown briefly in the MILCOM paper, it is not possible to break the $O(N^2)$ relationship between computations and radio nodes if an all to all connection scheme is maintained; however, the number of computations are also proportional to both the propagation paths and signal bandwidth emulated allowing for a parameter trade space that can be explored. Unfortunately, it is not as simple as taking the number of FPGAs and making sure there are enough resources for the amount of computations required. It is also necessary to factor in the networking aspect both in quantity and structure. This creates interesting trade-offs where propagation delay can be exchanged for signal bandwidth by increasing the number of network hops a signal takes along with the number of FPGAs. The goal of this study is to show this trade-space and detail other networking schemes that can be used for larger numbers of hops.

These tests intend to determine the maximum sample frequency that can be obtained for a variety of system requirements. A total of four different configurations are shown in Figure 4.6 and Figure 4.7 and each is realizable with the hardware presented in the materials section. Two configurations use a $(V, 8, 2)$-network and the other two use a hypercube network. For each configuration, many systems with different amounts of analog front ends and propagation paths are examined to determine the maximum sampling frequency that the system can support. This section will describe the algorithms used to accomplish this. For each system, exact locations

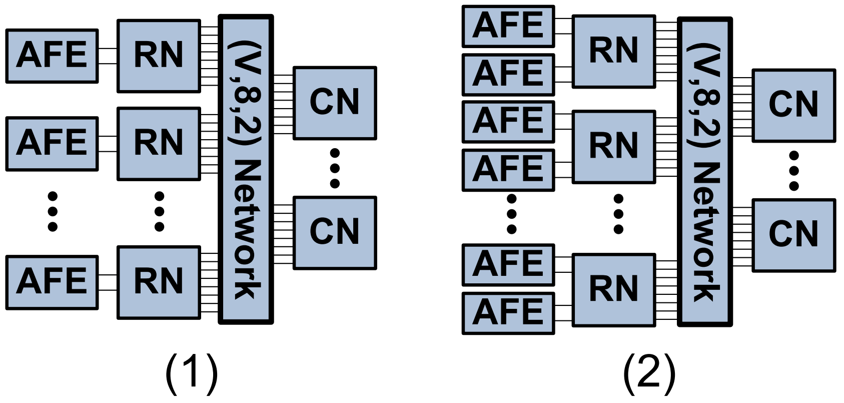


Figure 4.6: System configurations one and two that use a $(V, 8, 2)$-network

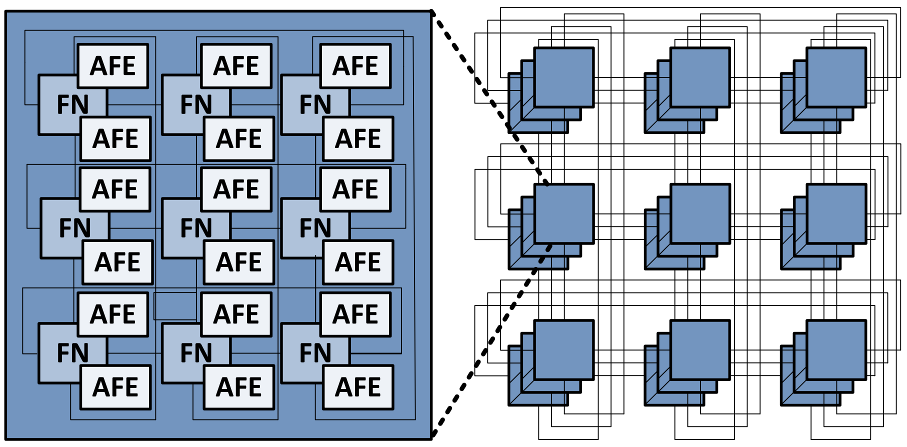
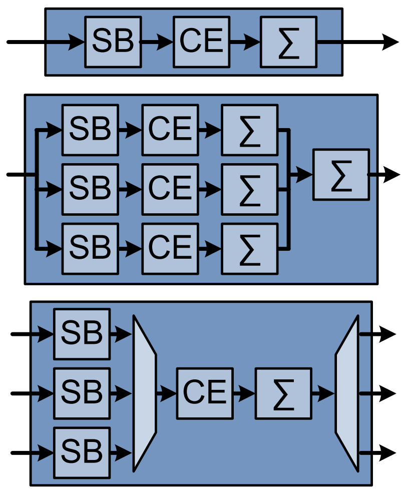


Figure 4.7: System configurations three and four that use a four dimensional hypercube network. A 3-ary 4-cube configuration is shown. The third configuration only has one AFE on each FPGA Node and the fourth has both as shown.



A) $Fsys \geq PP \times Fs$

B) $\text{ParStages} \geq \left\lceil \frac{PP \times Fs}{Fsys} \right\rceil$

C) $\text{SharePaths} \leq \left\lfloor \frac{Fsys}{PP \times Fs} \right\rfloor$

Figure 4.8: Different SPC optimizations that A) pipeline B) parallelize and C) share resources between the Signal Buffer (SB), Computational Element (CE), and the summation (Σ) components. The inequalities define the limits for the optimizations where PP is the number of propagation paths, Fs is the sample frequency, and $Fsys$ is the system clock frequency.

for all the instantiated hardware is determined as is the route each signal path will take. This defines the network load for each high-speed transceiver and the amount of resources utilized on each FPGA. Every point on each graph in the results section represents a routed system configuration where every signal path component is placed on an FPGA and all the signals to and from each component are routed through the network. To support this claim the fundamental building blocks of the DWCE system, the signal path component and the GTX network, are analyzed to determine performance. The SPC and GTX size, throughput, and timing results are then used by the placement and routing algorithms to create all the systems for each configuration.

4.2.1 Signal Path Component Optimizations

As discussed in Section 3.2, the SPC is the fundamental building block that is replicated many times to build large systems. Because of this replication the SPC was designed to be as small as possible and optimally utilize the resources available. The core can be pipelined, parallelized, and shared to this end. These different optimizations are shown in Figure 4.8. To pipeline the component the same hardware is used to compute each of the propagation paths by changing the timing of the delivery of parameters with each clock cycle. For each system clock cycle a new propagation path is computed and passed into an accumulator. This works well, as long as the number of propagation path computations multiplied by the sample frequency of the signal is less than the system clock frequency. When this inequality breaks, it is necessary to parallelize the computations. Parallelization is done by adding more computational sections of the component and then pipelining the computations through each parallel section. The amount of parallel stages necessary is equal to the number of propagation paths multiplied by the sample frequency and divided by the system clock frequency. Finally, the SPC hardware can be shared among multiple signal paths. This is done by duplicating the signal buffer and multiplexing each signal path through the signal path computational element. This will work as long as the number of signal paths through the core multiplied by the sample frequency and the number of propagation paths is less than the system clock frequency. This customize-ability is explored in Section 4.2.4 where various configurations of the signal path are built and run through the Xilinx tool-chain to determine the size of the signal path core. These results were fit to two equations that are used to interpolate and extrapolate the size of any realizable signal path core. To perform the fit, the equation was applied to every pair of results and the average of the scalar parameters were used in the final equation. The algorithms used to place and route each system determine which optimization is best and then use the interpolation results to estimate the size of each SPC. All of these optimizations allow for a good use of FPGA resources depending on the desired configuration of the WCE.

While the amount of SPCs that will fit onto a single FPGA can be estimated using the SPC sizing results, it is difficult to determine the maximum system clock frequency that can be attained. To better estimate this, a full system was created in VHDL and run through the Xilinx tool chain that utilized 80% of the DSP and BRAM resources and 70% of the available LUT-FF pairs. It placed 32 signal path components on the FPGA each emulating 15 Multipaths at a sample frequency of 40 MHz. This build was able to meet timing for a system clock speed of 240 MHz. A key to reaching this speed was intelligent placement of registers along the inputs to the signal path components and ensuring that the synthesis tools did not optimize out these registers. It was not possible to build every single system configuration presented here because of the time it takes for the Xilinx tools to realize each design.

4.2.2 $(V, 8, 2)$-Network Configurations

The first two configurations are created from a $(V, 8, 2)$-network as shown in Figure 4.6. The first places a single analog front end on each radio node. This configuration can build networks of 9 to 51 AFEs and allows for the highest sample frequency and the most number of propagation paths emulated. Nine is the minimum number because with eight and fewer RNs each RN can be connected to a single CN making a trivial network. The next system utilizes two AFEs per radio node to increase scalability. This allows for networks of 18 to 102 radios to be connected. For each of these configurations many systems are analyzed to determine the maximum sample frequency for the system. For each system the algorithm in Figure 3.8 is used to place Signal Path Cores on each FPGA in order to distribute the network and computational load evenly. This algorithm will not produce an ideal solution and can fail to find a solution even if one exists; however, if it does find a solution it is guaranteed to be a solution that can be routed. This fact causes some non-linearities in the results.

There are two keys to reducing network load, one can occur before a signal is processed by a SPC and one can occur after. If a signal path has not been processed, it's data is identical to every other non-processed signal path that originates from the same AFE. If multiple non-processed signal paths from the same AFE go over a single network connection, only one signal path worth of data need be sent. The second key is a result of the fact that once a signal path has been processed by a SPC, it can be summed with other processed signal paths that have the same destination AFE. If they are summed prior to being sent over a connection, the amount of data over that connection is reduced. An objective for these algorithms is to place SPCs on certain FPGAs to take advantage of these two keys and minimize network load. For the $(V, 8, 2)$-network, the optimal placement of signal paths from a network load perspective is to place all of them on the compute nodes. For configuration one, this setup requires only one signal path worth of data be sent over each network connection; however, this is not an ideal setup. This setup would leave about half the FPGAs without any SPCs and drastically under-utilized.

Section 3.1.5 briefly described the load balancing algorithm as shown in Figure 3.8. This paragraph describes how that load balancing algorithm is used for the scalability study. The goal of the $(V, 8, 2)$-network load balancing algorithm shown in Figure 3.8 is to start with all SPCs on the CNs and move signal paths from compute nodes to radio nodes to balance the computational load while ensuring that the network does not become overloaded. On a radio node there are two distinct paths the signal takes. The ingress path is defined as the path where signals are received from the AFE and sent out to the GTXs. The egress path is defined as the path where signals are received from a CN and sent to the AFE. When moving a SPC from a CN to a RN it is either put in the ingress path or the egress path. If the SPC is put on an ingress path then there is an additional signal that is being produced that must be sent from the RN to the CN. If instead the SPC is moved to the egress path, an unmodified signal must be sent to the RN that has not gone through a SPC and therefore cannot be summed at the CN. This results in additional traffic through the network from the CN to the RN. Therefore, it is possible to balance the SPC load by moving SPCs from CNs to RNs at the tradeoff of adding load to the network. Moreover, the network load from RN to CN and from CN to RN can be balanced by choosing to place a SPC in either the ingress or egress paths of the RN. The load balancing algorithm is not guaranteed to find a solution. In fact it can succeed or fail based on the random seed that is given to the ordering of the SPCs. This was a good solution that works in most cases and will not produce any false positives but may produce false negatives.

The algorithm to find the maximum sample frequency for a given $(V, 8, 2)$-network system is shown in Algorithm 1. Each system is defined by the number of AFEs and the number of propagation paths it has. For each system, the amount of SPCs that can fit on an FPGA and the number of signal paths that can go over a network is determined from the sampling frequency. For these parameters (signal paths per connection and SPCs per FPGA), the load balancing algorithm is run. If it succeeds then the sampling frequency is increased and if it fails the sampling frequency is decreased. This is done until the boundary between success and failure is found, indicating the maximum sampling frequency. These results are presented in the results section.

Algorithm 1: $(V, 8, 2)$-network routing and placement

```
Result: Multiple placed and routed systems and the maximum sample
        frequency ($Fs$) for each
for Radio Nodes ← 9 to 51 do
    for Propagation Paths ← 1 to 30 do
        initialize $Fs$
        while maximum Fs not found do
            determine max SPs per FPGA given $Fs$
            determine max network load given $Fs$
            run the Load Balancing Algorithm (LBA)
            if LBA successful then
                increase $Fs$
            else
                decrease $Fs$
        store $Fs$
display results
```

4.2.3 Hypercube Network Configurations

The third and forth configurations use a four-dimensional hypercube network as shown in Figure 4.7. This was avoided previously because of the large number of hops a signal path will take and because it was thought that the network would be a major limiting factor. However, this is a good network to reach a maximum number of AFEs, because every FPGA can utilize all available eight connections and the maximum size of the hypercube network is technically only limited by resources. Thirteen different hypercube configurations are analyzed starting with a $2 \times 2 \times 2 \times 2$ 2-ary 4-cube network. One of the smallest dimensions is incremented by one with each setup until a network of $5 \times 5 \times 5 \times 5$ (5-ary 4-cube) is reached that would have 625 FPGAs.

The challenge for routing a WCE system with a hypercube network is to determine the route each signal path will take and the FPGA that each signal path will be processed on to minimize the number of signal paths going over each connection. The routing algorithm first finds every possible route for each signal path which has a minimal number of hops. Next the algorithm starts placing routes, starting with the signal paths with the smallest number of hops, until all signal paths have been routed. The two keys to minimizing network traffic presented in Subsection 4.2.2 also apply to these configurations. For a given set of possible routes, each route will likely have connections with signal paths that originate from the source AFE and also connections which have signal paths that are destined for the same sink AFE. Routing the signal path over these connections will not add to the network load, which is taken into account when determining the network load for each route. Next, the algorithm selects the routes that add a minimal amount of a network traffic. For the selected routes there can potentially exist several FPGA nodes that the SPC could be placed and maintain this minimal network load. The FPGA which has the smallest amount of SPC cores already on it is selected along with the corresponding route.

After all iterations are finished, all the signal paths have been routed. The maximum number of signal paths that a connection must support is used to determine the maximum sample frequency that the network can be realized. Finally, the number of propagation paths are iterated from one to ten and for each, the maximum frequency is found based on the number of signal paths on each FPGA. Whichever frequency is lower is the maximum frequency that can be realized for that system configuration.

Algorithm 2: Hypercube network routing and placement

```
Result: Multiple placed and routed systems and the maximum sample frequency
        ($Fs$) for each
for All Hypercube Configurations do
    sort signal paths by minimum number of hops
    for each ordered signal path do
        find all minimum hop routes
        for each min route determine network load
        select the routes with minimum network load
        increase the max network load by one if needed
        for each selected route do
            for each min-network load FPGA location do
                if SPC Load < Curr Max Load then
                    selected route ← current route
                    Curr Max Load ← FPGA SPC Load
        store selected route
        store added network and SPC load
    for Propagation Paths ← 1 to 10 do
        find max $Fs$ based on max SPCs on FPGA
        ensure max network load can support $Fs$
display max $Fs$
```

4.2.4 Signal Path Component Results

The signal path component sizing results are shown in Table 4.2. The data from each entry in the table comes from a synthesized and mapped SPC. In order to estimate the size for any given number of shared signal paths, parallel paths, and propagation paths, a linear fit was done to this data. Two different fits were done, one for parallel paths and propagation paths, and one for shared SPs and propagation paths. The signal buffer and parameter delivery cores have to be changed between

Shared SPs	Parallel Paths	Prop Paths	LUT-FF Paris	DSP Cores
1	1	5	3304	19
1	1	10	4304	19
1	1	30	7056	19
1	2	10	6329	34
1	2	20	8158	34
1	2	30	9622	34
1	3	15	9429	49
1	3	30	11331	49
1	1	5	2959	19
2	1	5	3523	19
3	1	5	3838	19
1	1	10	3578	19
2	1	10	4726	19
3	1	10	6698	19
10	1	30	17173	19
10	1	1	3455	19

Table 4.2: Signal Path Component resource utilization

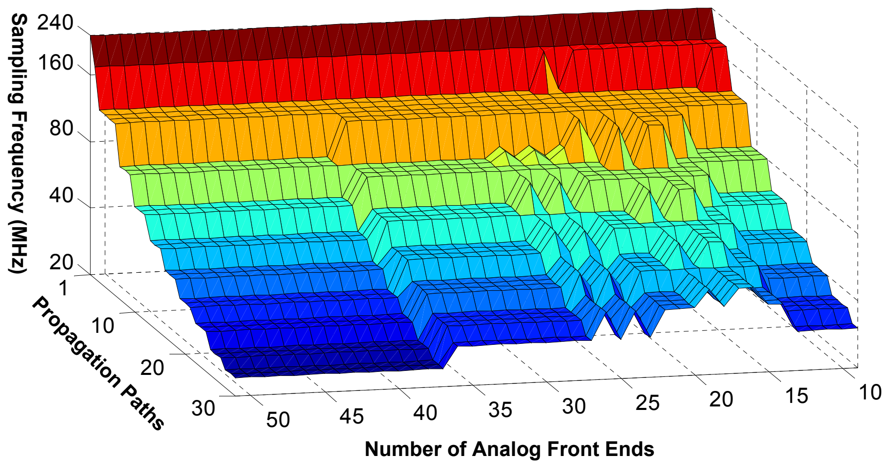
the two builds which causes the difference in sizes, even when both have the same parameters. Equation 4.1 shows the equation with the fit results for the SPC with parallel paths and Equation 4.2 shows the equation and results for the SPC with shared paths. Equation 4.3 was also derived from the results and used to extrapolate the number of DSP cores. These equations are used when estimating the SPC sizes in the routing and placement algorithms.

$$\text{LUTFF} \approx (\text{PropPaths} \times \alpha) + (\text{ParallelPaths} \times \beta) + \gamma \tag{4.1}$$

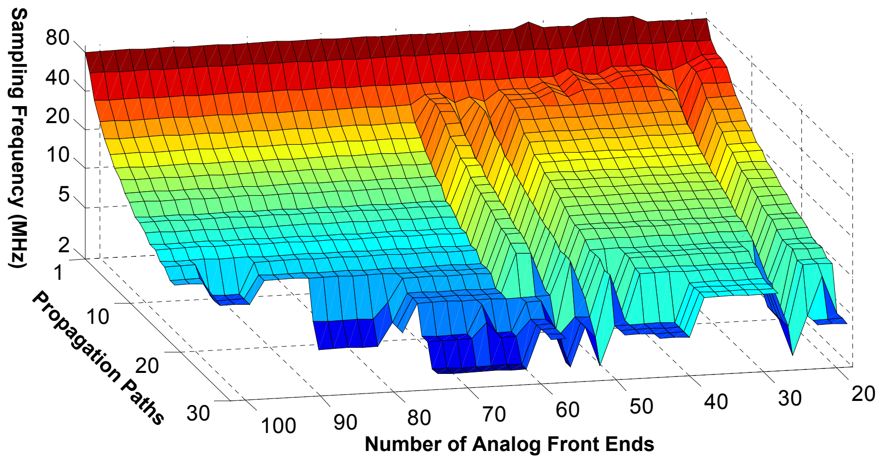
$$\text{where: } \alpha = 153 \quad \beta = 2025 \quad \gamma = 781$$

$$\text{LUTFF} \approx (\text{PropPaths} \times \text{SharedSPs}) \times 141.5 + 2030 \tag{4.2}$$

$$\text{DSPs} = (\text{ParallelPaths} \times 14) + 5 \tag{4.3}$$



(a)



(b)

Figure 4.9: Scaling study results for configurations one (a) and two (b)

4.2.5 (V,8,2)-Network Configurations Results

Figure 4.9 shows the scaling results for the $(V, 8, 2)$ Network configurations. These two plots show a plane where everything on and below the plane is a configurable system and everything above the plane is not likely to be possible. The exception to this rule is at the lower AFE numbers. It is possible to add compute nodes to the system by using a larger $(V, 8, 2)$ network (up to a $(51, 8, 2)$ network) and increase both the sample frequency and number of propagation paths. Graphs on this are not shown because they generally create system parameters that are much larger than what is needed by a wireless device. What is shown is the minimal amount of hardware required to form a given size AFE system, and the maximum configuration parameters that can fit on this hardware. Note that the z-axis, the maximum sampling frequency, is shown in a log scale.

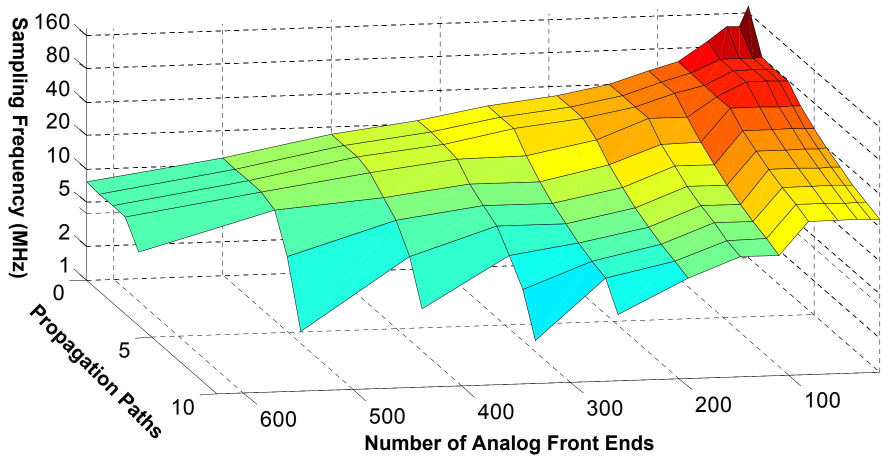
The most intriguing aspect of these results is they seem to go counter to the $O(N^2)$ relationship associated with AFEs and computations. Generally, it is possible to keep close to the same sample frequency and number of propagation paths while increasing the number of AFEs. Sometimes the maximum frequency increases while increasing the number of AFEs which is completely counter-intuitive. In fact, this is not breaking the $O(N^2)$ relationship. The number of FPGAs and computational resources are increasing as the number of AFEs are increasing. When the sample frequency increases with an increase in AFEs, the amount of resources available grows faster than $O(N^2)$. This result shows that both the amount of resources required and the size of the FPGA cluster are growing at approximately the same rate. This makes this configuration ideal for moderate sized networks. It is possible to shrink or grow the WCE based on the demands of the wireless device under test by adding or removing hardware, reconnecting the network, and reconfiguring the FPGA image by simply changing generics.

Another notable result are the large jumps and oscillations seen in the maximum frequency between various points on the graph. This is showing the realities of building the system. The jumps are due to different configurations of the signal path. On a low side of the jump, the system is not efficiently configured. Perhaps the maximum frequency allowed for nine propagation paths but ten were needed, so the signal path component had to create an additional parallel resources. In this case cycles would be wasted not computing any propagation paths. On the high side of the jump, it could be close to ideal resource utilization where the maximum frequency would allow for all ten paths to be computed in a single pipelined structure. The same can happen for using multiple signal paths with the same signal path core. In configuration two, there seem to be two planes with the maximum frequency jumping between them. For the higher frequencies only one signal path is routed through the signal path components and on the low frequencies, two signal paths must be routed through one signal path component.

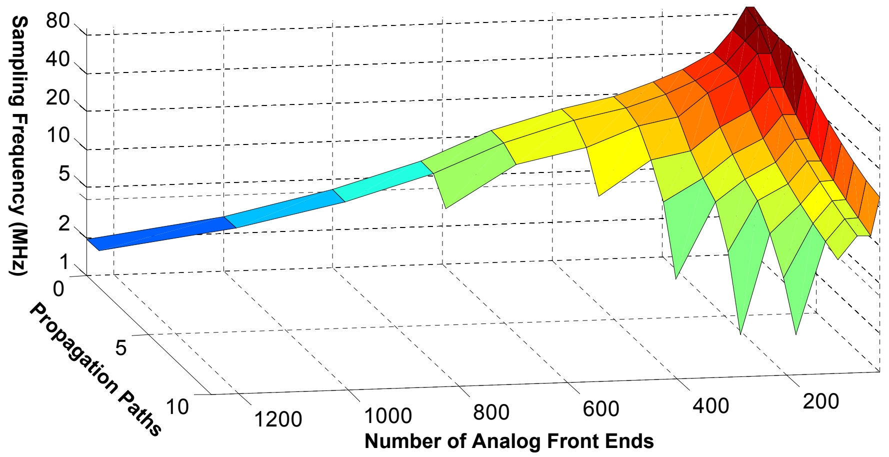
Lastly, there are places where no data is plotted. This indicates that it is not possible to create that particular system with the current configuration. In these cases, the required logic resources were greater than the available logic resources regardless of the sample frequency. This was a surprising result as it was generally assumed any system could be routed given a low enough sample frequency. The reason for this result is the fact that multiplexing and parameter storage require logic resources which increase with the sharing of signal paths and the number of propagation paths independent of sample frequency. At some point these resources alone take up the entire FPGA and no amount of reducing the sample frequency will help this issue.

4.2.6 Hypercube Network Configurations Results

The results from the hypercube network configurations in Figure 4.10 show the capability to reach a much larger number of analog front ends. These results show that with configuration four, a system of 1250 AFEs is feasible with a signal bandwidth of 1.9MHz with a single propagation path computation. As with configurations one and two, if a solution was not found there were not enough logic resources and it was not plotted in the chart. Additionally, there are systems plotted with "0" propagation paths. The points here represent the maximum frequency that the network can support. Plotting these points makes it possible to see if a system is network limited or resource limited. It may be possible to construct a lower-fidelity SPC that will fit to meet the sample frequency if desired. One anticipated trend seen in these results is the decrease in maximum sample frequency as the number of AFEs are increased. This shows that as the number of AFEs are increased, the computations and network



(a)



(b)

Figure 4.10: Scaling study results for configurations three (a) and four (b)

throughput required for the system increases at a faster rate than the amount of FPGA resources.

The limiting factor with the hypercube network was anticipated to be the network or the FPGA DSP elements, not the FPGA logic. The results show that it could be either the network or the FPGA logic depending on the number of propagation paths. The key to eliminating the network load is all due to the optimizations done to reduce the data sent over the network as described in Section 4.2.2. Without these techniques the maximum sample frequencies are much lower, decreasing them to a point that would make the largest systems unusable by all but narrow band wireless systems which are fast becoming outdated legacy devices. With the reduced network load the logic elements become the limiting factor. As mentioned this is due to the parameter delivery component. A future work could be redesigning the parameter delivery to use BRAM components to alleviate the limit on logic. This could push the limiting resource to the DSP cores and increase the maximum frequency over what is presented here. This would reduce the number of BRAMs available for the signal buffer, which will reduce the maximum propagation delay of the system.

The hypercube network allows the cluster to reach very large numbers of FPGAs; however, this comes at a cost of propagation delay. This increases with the size of the system in a hypercube network configuration. Each network interconnect adds 132 nanoseconds per hop the signal path takes. In the largest system the maximum number of hops is eight which corresponds to 1.056 microseconds. As a point of reference, the propagation delay through the combined transmit and receive path of the analog front end is measured to be approximately 1.4 microseconds. So, while any increase in propagation delay is a negative, the delay added by the number of hops is not as significant as it was originally feared to be.

4.2.7 Cost Effectiveness Study

Part of the question asked in the introduction section was if the DBES could scale cost effectively. The answer to this question would be yes if the DBES could be constructed for eighty percent of the cost of conducting one actual field test. There is a reason this fraction was selected. Based on experience running the BES (the fourteen node analog wireless channel emulator) and on conducting field tests, the cost of running a lab test is roughly ten to twenty percent of the cost of running the equivalent field test. This includes the time it takes to setup the radios, generate scenarios, run the scenarios with the radios, and analyze the results. So, if it costs eighty percent of the cost of a field test to build the system, then the cost of building a DWCE and running an in-lab test would be approximately the same as the cost of running an equivalent field test. The added benefit is the investment in the DWCE would pay off in reducing the cost of all subsequent testing by roughly eighty to ninety percent.

To see this comparison, the estimated hardware cost for each of the configurations presented was plotted along with the cost estimate to run a field test for a given number of wireless devices. Based on previous field tests, the estimate was approximately five million dollars to run a month long thirty-two radio test. To extrapolate this estimate, it is assumed that the cost of a test would scale linearly with the number of nodes to test. This assumption is made because if you double the number of nodes then you need twice the number of people and equipment to run the test so the cost would roughly double. The hardware cost was estimated by counting the number of FPGA nodes, AFEs, and chassis required and multiplying each by an estimated cost that includes peripherals as shown in Table 4.3. Figure 4.11 shows the results of this comparison. Note that the estimated cost of a field test was plotted at five, ten, and twenty percent of the actual estimate so that it would bracket the estimated hardware costs the different DWCE configurations.

Table 4.3: Unit costs used in hardware estimate

component	item cost	peripherals adding to cost
FPGA Node	\$12k	RTM board, MiniSas cables, usb cables
Chassis	\$11k	power supply, ethernet cables, clocking cables
AFE	\$3.5k	TRCal board, RF cables, μHDMI cables

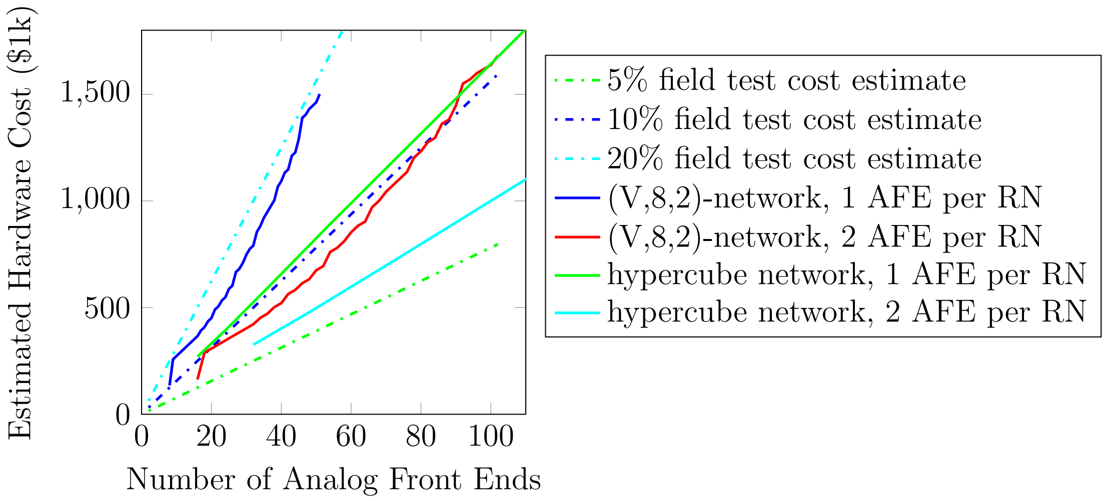


Figure 4.11: Estimation of DWCE hardware costs

This figure shows that the hardware costs of building a DWCE is roughly five to twenty percent of the cost of conducting a field test. Notice that the cost increase of adding nodes is linear or close to linear. This shows excellent scaling considering the $O(N^2)$ relationship of nodes to computations. This means that the configuration naturally allows for reduction in propagation paths or signal bandwidth to linearly increase the hardware costs with the number of nodes. If the desire was to keep the fidelity of the emulation constant while increasing nodes then a $O(N^2)$ relationship would; however, in talking with potential users of the system, they would trade fidelity for scale as they are typically interested in high-fidelity scenarios with low numbers of nodes and lower fidelity scenarios with higher numbers of nodes. Also important is that this cost estimate does not include the labor cost to actually build the system;

however, this number would be small in comparison to the hardware costs considering that the development of the system has already been done. Even if this is estimated at ten percent the cost of a field test, which is very high, then the total cost of building a DWCE and conducting the in lab test is less than half the cost of running the equivalent field test. While these are rough estimates and many of the exact numbers could be argued, this shows how economical it is to use a DWCE of this architecture as compared to going into the field.

4.3 Fidelity Study

The fidelity study is immensely important to the development and the use of the DBES. This study has been ongoing throughout the design and development stages of the DBES. At every point the question was asked: How accurate is this system to the real world environment that it is emulating? This question is extremely challenging to answer due to the infinite number of scenarios that the system attempts to emulate and multitude of current and future RF devices that could utilize this system. The answer to this question has been quite negative at times and steps were taken in the hardware and software design of the system to mitigate certain problems. If this study had not been concurrent with the design and development, the situation would have occurred where hardware was up and operational, but radios began to exhibit poor performance for unknown reasons. This rationale leads to another purpose for this study: user confidence. If a third party RF device developer utilizes this system, it will be easy for the developer to accuse the DBES of making their RF system perform poorly. Proving that this is or is not the case is a beneficial outcome of the fidelity study. There exist many scenarios that the DBES cannot replicate and being able to determine if a given situation falls into one of these corner cases or not is crucial.

In order to answer the fidelity question, the problem was broken down into three stages: theoretical, experimental, and observational. The theoretical fidelity study focuses on evaluating DBES fidelity using mathematical analysis and theoretical simulations. The experimental fidelity study evaluates the DBES itself by using RF signal generators to create signals and observing the effect the DBES has on those signals. Many of the theoretical predictions are validated with actual hardware in the experimental section. Technically, the combination of theoretical and experimental should prove the fidelity of the system for a majority of cases. However, performing a test with actual RF devices in the field and with the DBES in the lab will provide an observational result that is much easier to understand. This final study had to be observational because it is very difficult or impossible to obtain a control metric for the experiment depending on the scenario. For the same reason that no two field tests are identical, the field test will not be identical to the lab test. To illustrate this point imagine and event where a car randomly drives by a test event. This car could induce a multipath effect that changes the connectivity between a pair of radios and thus the behavior of the entire network. To compound the problem real-world results that could be obtained for the radios employed in this experiment were considered sensitive information and could not be published. For these reasons, the final part of the test is observational only and DBES behavior is not compared with a control; however, predictions are made wherever possible and the results are analyzed based on those predictions.

4.3.1 Theoretical Fidelity Analysis

The theoretical fidelity of the DBES is analyzed in this section. All the functionality of the DBES can be placed into three categories: signal power fidelity, signal delay fidelity, and signal summation fidelity. Any issue with the fidelity of the signal produced by the DBES can also be placed into one or more of these categories. For each of these categories components are analyzed to determine their contribution or detriment to the fidelity of the system.

To analyze the various components the signal path has been modeled and thoroughly analyzed to determine it's noise characteristics. The theoretical analysis consists of adding randomly distributed noise in various scenarios to each of the components in the signal path and running a simulation to see the effect on the output signal. Equation 4.4 is the signal path equation and equation 4.5 is the signal path equation with errors added in.

$$y_j(n) = \sum_{s=1}^{S} \sum_{m=1}^{M} \alpha_m x_{si}(n - d_{mi}(n)) e^{-\mathrm{j} d_{mi}(n) \omega_{rf}} \tag{4.4}$$

$$y_j(n) = \sum_{s=1}^{S} \sum_{m=1}^{M} \{[(\alpha_m(n) + \hat{\alpha}_m(n))(x_{si}(n - d_{mi}(n) + \hat{d}_{mi}(n)) + \hat{x}_{si}) + \hat{Q}1] \\ e^{-\mathrm{j}(\omega_{rf}(d_{mi}(n)+\hat{d}_{mi}(n))+\hat{Q}2)} + \hat{Q}3\} \tag{4.5}$$

All error variables are designated with a hat. Each of these variables will be discussed in turn and how they are modeled. The errors associated with signal power fidelity are quantization noise ($\hat{Q}$), and attenuation noise ($\hat{\alpha}$). The errors associated with signal delay fidelity are delay interpolation error ($\hat{d_i}$), signal interpolation error ($\hat{x}_{si}$), and Doppler noise which occurs due to the change signal interpolation error over time. The sources of error in signal summation fidelity occur due to the summation of the different propagation paths and signal paths $\left(\sum_{s=1}^{S} \sum_{m=1}^{M}\right)$. Some of these variables have a type of randomly distributed noise associated with it that will be described along with the rationale for using the various distributions. Other variables have error that is scenario dependant: meaning that the error can be nothing or it can be massive depending on the input parameters. For each of these scenario based noise components, different scenarios will be presented that show good, normal, and bad behavior. Approximations on the parameter thresholds for these different types of behavior will be presented.

The remainder of the theoretical fidelity study is organized as follows. The first section describes the simulation of the baseline signal path without the addition of error. Each subsequent section will describe each of the categories of fidelity with those sections broken up into subsections describing and showing results for each of the error sources in Equation 4.5.

4.3.1.1 Signal Path Simulation

This section discusses the simulation used to analyze the theoretical noise additions in the remainder of the sections. It first discusses the baseline simulation and it's setup and proceeds to show ideal results that act as the control for the remainder of the fidelity study.

Each source can be categorized as adding error or noise. The error sources do not distort the signal but produces an inaccuracy to the real world equivalent. Noise sources distort the signal when present which can interfere with the radios ability to decode the message if it is too large.

The baseline signal path simulation does not explicitly add in noise. The use of double precision drastically reduces all the $\hat{Q}$ parameters such that they are negligible. The attenuation error and the delay error are scenario dependant and do not produce noise so they will not be seen here. The only significant source or error that can be seen is the signal interpolation error; however, this is scenario dependant and only noticeable when Doppler is utilized. Thus the baseline will not include a delay change. This ideal simulation will be used as the baseline to compare how the real world noise additions affect the computations.

The simulation starts by setting up the signal of interest and the parameter set. It does this with a real signal in the RF domain and then downconverts the signal to baseband to get the in-phase (I) and quadrature (Q) components. A filter is used to remove the aliased high-frequency signal. The signal utilized is a sum of sinusoids at different frequencies across the bandwidth of interest. They are spaced far enough apart in the frequency domain to see errors to individual frequency components. The parameters used for this baseline consist of a single path with a simple attenuation and a small but constant delay that lines up with the sample rate such that no

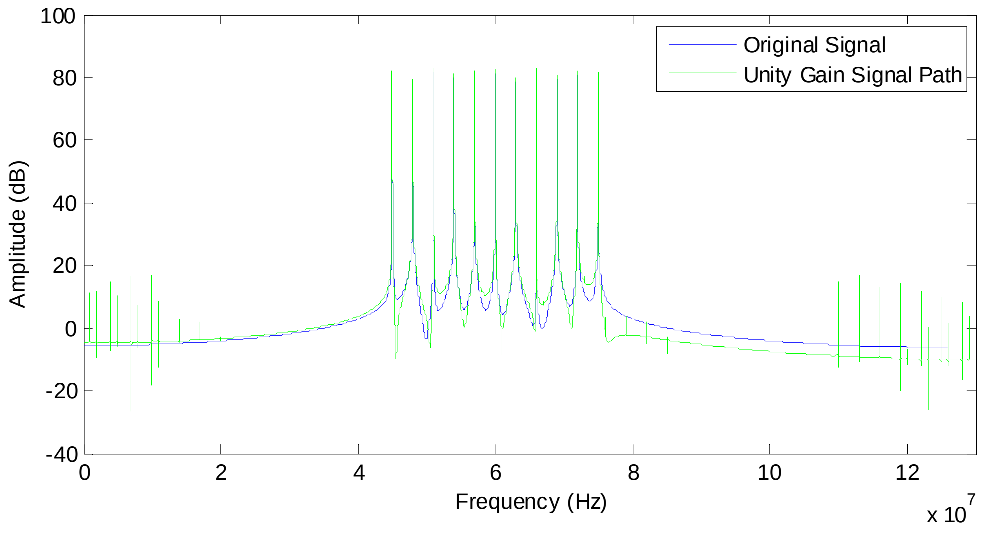


Figure 4.12: Signal path simulation showing unity gain

interpolation is done. Next, the signal is processed sample by sample and modified by the TDL equation. Once completed it is upconverted to RF again to produce the final result.

Running the simulation through the unity gain produces the same output as input as shown in Figure 4.12. The attenuation is confirmed in Figure 4.13. The Doppler shift was tested as well and the frequency offset was found to be accurate across a number of shifts. This does produce error however, so the results will be presented in Subsubsection 4.3.1.3.3. This gives good confidence that the simulation is set up correctly.

4.3.1.2 Theoretical Signal Power Fidelity Experiments

The sources of error associated with the signal power fidelity that can be simulated are quantization error and attenuation error. It is important to ensure these sources are significantly small, but as will be seen in the experiment signal power fidelity section (Section 4.3.2.1), these sources of error are insignificant as compared to hardware noise.

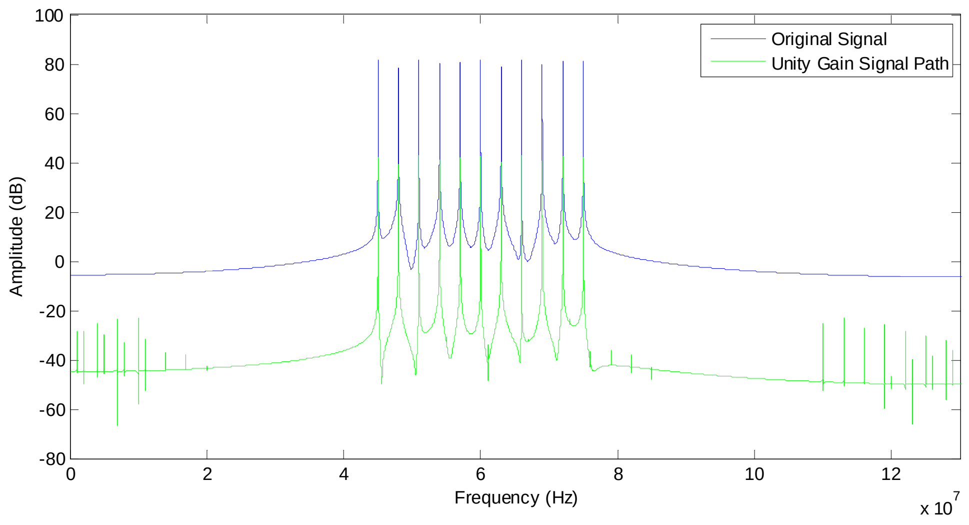


Figure 4.13: Signal path simulation showing 40 dB of attenuation.

4.3.1.2.1 Quantization Error

The quantization error ($\hat{Q}$) variables designate rounding errors due to fixed point multipliers. The quantization that occurs after each fixed point multiply can be modeled as a summation with a uniformly distributed random variable. The maximum error is half a bit in magnitude and is bounded as shown in equation 4.6 where *nbits* is the number of significant bits in the output of the multiply. These appear in three places in Equation 4.5. $\hat{Q}1$ is due to the attenuation multiplication, $\hat{Q}2$ is due to the delay multiplication, and $\hat{Q}3$ is a result of the complex exponential multiplication.

$$-2^{-(nbits-2)} < \hat{Q} < 2^{-(nbits-2)} \tag{4.6}$$

One thing not explicitly shown in the full noise equation (Equation 4.5) is that the alpha multiplication is more complicated than a fixed bit multiply. In fact the attenuation parameter is split into attenuation shift and attenuation mantissa parameters such that $\alpha = \alpha_m * 2^{-\alpha_s}$. Doing this results in less resource utilization on the FPGA because a shift uses very little logic as compared to a large bit-width multiplier. As a result there are actually two $\hat{Q}$ parameters associated with the attenuation multiplication and it can be written as in Equation 4.7. This modification is important because $\hat{Q}1b$ is the significant noise source as compared to all the quantization errors.

$$\alpha * x_{si} + \hat{Q}1 \Rightarrow ((\alpha_m * x_{si}) + \hat{Q}1a) * 2^{-\alpha_s} + \hat{Q}1b \tag{4.7}$$

A simulation was run comparing the baseline signal path with a simulation introducing quantization error. The number of bits used in the multiplication for the Q errors are as follows: $\hat{Q}1a$ has 16 bits, $\hat{Q}1b$ has 23 bits, $\hat{Q}2$ has 14 bits, $\hat{Q}3$ has 23 bits. With the parameters set at unity gain, there is no discernable difference between the simulation with and without the error introduced. It is only noticeable when the attenuation parameter is set to a large value. The $\hat{Q}1b$ parameter becomes noticeable and the noise floor becomes apparent as shown in Figure 4.14. This is not truly added by the digital system however, as there is a fixed bit width the analog front end supports. As the signal gets attenuated more and more, it will begin to drop below the noise floor and disappear. This is actually intended behavior as it is exactly what a low-power signal being received in the real world would look like.

4.3.1.2.2 Attenuation Error

The $\hat{\alpha}_m$ variable is the error in the attenuation as compared to the real world attenuation. The error is due to the fact that the attenuation parameter is only updated at the parameter rate which is much slower than the sample rate. To characterize the error in this instance, a random variable is used to represent the error of the current parameter for any given sample period. The quantity of error is dependent upon:

1. the scenario selected
2. where the parameter period is located in the selected scenario
3. where the sample period is located in the selected parameter period

Since there are quite a variety of possible scenarios (nearly infinite) and the exact parameter period within a given scenario will vary as will the sample period within the

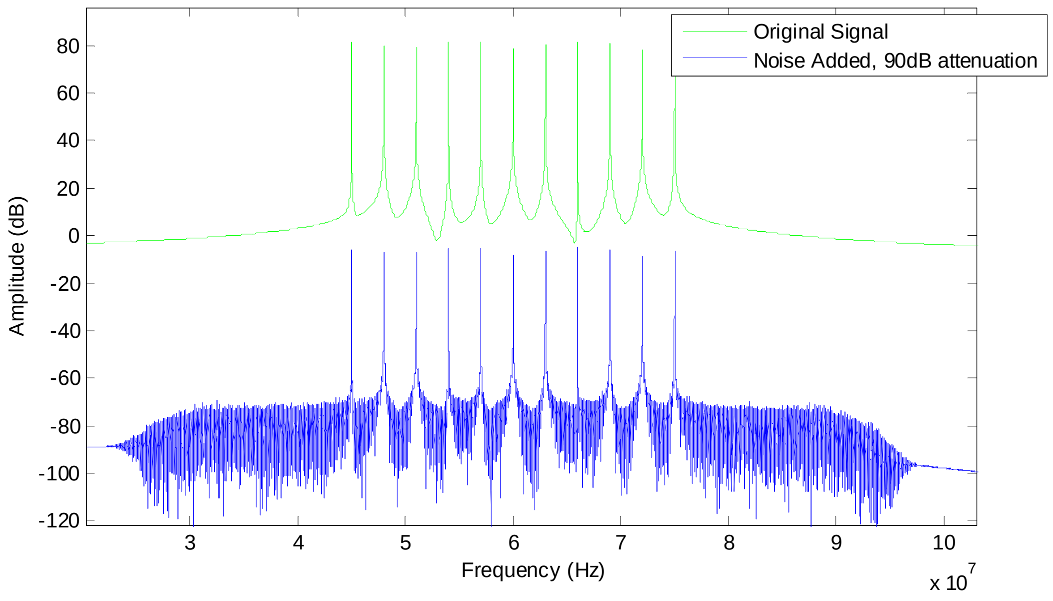


Figure 4.14: Signal path simulation showing quantization error with 90 dB of attenuation.

parameter period, a random variable will be introduced to represent the inaccuracy. It will vary over the range of possibilities for the given parameter period. Depending on the attenuation of the channel over time for the specific scenario and given the listed assumptions, the maximum error within the pth parameter period for path m can be described in equation 4.8.

$$max(\hat{\alpha}_m(n)) = \alpha_m(p) - \alpha_m(p-1)) \qquad \text{where:} (p-1)f_p < nf_s < pf_p \tag{4.8}$$

$\hat{\alpha}_m(n)$ is the estimated error for a given sample n within a given parameter period p for a given path m. f_p is the parameter update rate. f_s is the sample rate. Equation 4.8 will represent the maximum inaccuracy over the parameter period p. $\hat{\alpha}_m$ will represent a random variable for path m, which will be assigned a uniform distribution function within the bound specified by equation 4.8. Zero is the lower bound for the random variable, which is the case of a constant attenuation for two adjacent parameter periods. For a given scenario a uniform distribution may not be an ideal representation. Also, for a given parameter period of a given scenario the representation may not be ideal. However, for a random sampling of parameter periods across all possible scenarios, a uniformly distributed random variable for the attenuation error across the range of possible error quantities can be considered a reasonable approximation.

Next, the difference between two different attenuation values must be estimated. The worst case terrestrial scenario is two cars closing rapidly. Let's assume a free space path loss model as shown in Equation 4.9.

$$\begin{aligned} FSPL(dB) = & 20 * log10(d) + 20 * log10(f) + 32.44 \\ & d : \text{distance} \\ & f : \text{frequency} \end{aligned} \tag{4.9}$$

This formula is used in a scenario where two radios are closing at 250 miles per hour, which is much higher than expected terrestrially. The difference between steps at 1 millisecond intervals is taken which would be the maximum error as shown in Equation 4.8. The results are that at most we would expect to see a 1 dB error in attenuation and that is only when the radios are less than 10 meters apart. At 10 meters apart the error is .15 dB and it grows exponentially smaller as the radios are farther apart. This error is quite small and is not expected to affect radio performance.

4.3.1.3 Theoretical Signal Delay Fidelity Experiments

The theoretical signal delay fidelity experiments include the delay interpolation error, the signal interpolation error, and the Doppler error and noise experiments. Matlab simulations were used in each of these tests to determine the noise levels and accuracy of the various components to signal delay fidelity.

4.3.1.3.1 Delay Interpolation Error

The delay interpolation error is the inaccuracy caused by linear interpolation of the delay parameter throughout the parameter period, which is represented by $\hat{d}_{mi}(n)$

in Equation 4.5 for sample n of path m. As with the attenuation error, this error is dependent upon the scenario characteristics. However, unlike the attenuation, the delay is linearly interpolated at every sample period reducing the amount of error at each point. This interpolation analysis is very different than the signal interpolation analysis as an object moving through the real world will cause a small change in delay over the course of the parameter period of one millisecond. In fact, so long as there is no acceleration between two objects there is a linear change in the delay which will only produce quantization error in the delay parameter. To see how large this error could be, the same test as above was conducted where a car passes a second car at a closing speed of 250 mph. The cars pass at ten meters. This is a worst case scenario because as they pass there is a very rapid change in the second order derivative of the delay which produces a very large error. Additionally the close proximity that the cars pass produces a small delay. The combination of the two results in the largest percent error that will be seen in virtually any scenario. In this case the maximum error seen is 1.7 picoseconds. The delay at that point is approximately 10 nanoseconds. This means that the percent error is 0.17%. Given that this is a worst case scenario and the percent error is so small this is not anticipated to be an issue for the system.

4.3.1.3.2 Signal Interpolation Error

The signal interpolation error: $\hat{x}_{si}$ is complicated but can be best described by a distribution. It took a significant amount of research to determine the distribution of this signal and an interesting result is that it is very similar to the average of a normal and exponential distribution. The analysis was limited to a linear interpolation implementation. To characterize the probability density function the following steps were performed:

1. generate a very long signal

2. linearly interpolate that signal between many points

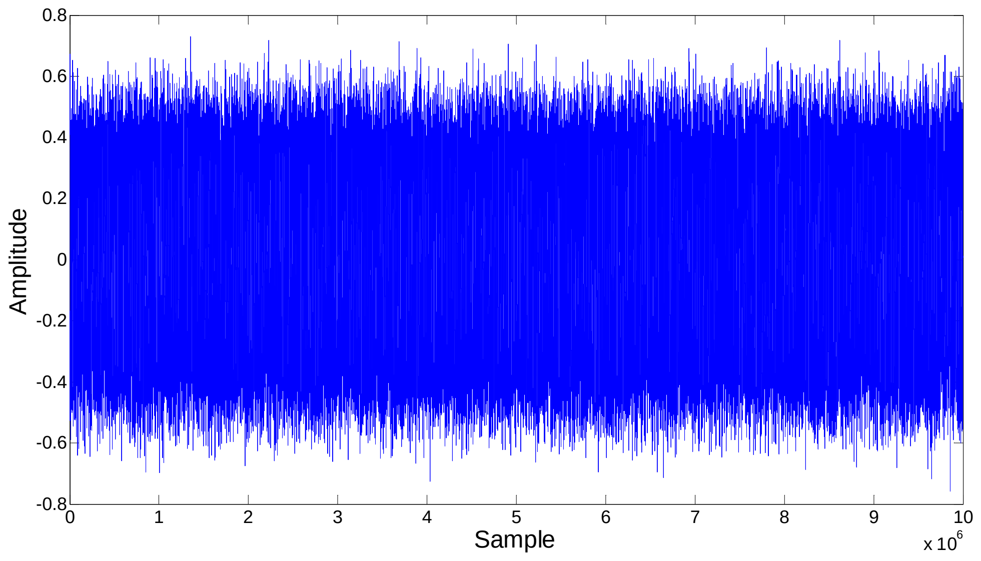


Figure 4.15: Signal used for the interpolation error analysis viewed in the time domain.

3. determine the error of that interpolation

4. produce a histogram of the error

5. characterize the shape of the histogram with standard distributions

The signal generated was bandlimited noise. This was done to statistically represent any bandlimited signal. In order to determine the error in the interpolation this signal was oversampled by a factor of 50. Figure 4.15 shows the time domain representation of the signal and Figure 4.16 shows the frequency domain representation. Figure 4.17 is an enlarged view of 4.16 showing just the signal of interest. By oversampling it is possible to pick to points on the signal that are 50 samples apart, interpolate between those two samples, and then determine the error in the interpolation by comparing that actual signal with the result from the linear interpolation. For these results this error calculation is done almost $500 * 10^6$ times which was as much a computer with 8 Gigabytes of memory could handle. In order to determine the Probability Density Function (PDF) of the error, a histogram of the error was

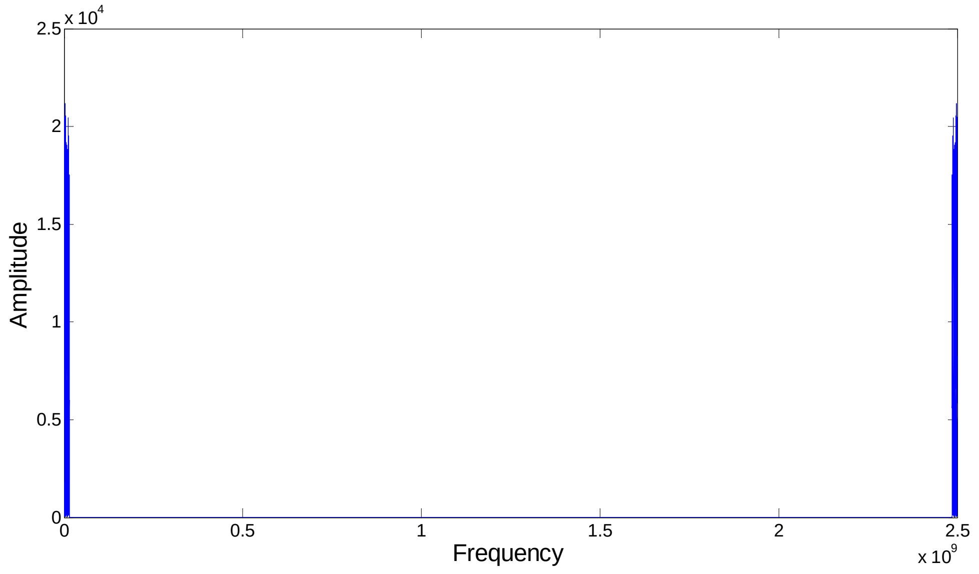


Figure 4.16: Signal used for the interpolation error analysis viewed in the frequency domain. It is oversampled by more than a factor of 50 to allow for a comparison with the interpolation

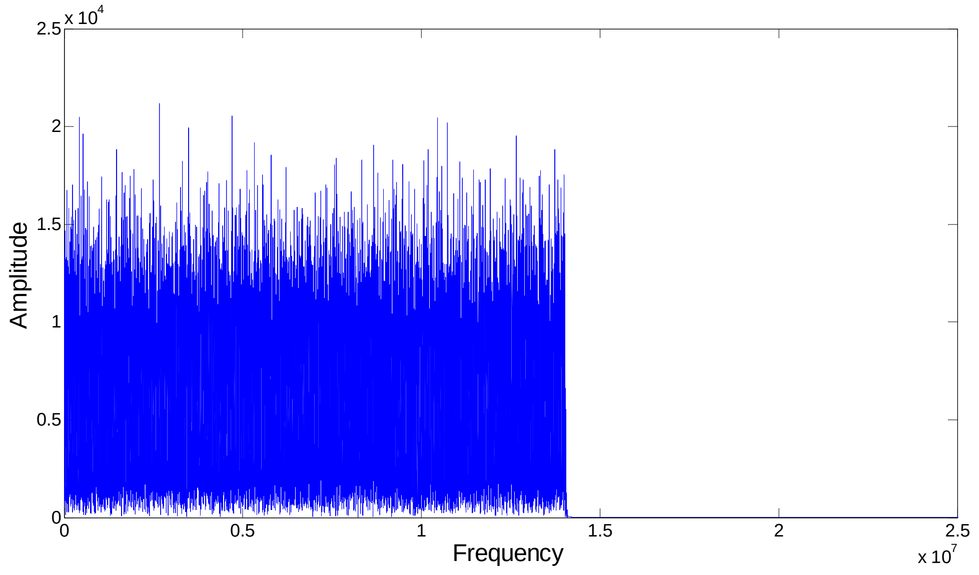


Figure 4.17: Signal used for the interpolation error analysis viewed in the frequency domain and enlarged to show the signal of interest.

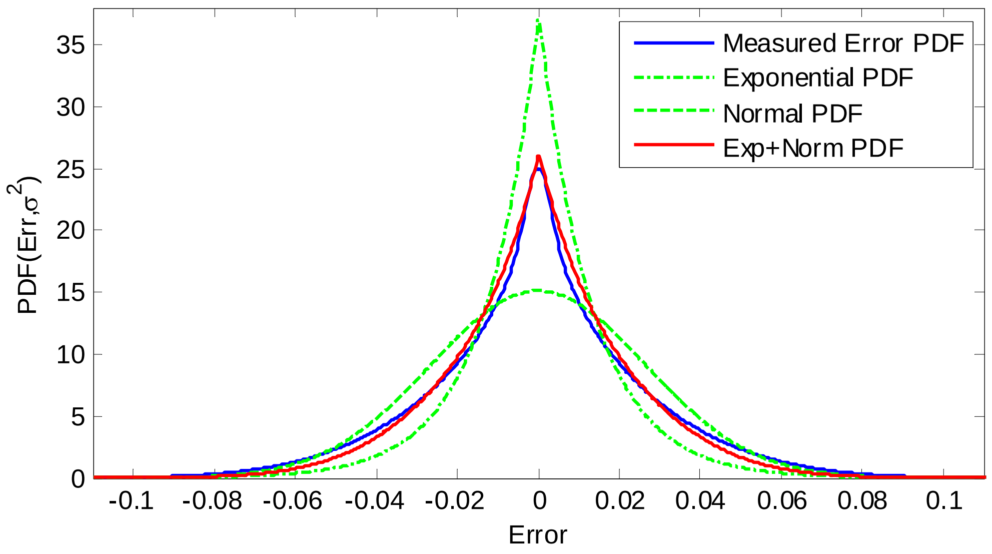


Figure 4.18: Distributions of the measured error and normal, first-order zero-mean exponential, and averaged normal and exponential PDF distributions with identical standard deviations.

calculated and normalized. The standard deviation of this histogram was used to show what a normal distribution and a first-order zero-mean exponential distribution would look like. Figure 4.18 shows these results.

The first-order zero-mean exponential distribution is not a traditional exponential distribution. Equation 4.10 shows the modified exponential distribution and Equation 4.11 shows a normal distribution. Both these equations are correct probability density functions as the integral from negative infinity to infinity with respect to x is equal to one.

$$\phi_{exp}(x,\sigma) = \frac{1}{2\sigma}e^{-\frac{|x|}{\sigma}} \tag{4.10}$$

$$\phi_{norm}(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x}{\sigma})^2} \tag{4.11}$$

These results show that neither an exponential or normal distribution accurately describes the error seen. The peak of the exponential was too high and dropped too quickly while the normal distribution peek was too low and rolled off too slowly.

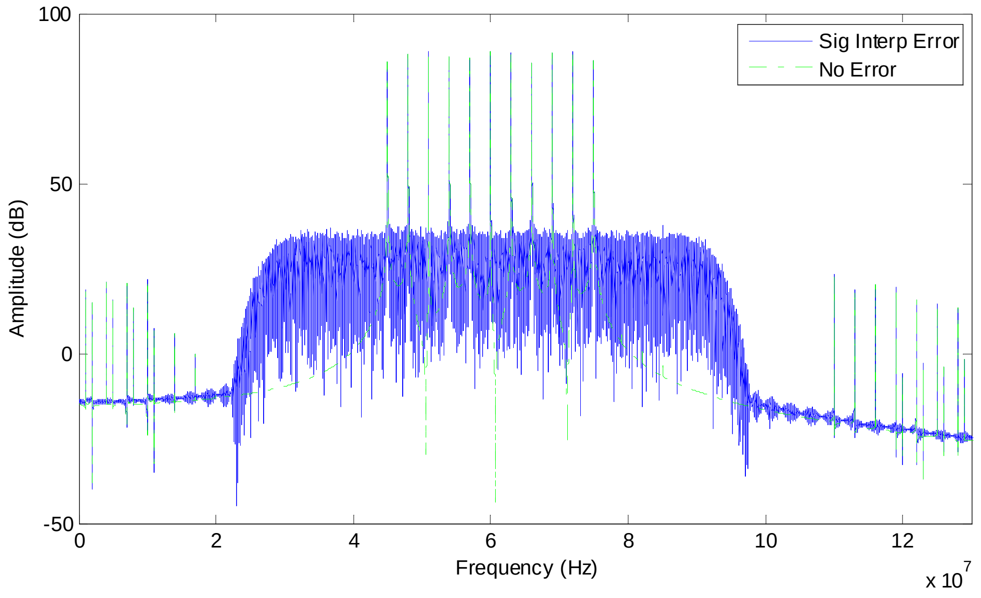


Figure 4.19: Theoretical noise difference when adding signal interpolation error

Because of this the two distributions were averaged. The result is very close to the distribution seen. The reason for this is unknown; however, it is beyond the scope of this dissertation to explain. This averaged normal and exponential distribution is close enough to describe the signal linear interpolation error noise and therefore it was used. Equation 4.12 shows this distribution.

$$\phi_{err}(x,\sigma) = \frac{1}{2}\left(\frac{1}{2\sigma}e^{-\frac{|x|}{\sigma}} + \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x}{\sigma})^2}\right) \tag{4.12}$$

The only parameter to this distribution is the standard deviation as the mean error should be zero. This experiment also reveals the value this parameter should be. Looking at Figure 4.15 the amplitude of the siganal on average is approximately one, varying between positive 0.5 and negative 0.5. Therefore, the standard deviation found from this can be linearly scaled with the amplitude of the signal in the signal path analysis. The standard deviation found from this experiment was 0.0264, which can be used with the averaged normal and exponential distribution to describe the linear interpolation error of the signal.

A random number generator was coded that had the distribution found above. This was used to add noise to the signal interpolation. The results are shown in Figure 4.19. This adds quite a large amount of noise; however, this simulation shows that it is around 50 dB down from the peak signal of interest. This happens to be the expected signal to noise ratio seen from the AFE analog transmit hardware. Therefore, if these simulations are correct, the noise is just small enough to not be noticeable on the real machine. This conclusion is good and will need to be verified with real-world tests. The next section 4.3.1.3.3 attempts to look at more specific scenarios as opposed to a general one to get a better idea of the error introduced.

4.3.1.3.3 Doppler Error and Noise

This section shows theoretical Matlab simulations run to determine how accurate the Doppler effect can be and how much noise is generated by inducing this effect. It is possible to be very accurate with the change in frequency and several tests were run that show that all frequency components are shifted to within the accuracy of the simulation. However, it was observed that the act of inducing a Doppler Shift adds noise to the signal. It is discovered that this noise is due to the signal interpolation error and that it is scenario dependant. While the distribution of error presented in Section 4.3.1.3.2 is a generalization no matter what scenario is given, the particular scenario of changing delay induces a specific type of error that can be very significant. It was observed in simulation that as the Doppler (or change in delay) was increased it induced a significant amount of noise. It was determined that this definitely comes from the signal interpolation, but the exact mechanism is not entirely understood and will not be described in this thesis. Instead, the error and the factors that induce the error are characterized and thresholds are presented where the error will not have an impact on the system's emulation of the RF environment.

Figure 4.20 depicts the Doppler error of a signal where the RF Center frequency is 60MHz and the Doppler is produced by two objects closing at approximately one

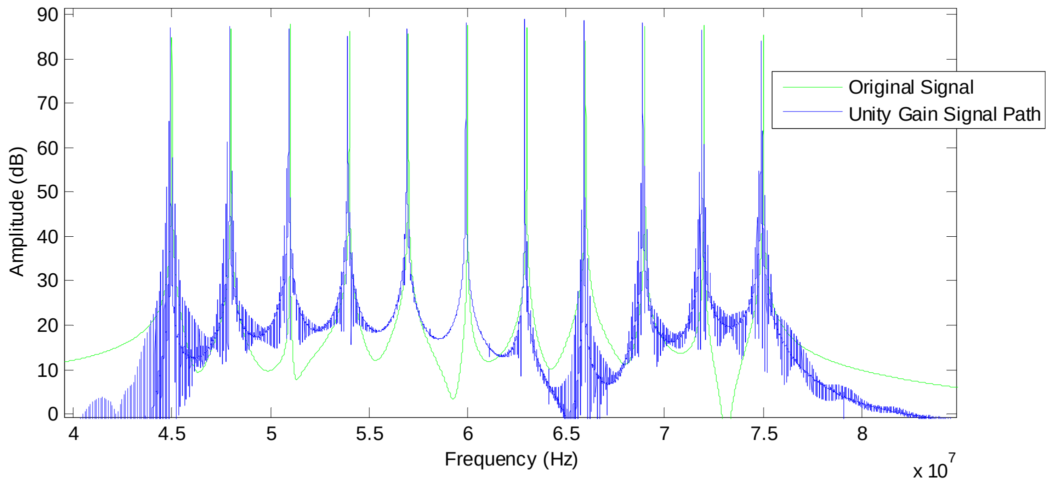


Figure 4.20: Signal of interest under heavy Doppler corresponding to one million miles per hour

million miles per hour. This extreme example is shown to help describe the issue. The first thing that is apparent is the multitude of spikes that is seen around each of the shifted sinusoids. The second thing to notice is that these spikes reduce in number and magnitude for the sinusoids at the center of the RF band. This observation can tell us a number of different things. First this effect is due to the signal interpolation. This becomes apparent when looking the baseband I and Q samples that the signal interpolation is performed on as shown in Figure 4.21. Stating the observation in another way: as the signal is farther from baseband the noise is increased. This can be explained as the lower frequency signals have more samples that represent their signal and thus there is less error in the signal interpolation. The higher frequency signals may only have three to four samples that represent the waveform resulting in a very high amount of error in the signal interpolation. This means that one way to reduce this error is to oversample the signal as much as possible. Unfortunately, there is a system limit to oversampling the signal based on the ADC and DAC clock rates of the AFE. Additionally, increasing the sample rate proportionally increases the network traffic and amount of computations the entire system has to perform so

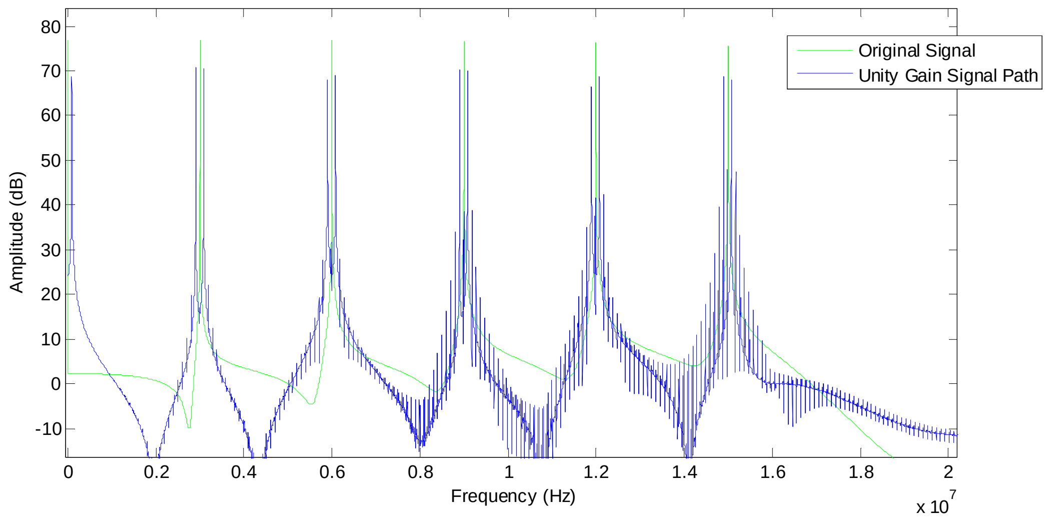


Figure 4.21: In-phase baseband signal of interest under heavy Doppler corresponding to one million miles per hour.

this must be done with care.

All this analysis has been done on a scenario that the DBES will never reproduce. The question is: under what scenarios does this noise impact the performance of the system? There is no firm answer to this question because the impact will vary from radio to radio and there is no hard line where the noise spikes, it gradually increases as the Doppler shift increases. A limitation in answering this with simulation, is that the resolution of the Fourier transform limits the ability to view the spikes. If very large amounts of samples are used the spikes are more visible. This is why only a estimate is possible.

The simulation examined the error in Doppler for terrestrial speeds. The good news is that with two cars moving toward each other at a closing speed of 250 miles per hour, the Doppler error appears to be minimal. These results are shown in Figure 4.22. For airborne speeds the Doppler error can be seen but because of the long simulation times and the way the noise slowly increases it is difficult to draw a line. This analysis will be left for the actual hardware.

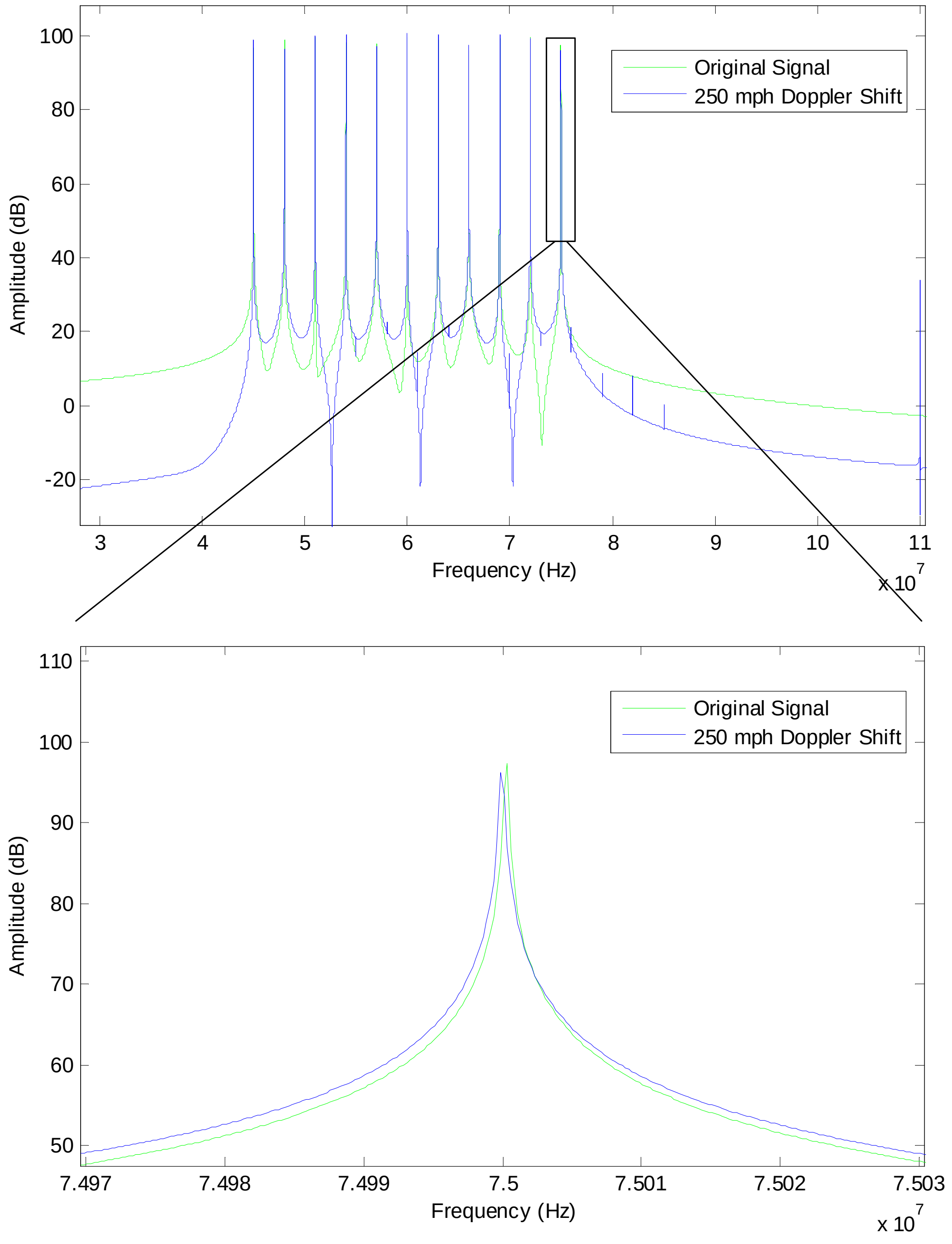


Figure 4.22: 250 miles per hour Doppler effect shown with enlargement.

4.3.1.4 Theoretical Signal Summation Fidelity Experiments

Two different scenarios are run to show the theoretical signal summation fidelity. The possible scenarios for these experiments is infinite. The first was selected to give strong proof for the capability of the DBES to perform these summations and it shows how a weak signal can be lost because of a strong signal. The second was selected because it is a common occurrence in the real world and it shows the greatest theoretical weakness to the signal summation fidelity which is the ability to cancel signals. Other scenarios could certainly be performed; however, the results would be similar in that either they would sum together well with no issues, cancelation would be a problem due to signal interpolation, or a weak signal could not be properly represented in the presence of a large signal.

4.3.1.4.1 General Signal Summation

For this experiment an outlandish scenario was created that would be virtually impossible to recreate in the real-world. In order to recreate this scenario you would need to set up quickly moving mirrors and so this scenario was dubbed the ”House of Mirrors” scenario. The point is to test the capabilities of the DBES by showing that the attenuation and delay parameters for each propagation path can be controlled independently. The scenario produces nine different Doppler shifts and nine different attenuation settings for nine different propagation paths. The Doppler was set to be from negative 400 kHz to positive 400 kHz in 100 kHz steps and the attenuation was set to increase linearly by five decibels for each propagation path. Because Doppler exists, the scenario used many parameter sets changing over time to produce this result which further tested the DBES capabilities. These parameters were given to the model and the results are in Figure 4.23.

The simulation shows that this scenario worked without an issues. A single sinusoid input resulted in nine different sinusoidal outputs simultaneously. All the amplitudes are correct and each of the frequency shifts are accurate as well. This fictional scenario tests a lot of the DBES's capabilities at once including: multipath, signal interpolation, delay interpolation, attenuation, and parameter delivery just to name a few. These results show that the DBES should theoretically be able to handle

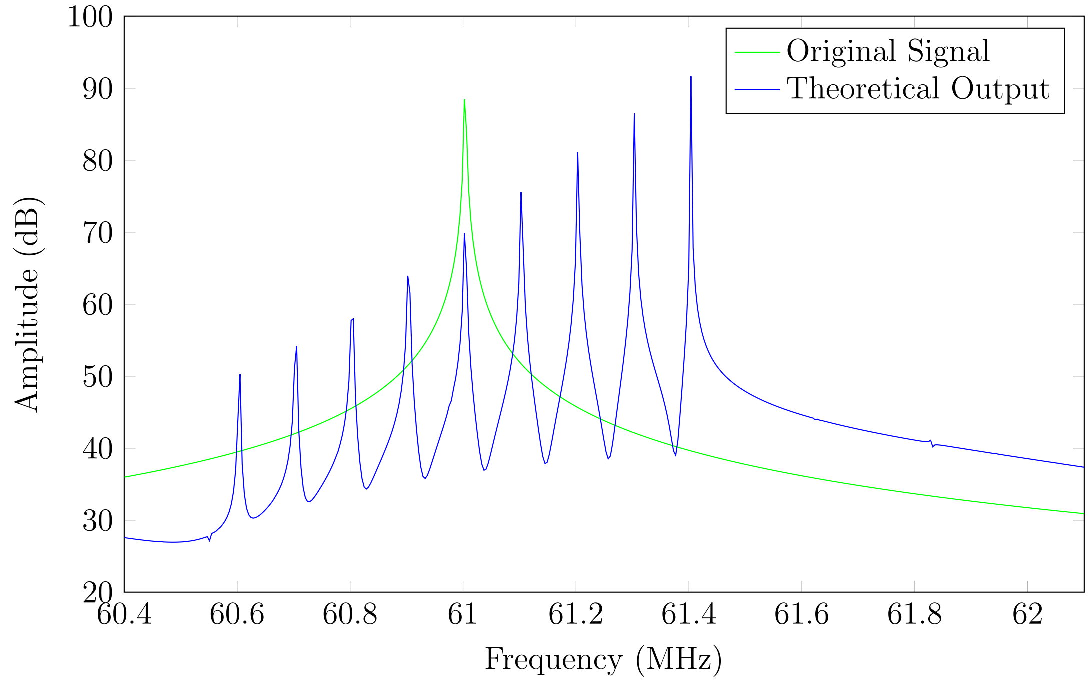


Figure 4.23: Theoretical results from a multipath scenario demonstrating DBES capabilties

all this different functionality well.

These results don't show a known limitation that will be visible in this experimental fidelity experiment. A small signal can easily be digitally lost in the presence of a large signal. The DAC only has 16 bits to represent a signal, so if a small signal is 2^{15} times smaller than a large signal, it cannot be represented. The experimental section will show that this is ratio actually much worse due to noise generated by the AFE and the digital loss of the signal occurs well after the small signal can be buried in noise generated by the large signal.

4.3.1.4.2 Frequency Selective Fading

Frequency selective fading is a phenomenon that occurs when a two or more propagation paths have a delay that results in the paths canceling each other out. For these simulations a delay difference between propagation paths corresponding to one half of the inverse of the cutout frequency is used to create the fading profile. For example: if one propagation path has a delay of 100 ns then to produce frequency cutout at 75 MHz another propagation path should have a delay of 0.5/75MHz+100ns or 106.667 ns. In this scenario when a 75 MHz signal is sent along both propagation paths, the result is one arriving 180 degrees out of phase with the other and thus the signals cancel each other out.

The TDL model used by the DBES can easily replicate these scenarios. The signal path model was used and two propagation path parameters were given with identical attenuation and differing delays as described above. The center frequency of the model was set at 70 MHz and the model had a bandwidth of 30 MHz. The input signal used was a sum of sinusoids separated by 100 KHz across the entire bandwidth of the input. This was done to obtain good resolution of the fading profile. The simulation was run six times with a different cutout frequency for each simulation. For each run three signals are plotted: the original signal input to the DBES, the output of the DBES, and a signal representing the ideal fading. The ideal fading signal was obtained by summing the original signal with the same signal delayed by the amount prescribed by each fading cutout frequency. This is another reason a sum of sinusoids signal was used, it was easy to mathematically add the delay to the time of each sinusoid so that no interpolation was required. In addition after the fourier transform was performed, the data was reduced by taking the maximum for each 130 kHz interval. This effectively takes the peak of each sinusoid, smooths the line, and reduces the clutter in the plot so that the shape of the curve can be seen easily. Figure 4.24 shows these results.

These results show that while fading does occur, it is not quite as good as the ideal scenario. There can be as much as ten to twenty dB inaccuracy in the result at a given frequency. Note that cutout frequencies close to the center frequency are typically very close to the ideal and the farther away the cutout frequency is, the farther from ideal the result becomes. This is clearly an issue with the signal interpolation error.

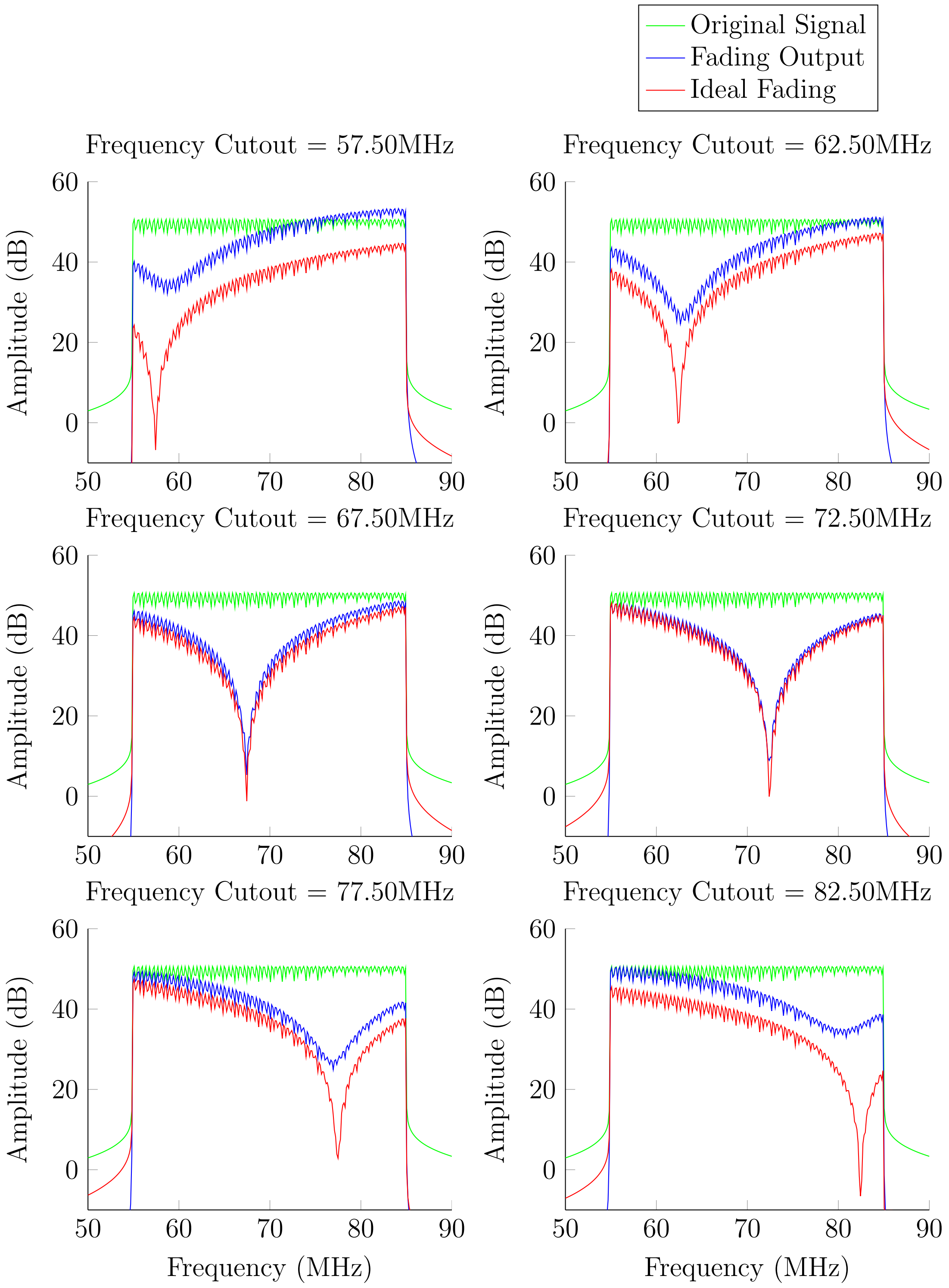


Figure 4.24: Theoretical frequency selective fading at multiple frequencies

Since the two signals have different error added into them because of the different amount of interpolation done, they do not cancel ideally when summed together. Improving this result is possible by raising the sampling rate of the signal by digital upsampling prior to the signal path, or by introducing a higher order interpolation scheme.

Another interesting note is that the signals are not canceled at baseband. By simply looking at the I and Q signals before mixing to the RF signal, one would have no way of knowing if frequency fading would occur or not. This means that if fading is to occur in the actual DBES, the actual signal power cancelation will happen in the analog domain inside the analog front end.

A final result that further confirms the signal interpolation issue with this scenario, is that the quality of the cancelation depends on the location of the initial delay value. Depending on how much signal interpolation is done with the initial delay value, different results are produced. To show this the same model was used, but instead of changing the cutout frequency, the initial delay was changed. The reported initial delay is not in seconds but rather in samples: the actual delay divided by the sample rate. This is done because an integer number results in no interpolation of that propagation path. The delay was increased by 0.2 samples from run to run and the results are shown in Figure 4.25.

The fading profile changes with each graph and ideally, they should be the same. This shows that depending on how the error in signal interpolation between the two propagation paths line up, it is possible to get more or less accurate fading profile.

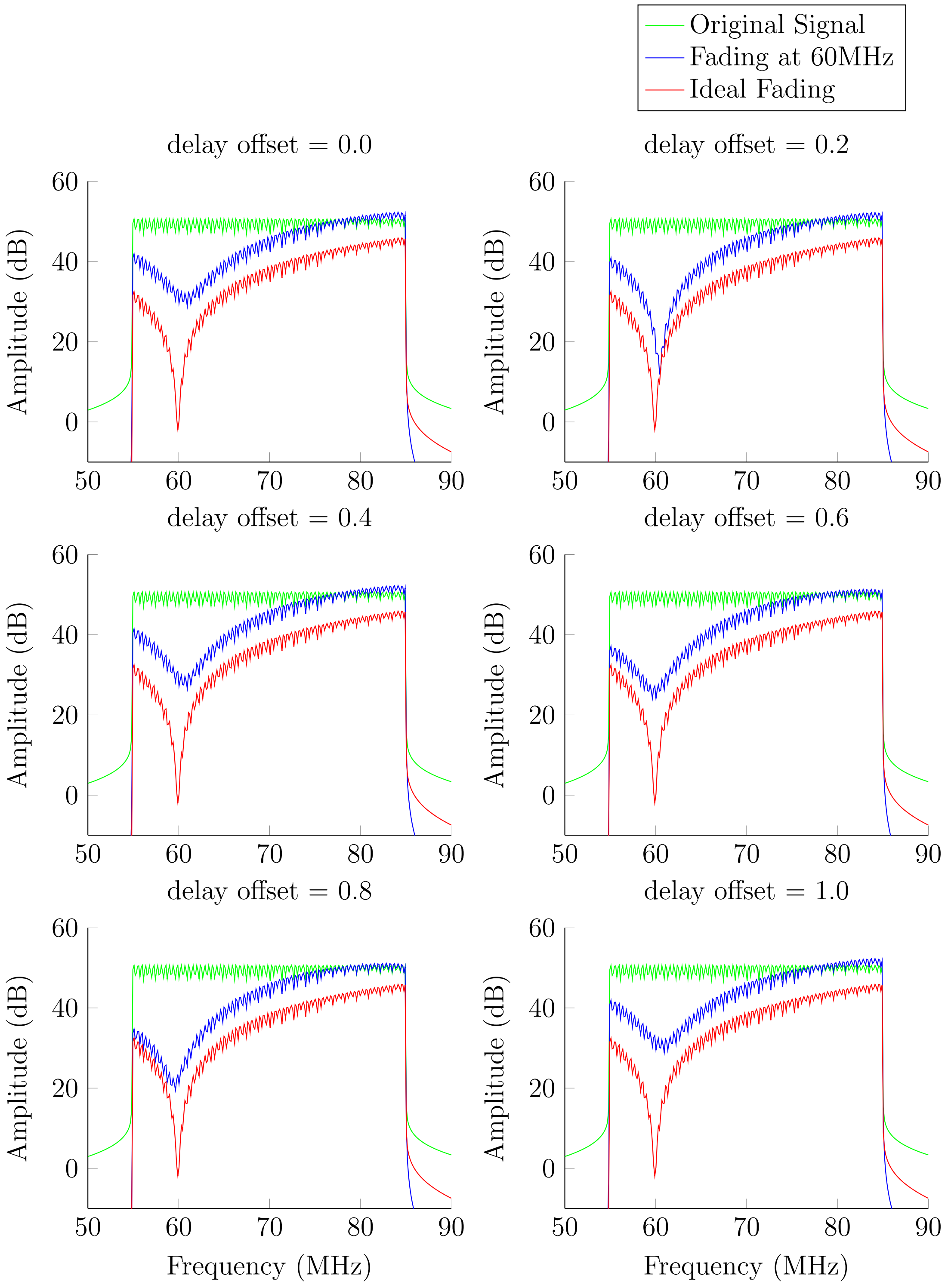


Figure 4.25: Theoretical frequency selective fading at 60MHz with different delay offset parameters measured in sample periods

4.3.2 Experimental Fidelity Analysis

This section experimentally confirms the theoretical analysis of the previous section. The goal of this section is to analyze the capabilities of the DBES and determine their fidelity in terms of accuracy and noise. To achieve this, these studies exclude the use of radios and instead use a signal generator to produce an RF signal at the input of the DBES and a spectrum analyzer to view the RF output. It is impossible to cover all the scenarios in which the DBES would be expected to perform; however, the primary functions of the DBES are covered here. As in the theoretical study, the experimental study is grouped into three sections: signal power fidelity, signal delay fidelity, and signal summation fidelity.

The signal power section analyzes the noise and accuracy of the output power of the DBES. There two primary factors that contribute to this fidelity metric: spurious noise produced by the hardware and the dynamic range of the output power.

The signal delay section analyzes the ability of the DBES to implement a specified signal delay and a specified change in delay. Contributors to the signal delay fidelity include the minimum and maximum propagation delays of the system, the signal interpolation noise, and the Doppler effect noise.

The summation of propagation paths section investigates the ability of the DBES to combine multiple signals. This combination of paths is what creates many real world effects generically referred to as multipath effects. The two primary contributors of noise in this section is the hardware limitation of the DAC and the accuracy of canceling two signals.

For all of these experiments the DBES was configured as a two-node system as shown in Figure 4.26. This two node setup used exact same FPGA images that allowed for the 32-node system; however, only three FPGA nodes were used: two radio nodes and a compute node that connected them. In this manner, it was expected that any issues with the signal path accuracy in the large system would be seen on this system, with the exception of issues in summing multiple signal paths. As the figure shows, an Agilent E4438C signal generator and a Tektronix SA2600 spectrum analyzer were used to produce and view the signals. The signal was input and output through the TRCal boards to ensure these boards did not produce any unwanted effects.

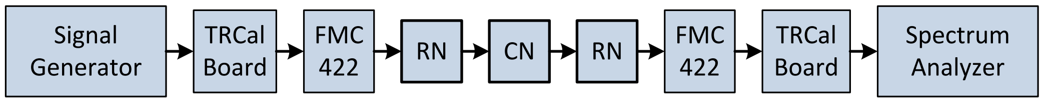


Figure 4.26: Two node setup used in fidelity experiments

4.3.2.1 Experimental Signal Power Fidelity Studies

The signal power experiments run included the analog front end noise analysis and the attenuation results. From the theoretical study, it is not expected that any of the noise due to quantization should be visible here and it is also expected that there will not be any detectable error in the attenuation of the signal aside from the desired effect of the increasing signal to noise ratio as the signal becomes very small.

4.3.2.1.1 Analog Front End Noise Analysis

The analog front end in most scenarios is the main contributor of noise in the actual system. There are three significant noise sources: the Local Oscillator (LO) feedthrough, the sideband rejection, and spurious noise. The LO feedthrough exists because of a DC current imbalance between the baseband analog I and Q signals prior to mixing them with the local oscillator to the RF frequency. When the I and Q signals are mixed with the LO and summed, the mismatch in current results in imperfect rejection of the local oscillator signal and therefore produces a narrowband signal at the LO frequency. The chip that performs this mixing calibrates the currents to match, but it is impossible to match exactly.

The second major source of noise is due to imperfect sideband rejection. When two signals are multiplied together two images are produced, one at the sum and the other at the difference of the two signals frequencies. A direct conversion system such as the one used in the FMC422 uses the in-phase and quadrature parts of a signal to reject the difference of the two signals. In an actual system this rejection is not perfect and is generally measured as the difference between the signal of interest and the sideband signal. Again, steps are done on the FMC422 board to calibrate this noise to be as low as possible.

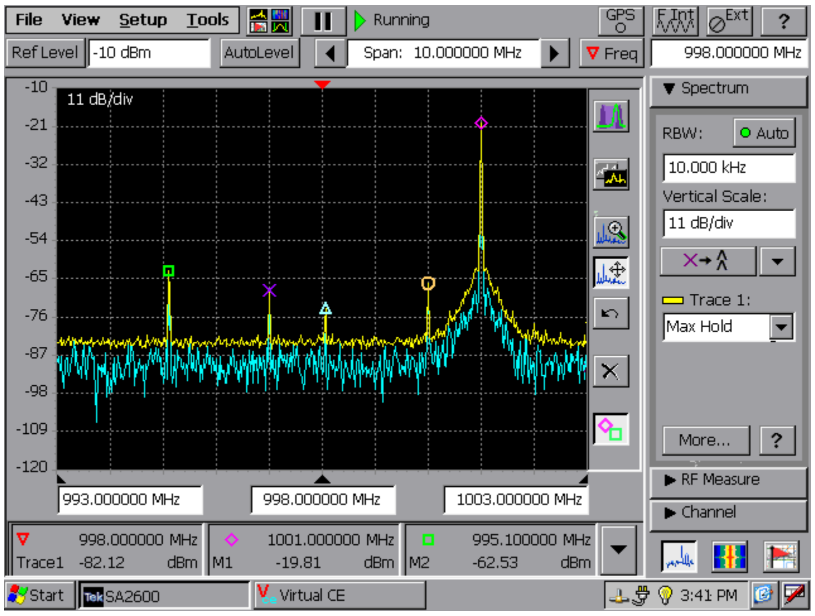
The spurious noise occurs because of our setup. The receive LO and transmit LO are not the same frequency. They are set one megahertz apart. This is done because of hardware in the Lime microsystems chip. The transmit and receive local oscillators are generated independently and as a result, are not the exact same in phase and frequency. It was observed that different noise was seen surrounding the signal of interest that changed from resetting the boards. This noise was extremely large in some cases which made it necessary to avoid this situation. The signal is brought in at one MHz lower frequency than it is output, and is digitally shifted to account for this difference. This results in a net equal frequency change and removes the large noise that was not reported here. The problem is that it introduces spurious noise. A digital LO feedthrough and sideband rejection occurs as well as the analog one described above. These are generally very small for a radio receiver and are ignored in normal radio cases. In the DBES case however, the received signal is fed back to the output and it becomes significant. The LO and sideband signals are mixed and aliased on the output with the transmit LO and the signal of interest. This shows up as spurious noise throughout the band of interest. Whereas the sideband frequency is easy to predict, this noise is not due to the mixing and aliasing nature. If you take the sum and difference of each of the noise sources and the signal of interest you will find these locations. The good news is that this noise is generally the smallest of the three presented here.

Presented in Figure 4.27 is a typical noise after the AFE has been calibrated for our system when outputting a single frequency. Better and worse calibration results have been observed. There are a few tricks that can be implemented to improve the calibration but those steps are not going to be described in detail here because they are specific to the RF frequency used and would only improve the noise in special circumstances. This result was obtained without any special calibration techniques; the calibration routine was run on the AFE as delivered by the manufacturer. Marker one, the light purple diamond, shows the signal of interest. Marker two, the green square, shows the sideband noise measurement which here is 42 dB below the signal of interest. Marker three, the light blue triangle, is the LO feedthrough. Markers four and five, the purple cross and orange circle, are the spurious noise due to the LO and signal mixing. All the noise here with the exception of the LO feedthrough all decreases when the signal of interest decreases. This means that typical signal to noise performance of 40 dBc should be expected. The LO feedthrough noise does not decrease with decreasing signal and can be an issue if there is not enough attenuation between the radio and the DBES on the transmit side. Once the TGC is implemented, this too will decrease with increased attenuation which will improve the SNR at low signal levels.

4.3.2.1.2 Attenuation Results

The integrity of the signal output is a primary concern for the fidelity of the system. The challenge is that the power output of the system is over a very large range. In the real world, a good radio can pick out a signal as low as -115 dBm which is also a typical noise floor. In addition, received signals can be higher than -30 dBm depending on the output power of the transmitting radio. It is desirable for the DBES to be able to recreate signals in this range. This experiment is designed to show this capability.

A scenario was run that linearly changes the Decibel attenuation parameter from the maximum value to the minimum value and back over the course of thirty seconds.



Ref Level -10 dBm

Span: 10.000000 MHz

998.000000 MHz

11 dB/div

RBW: Auto

10.000 kHz

Vertical Scale:

11 dB/div

Trace 1:

Max Hold

993.000000 MHz 998.000000 MHz 1003.000000 MHz

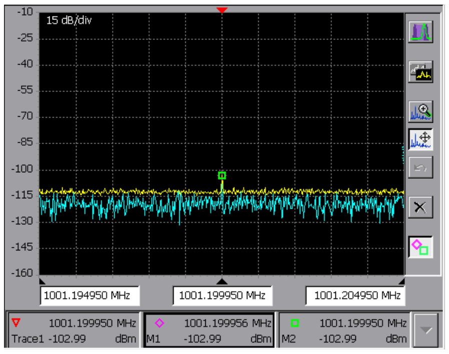
Trace1 998.000000 MHz -82.12 dBm

M1 1001.000000 MHz -19.81 dBm

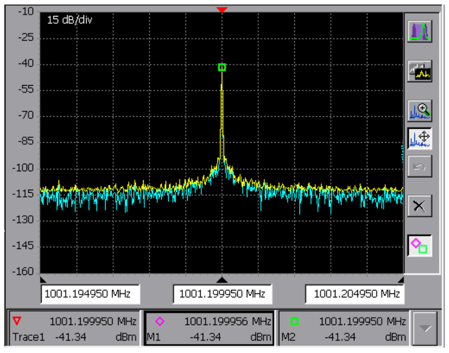
M2 995.100000 MHz -62.53 dBm

Figure 4.27: Typical AFE noise showing local oscillator, sideband, and spurious noise sources

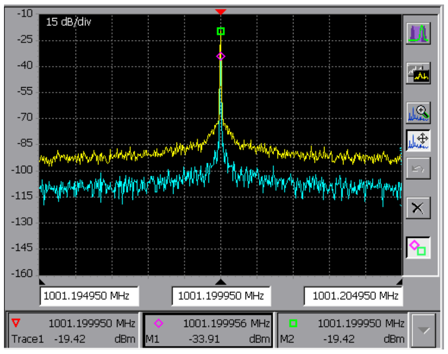
Note that this produces an exponential change in the power of the signal over time. The ideal result is that the signal appears from below the noise floor at some point, rises to a peak and then falls back below the noise floor without any extraneous noise. As described by the AFE noise section, this will not happen as the sideband and mixed sideband signals will grow and shrink with the signal of interest. An additional issue is that the local oscillator noise will be present at the same level throughout the experiment which could be a problem depending on the scenario. As described in the AFE section these issues cannot be easily dealt with so the actual signal is focused on during this experiment. The results of the scenario are shown in Figure 4.28.


15 dB/div
-10
-25
-40
-55
-70
-85
-100
-115
-130
-145
-160
1001.194950 MHz
1001.199950 MHz
1001.204950 MHz
Trace1 1001.199950 MHz -102.99 dBm
M1 1001.199956 MHz -102.99 dBm
M2 1001.199950 MHz -102.99 dBm

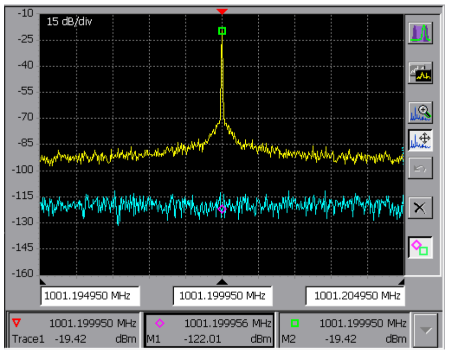
(a) The signal is just appearing above the noise floor



(b) The Decibel level of the signal linearly grows with respect to time



(c) The maximum signal is shown along with the now decreasing signal


15 dB/div
-10
-25
-40
-55
-70
-85
-100
-115
-130
-145
-160
1001.194950 MHz
1001.199950 MHz
1001.204950 MHz
Trace1 1001.199950 MHz -19.42 dBm
M1 1001.199956 MHz -122.01 dBm
M2 1001.199950 MHz -19.42 dBm

(d) The maximum signal and the noise floor after the scenario concludes

Figure 4.28: Snapshots from the experimental attenuation results showing more than 90dB of dynamic range

These results show that the signal does indeed rise from the noise floor, peaks, and then decreases below the noise floor. This result shows that the DBES is capable of a dynamic range of greater than 90 dB as measured by the difference between the maximum signal shown and the minimum signal that can be seen. A very small bandwidth is shown so that the noise floor of the spectrum analyzer is as low as possible. It should also be noted that while a single signal can be produced at that range, inside the DBES there is only one bit representing that signal. This means that while the signal can be represented, it may not carry enough information for a radio to decode. This is radio dependant and to answer this question one would need to acquire the radio in question and test it to find out.

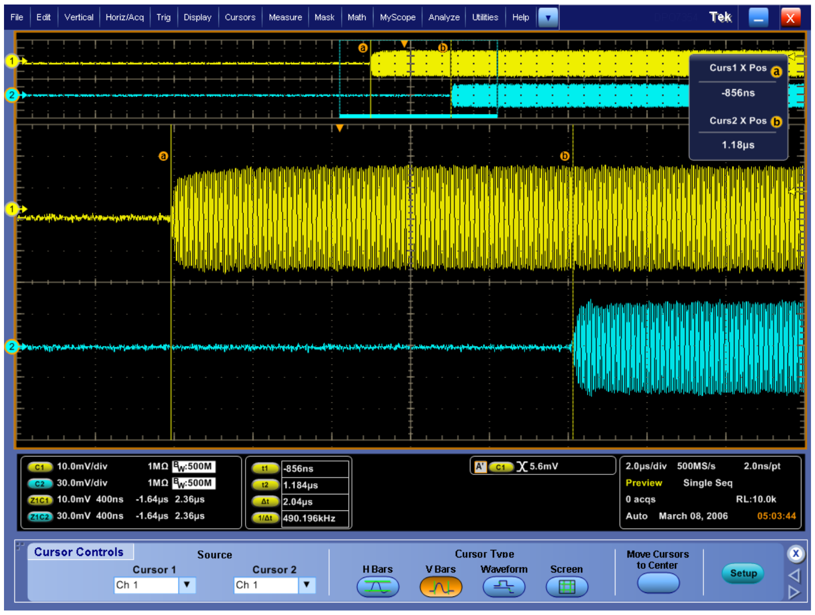
4.3.2.2 Experimental Signal Delay Fidelity Experiments

The signal delay fidelity experiments measure the ability of the DBES to accurately delay signals and the noise inducing this delay causes. These experiments include the propagation delay, static signal delay, and the Doppler shift experiments. In the theoretical section, there was the signal path and doppler interpolation error analysis.

4.3.2.2.1 Propagation Delay Measurements

The propagation delay of the system is important because it is typically a constant source of inaccuracy. Scenarios can be developed such that each radio stays a specified distance away from every other radio and then the delay through the system can be negated; however, this will not typically be possible. Ideally, this measurement will be zero; however, this is impossible.

The setup for this scenario is similar to the one described at the beginning of the experimental fidelity study; however, an oscilloscope (Tektronics DPO7354) is used in place of a spectrum analyzer, and the input signal to the DBES is split with an RF splitter and also input into the oscilloscope. The output of the DBES at the TRCal board was connected to another channel on the oscilloscope so that the difference between the input and output could be seen. The challenge with this measurement is to get in input signal to ramp up to full scale quickly. If a signal generator turns on it can take 10's or 100's of microseconds to ramp up to full scale which would make this measurement impossible. Ultimately, a spectrum analyzer that allowed for amplitude modulation was used. The modulation frequency was set as low as possible (around one kilo-Hertz) and the amplitude change was set from zero to full. With a carrier frequency of 105 MHz, this produced the desired rapidly increasing signal necessary. Figure 4.29 shows a screen capture of the oscilloscope that was set for a single trigger at a threshold above the noise floor.



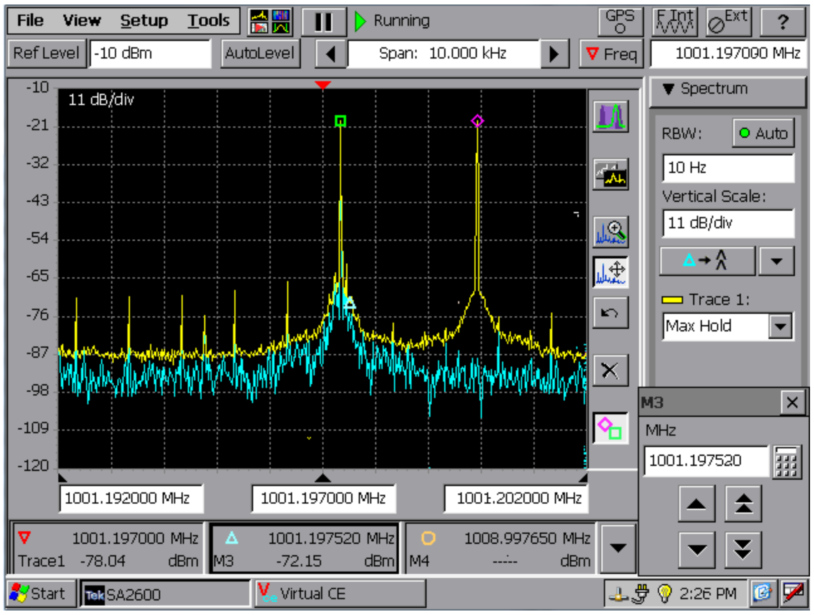
```
Curs1 X Pos  -856ns
Curs2 X Pos  1.18µs

C1    10.0mV/div        1MΩ  BW:500M
C2    30.0mV/div        1MΩ  BW:500M
Z1C1  10.0mV  400ns  -1.64µs  2.36µs
Z1C2  30.0mV  400ns  -1.64µs  2.36µs

t1    -856ns
t2    1.184µs
Δt    2.04µs
1/Δt  490.196kHz

A' C1 ╳ 5.6mV

2.0µs/div   500MS/s        2.0ns/pt
Preview     Single Seq
0 acqs                     RL:10.0k
Auto   March 08, 2006      05:03:44

Cursor 1  Ch 1
Cursor 2  Ch 1
```

Figure 4.29: Measurement of the propagation delay through the system

This shows that the entire system has approximately a 2.04 microsecond delay. Other carrier frequencies were used in each tuning band of the FMC422 and in all cases the result was the same amount of delay. The speed of light can travel approximately 600 meters in this time. This 600 meters therefore corresponds to the DBES minimum distance because if it takes two microseconds for a signal to propagate through the DBES then it cannot emulate a delay of less than two microseconds. It can emulate two delays that are only a nanosecond apart as shown in the next paragraph, but the absolute delay of each must be greater than two microseconds. Fortunately, most field radios are designed to work at much longer distances than that and a two microsecond delay is not an issue. However, this could be a problem for other wireless devices such as IEEE 802.11 (Wi-Fi) and IEEE 802.15 (bluetooth) standard waveforms which are designed to operate over much shorter distances.

4.3.2.2.2 Doppler Shift Results

The Doppler results presented here correspond the the theoretical predictions in the signal interpolation error sections 4.3.1.3.2 and the Doppler error and noise



Ref Level -10 dBm
AutoLevel
Span: 10.000 kHz
Freq 1001.197000 MHz
RBW: Auto
10 Hz
Vertical Scale:
11 dB/div
Trace 1:
Max Hold
M3
MHz
1001.197520
1001.192000 MHz
1001.197000 MHz
1001.202000 MHz

```
1001.197000 MHz     1001.197520 MHz     1008.997650 MHz
Trace1  -78.04  dBm M3  -72.15  dBm     M4  -.--  dBm
```

Figure 4.30: Experimental Doppler results showing good shift of approximately 2.56 kHz corresponding to a closing speed of approximately 1600 mph

section 4.3.1.3.3. The DBES was setup as a two node as shown in Figure 4.26 with a signal generator and spectrum analyzer. Matlab was used to generate a single propagation path with a changing delay over several seconds so that a Doppler Shift is seen. Figure 4.30 shows a large doppler shift of a signal at 1001.2 MHz corresponding to a closing velocity of approximately 1700 miles per hour. The DBES produces the correct Doppler shift of approximately 2.56 kHz. The center frequency of the DBES was at one gigahertz. Notice that some noise is seen but it is very low because the signal of interest is close to the center frequency. This is exactly as predicted in the theoretical section.

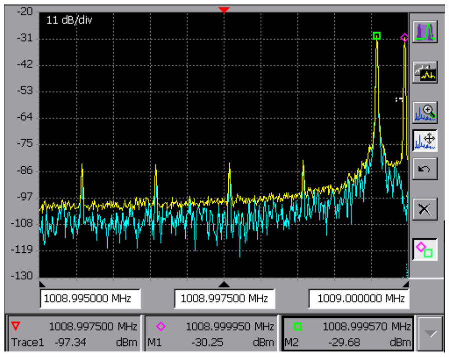
The next set of graphs shown in Figure 4.31 displays results from several tests run with different closing velocity speeds. For each of these graphs the DBES was loaded with a different set of parameters that corresponds to the desired closing velocity. With the low velocity shifts of less than 250 mph no noise is seen on the spectrum analyzer. At 500 mph noise can be seen but it is difficult to pick out but starting at 1000 mph and higher, the noise becomes very apparent on the spectrum analyzer.

This noise is different than anticipated by the theoretical analysis. Based on repeated simulations the notion was that the noise spurs would increase in amplitude as the closing velocity increased. This was an incorrect assumption as shown by this figure. It turns out that the spread of the spurs increases with closing velocity, but the amplitude stays the same. This was missed by the simulations, probably because of the inaccuracies in the digital fourier transform and the necessity for very large time series data to decern these spurs. This result confirms that at normal terrestrial speeds less than 200 mph the Doppler effect should not produce significant error as even though these spurs would still exist, they would be less than 10 Hz away from the correct signal which is not likely to cause issues in the radios. It also confirms the signal interpolation is an issue and further investigation should be performed to improve it.

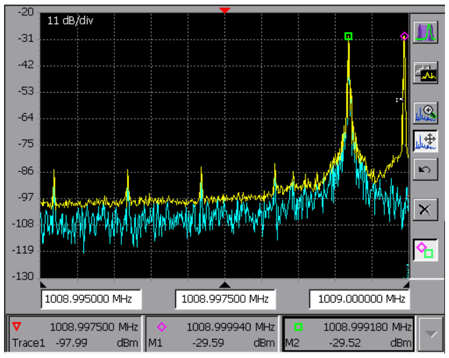
4.3.2.2.3 Static Signal Delay

The static signal delay determines the ability of the DBES to accurately delay a signal by a fixed amount. As shown in the propagation delay results, the delay must be greater than approximately 2 microseconds but after that point the question is: how accurate of a delay can be produced and how much noise does it add to the result? The accuracy of the delay can be answered by simply looking at the digital capabilities of the DBES. The delay parameter has thirteen bits to the left of the decimal and nineteen bits to the right of the decimal. The parameter is measured in sample periods so with a sample rate of 41.96 MHz the static resolution is approximately 45 femtoseconds. In fact when interpolating the system uses 24 bits, resulting in a dynamic delay resolution of approximately 1.4 femtoseconds. The static resolution results in an positional accuracy greater than 20 microns which at first sounds more accurate than needed, but as shown in the frequency fading section, it is necessary to achieve appropriate nulling of high-frequency signals.

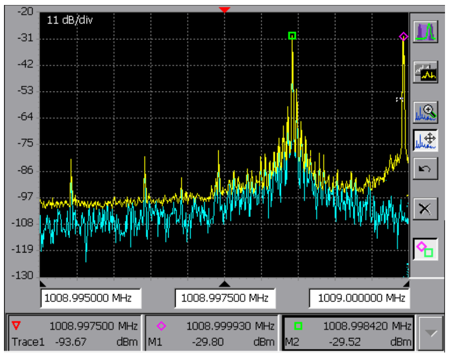
In order to achieve this resolution, the signal is interpolated between sample



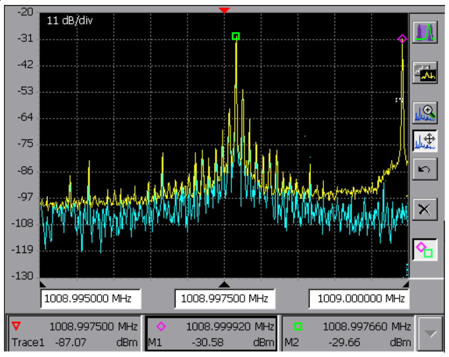
(a) Velocity of 250 mph, shift of 376 Hz



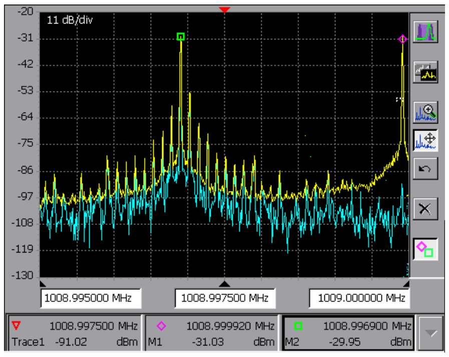
(b) Velocity of 500 mph, shift of 752 Hz



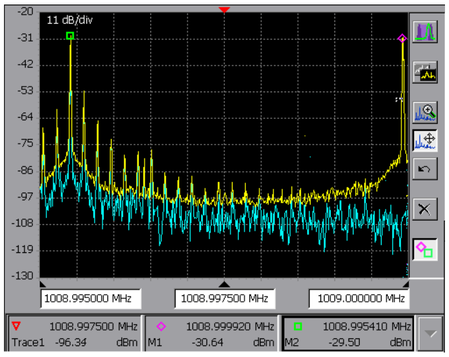
(c) Velocity of 1000 mph, shift of 1.50 kHz



(d) Velocity of 1500 mph, shift of 2.26 kHz



(e) Velocity of 2000 mph, shift of 3.01 kHz



(f) Velocity of 3000 mph, shift of 4.51 kHz

Figure 4.31: Experimental Doppler results showing noise from signal interpolation.

points. This means that there is inherent error in the signal produced by this delay. The theoretical doppler shift result shows that the interpolation produces noise with a change in delay, but does the signal interpolation produce noise with a static delay? The worse case scenario for the noise would be when the delay parameter is 0.5, meaning the interpolated result is mid way between two samples. This delay parameter was put into the DBES and a static sinusoid was put through the system. No noise was seen on the output as compared to a signal with zero delay and no interpolation. If the signal was not static then it may be possible to see noise as predicted in the theoretical signal interpolation section, but it was not possible to verify this experimentally.

4.3.2.3 Experimental Signal Summation Experiments

In this section there are two experiments: the general summation of propagation paths and the frequency selective fading experiments. Both these experiments are continuations from the theoretical signal summation fidelity tests and use the actual DBES to verify the theoretical predictions determined by those tests.

4.3.2.3.1 General Summation of Propagation Paths Evaluation

This is the experimental study of the same outlandish scenario that was created for the theoretical section: the "House of Mirrors" scenario. Again, the point is to test the capabilities of the DBES by showing that the attenuation and delay parameters for each propagation path can be controlled independently. A single sinusoid input is turned into nine sinusoid outputs but manipulating nine different propagation paths. The Doppler shift was set to be from negative 4 kHz to positive 4 kHz in one kilohertz steps. The attenuation was set to be from 50 dB to 10 dB in 5 dB steps. The results are shown in Figure 4.32.

The simulation shows that this scenario worked and produced accurate Doppler and attenuation; however, there was a significant noise issue. There is a spur located approximately one kilohertz above the largest shift that is larger than the smallest

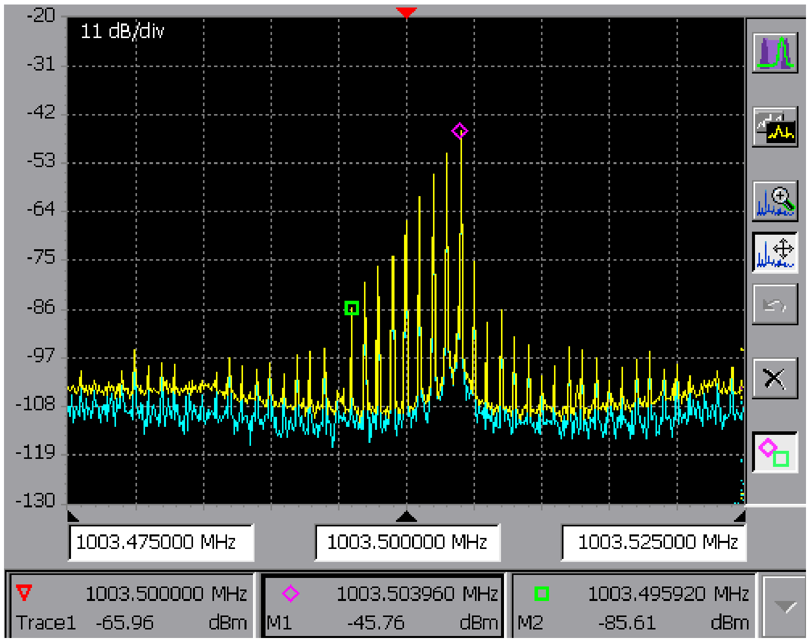


Figure 4.32: Experimental signal summation results

sinusoid output. This highlights a weakness of the DBES: generating very large signals and very small signals simultaneously is very challenging. As shown in the AFE noise section, the spurious noise is typically 40 dB below the largest signal. This means that if a signal of interest is 40 dB below a second signal of interest, the first signal of interest will be at the noise level generated by the second signal of interest. There is little that can be done about this, short of replacing the AFE with another, more expensive AFE. This is one of the tradeoffs of using a direct conversion methodology. Each direct conversion AFE cost approximately 15 times less than an intermediate frequency system, but the noise performance in band is not as good as an IF system.

4.3.2.3.2 Frequency Selective Fading Results

The frequency selective fading results from the theoretical section showed a great deal of inaccuracy in the frequency location and the amplitude of the fade. It was unsure if this behavior would be seen, or if any nulling would be seen. As mentioned in the theoretical fading section, the amplitude and the notch of the fade does not occur at baseband, but rather when the I and Q signals are mixed together with

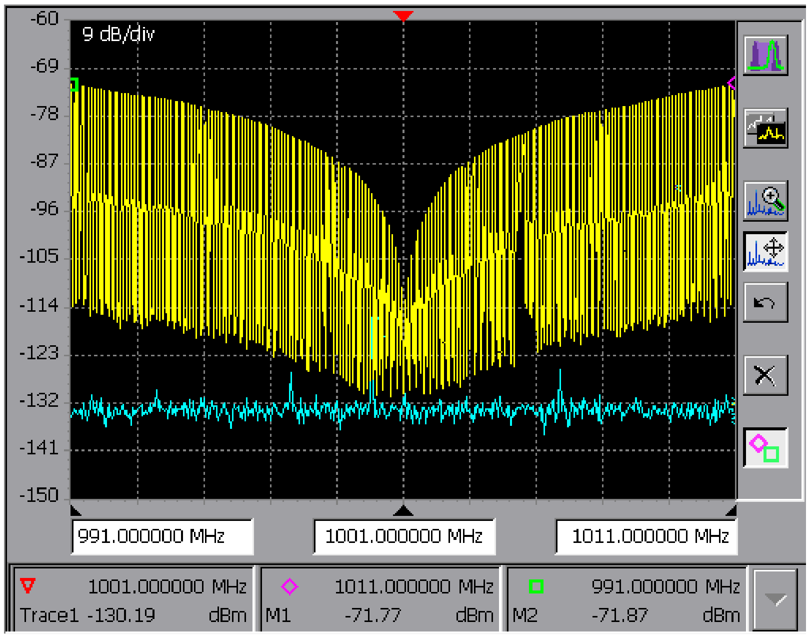
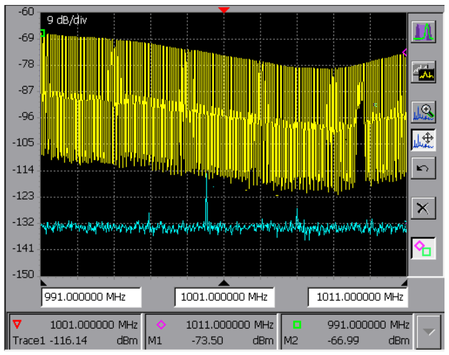


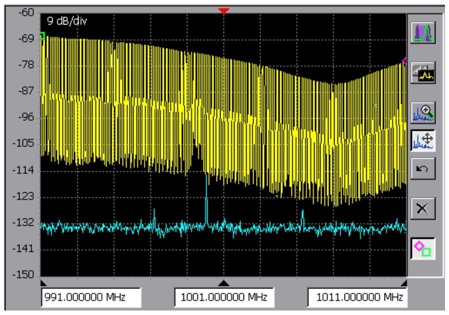
Figure 4.33: Fading at 1001 MHz with a center frequency at 1000 MHz

the LO. While the mathematics showed that this should occur, the concern was that there were real world ”oddities” that could keep this from happening. The theoretical section result was that the farther away from the RF center frequency, the worse the null. Therefore, a null of 1001 Mhz was used with an RF center frequency of 1000 MHz to attempt to attain a good fade. Once the DBES was running with the fading parameters in the system, the signal generator was stepped through several different frequencies across a 20 MHz band. The maximum hold on the spectrum analyzer held the output of the DBES for each frequency which makes it possible to see a broad spectrum result in one picture. To reiterate: this is not a single output of the DBES, it is the maximum response to several different sinusoid inputs. The results shown in Figure 4.33 are excellent. The fade happens at exactly 1001 MHz and the null is quite deep. This result means that it is possible for fading to occur and that it can be a very good fade. The problem is that this is only one result and is close to the best that the DBES can achieve at this time.

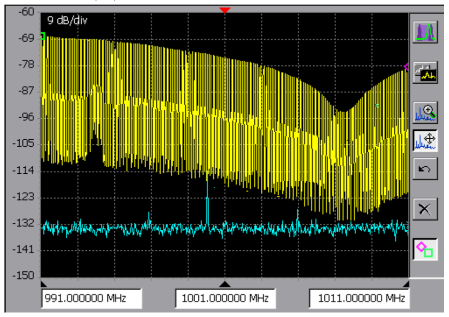
The theoretical fading section found that the delay offset, a parameter that has no real world meaning, would affect the quality of the fade. The delay offset is the amount of delay the first propagation path has. The second propagation path is added to null the first, and therefore has a delay slightly more than the first path. Ideally, it should not matter what the delay offset is, the result should be the same. It was seen in the theoretical study that this was not the case. The next test was done to verify if the theoretical study was accurate in it's predictions. Figure 4.34 shows a fade at 1009 MHz with a center frequency of 1000 MHz with each graph showing a different delay offset. These results show a fade is not accurate in the location or the depth and one that moves around under conditions that it should remain fixed, just as the theoretical results predicted. It is good that the experimental results matched the theoretical, but unfortunately the results are not as good as one may hope. As found in the theoretical section, this result is due to the signal interpolation accuracy. This is another case of signal interpolation decreasing the fidelity of the emulation and again, increased sample frequency or improved interpolation scheme would improve these results.



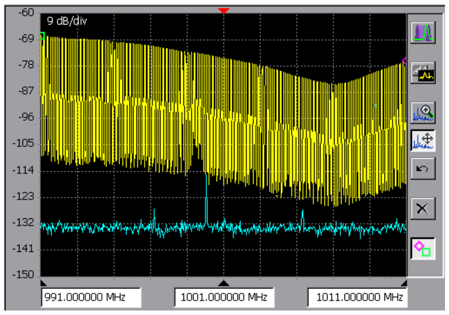
(a) Delay offset = 0.0



(b) Delay offset = 0.2



(c) Delay offset = 0.4



(d) Delay offset = 0.6

Figure 4.34: Frequency selective fading seen at 1009 MHz with different delay offsets. Delay offsets presented are in units of sample periods. The sample frequency used was 41.96 MHz resulting in a sample period of approximately 23.83 nanoseconds.

4.3.3 Observational Fidelity Analysis

The observational study consists of several scenarios run for different numbers of radios. The parameters for these scenarios were created by custom Matlab scripts and by REMCOM's Wireless In-Site. The experimental and theoretical fidelity studies prove that the DBES works as it was intended. From that point of view, this section is not necessary to confirm the fidelity of the system; however, the DBES was intended to be used with real wireless devices and up to this point in this dissertation, none have been connected to the DBES. From an academic point of view, this section is observational science as it has not been possible to construct a control for all but one of these experiments. In observational science the goal is to simply observe behavior and then draw conclusions or more questions from the results. From a practical point of view, the purpose of this fidelity study is to gain confidence in the capabilities of the DBES without the need to understand the complexities of the experimental and theoretical studies. A potential user of the system does not want to hear about signal power fidelity or any of the other complicated theoretical and experimental studies that were performed. Many users just want to know if they can connect their radios to the DBES and have it work. The goal of this section is to conduct observational science to improve a layperson's confidence of the system and examine DBES performance with radios.

This section consists of three scenarios with two radios and one scenario with five radios. For the two node scenarios, the DBES was set up just as in the experimental section with a compute node, two radio nodes, and two TRCal boards. For the five node scenario, five radio nodes with their accompanying analog hardware were all connected to a single compute node. The entire use case (Section 3.6) describes how the DBES was set up for each of these scenarios. The major difference between the signal generator tests and using actual radios was getting the correct power levels into the DBES and ensuring the radios were isolated from each other as described in Section 3.6.4.

Even though this is largely observational science, steps were taken to be able to qualify the outcome with the two node scenarios. The first scenario is an attenuation scenario where a control was used to compare radio behavior with and without the DBES. In the second and third scenarios, a signal generator and spectrum analyzer was used to view the DBES behavior without radios. This allows predictions to be made about the radio behavior and gives confidence further confidence in the observed behavior of the DBES with the radios. The final scenario with five nodes is strictly observational. The radios were connected, the DBES was run, and results were gathered. However, the scenario was run three times to see discrepancies between runs.

It is important to the developer and owner of the ad-hoc networking radios used in these experiments that specific metrics that could allow someone to determine performance characteristics of the radio are not revealed. Several steps are taken to ensure this does not happen including not reporting the name of the actual radio used and obfuscating reported data received from the radio. For these tests the Signal Power (SP) and Signal Quality (SQ) metrics are gathered from the radios. The radios report signal statistics on who they are connected to and how good the link between the radios is every five seconds. The original radio statistics reported are altered by a linear transformation to obtain the reported SP and SQ metrics. In addition, some tests also have data pushed through the radio. The amount of data transmitted and received will be reported in packets. The size of each packet in bytes will not be reported. The size of each packet will remain constant within a single experiment; however, the size of the packet could change from experiment to experiment.

At the time of this dissertation the phase shift has not been implemented as it's necessity was not realized until late in the development cycle. The results for the tests do not include a phase shift and this could certainly impact the fidelity of the scenario. The phase shift implementation is determined and will be added as soon as time permits.

4.3.3.1 Radio Attenuation Results

While tests in Section 4.3.2.1.2 were performed to ensure that the DBES handles attenuation properly, it is important to make sure the DBES affects radios in the same manner as a simple attenuator would and that the noise generated by the DBES would not affect the radio. This is the only scenario experiment that has a control and is not strictly observational. The radios were connected with a variable attenuator and the SP and SQ of each radio were recorded. The DBES was then connected in the same manner and the attenuation parameters were digitally changed inside the DBES. If the DBES matches the radio behavior seen with an attenuator, then the test will be considered a success. The results are shown in Table 4.4. Note that the exact

Table 4.4: Attenuation results with radios

Prog	Radio 1		Radio 2		DBES	Radio 1		Radio 2	
Attn	SP	SQ	SP	SQ	Attn	SP	SQ	SP	SQ
0	94	9.5	94	9.5					
5	89	9.7	89	9.7					
10	84	9.7	84	9.7					
15	80	9.7	80	9.7					
20	75	9.0	76	9.0					
25	71	8.8	71	8.8					
30	65	8.0	66	8.0					
35	60	7.8	61	7.8					
40	54	7.8	55	7.8	0	56	7.7	56	7.7
45	52	7.8	50	7.8	5	51	7.7	51	7.7
50	47	7.8	47	7.8	10	48	7.7	46	7.7
55	42	7.8	41	7.8	15	44	7.7	42	7.7
60	37	7.7	37	7.8	20	38	7.7	36	7.7
65	33	7.7	32	7.8	25	34	7.7	31	7.7
70	28	7.7	27	7.8	30	29	7.7	27	7.7
75	22	7.7	22	7.8	35	23	7.7	21	7.7
80	18	7.7	17	7.8	40	19	7.0	17	7.2
85	13	4.5	13	6.0	45	14	5.0	12	3.5
90	8	2.3	9	2.5	50	9	2.2	9	2.2
95	4	4.2	3	out	55	5	2.2	5	2.2
100	out	out	out	out	60	out	out	out	out

amount of attenuation added is not reported. There was a fixed amount of attenuation between the radios for both tests that was added to the attenuation reported here. Also, for this test the DBES added about 40 dB of attenuation through the system with no digital attenuation. This corresponds to the maximum signal that the DBES could output and is why the SP and SQ metrics for a DBES attenuation of 0 dB match the metrics for the attenuator set to 40 dB. As mentioned in Section 3.6.4 the isolation was an issue for the DBES. Because of the isolation issue, large attenuators have to be placed immediately on the output of the radios. This attenuates both the transmitted and received signal. The output power of the AFE could only be so large, which therefore limits the maximum signal strength received at the radio.

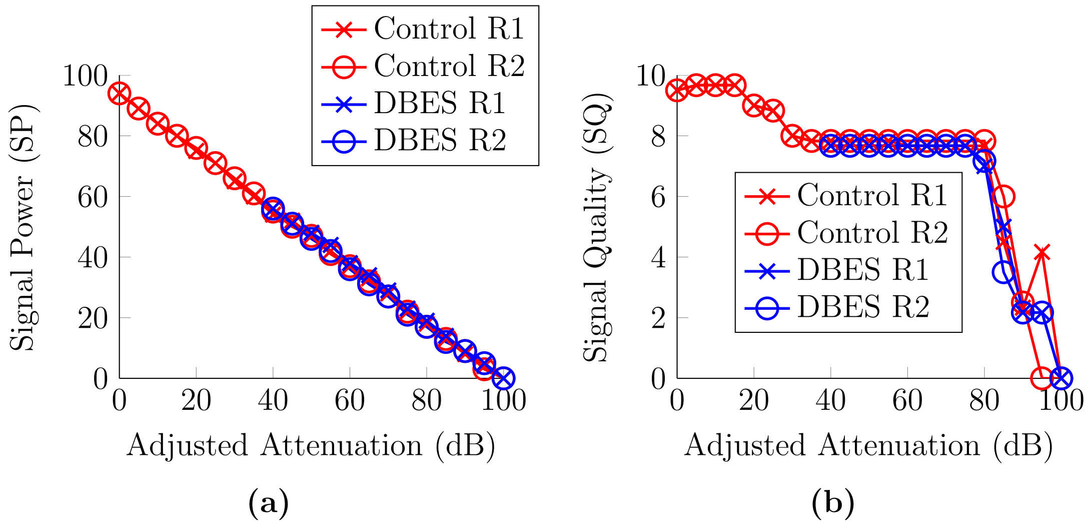


Figure 4.35: Radio attenuation results showing the signal power and signal quality

As you can see, when the radios were connected to the DBES they behaved identically as when they were connected with attenuators over a range of approximately 60 dB. This means that the DBES with radios attached obtained about 60 dB of dynamic output power range. Theoretically, the DBES should be capable of more than 90 dB; however, it was not possible to show that with the radios tested because of isolation issues explained in Section 3.6.4. This means that there will be a radio received power ceiling that may not be true to the real world. If a different radio was used that could receive a signal 10 dB lower than this radio, it is possible that 70 dB of dynamic range could be claimed because theoretically the DBES should be representing signals at that level accurately. This is mentioned to simply point out another case of a metric depending on the characteristics of the radio used.

This test also gives some meaning to the SP and SQ metrics. As can be seen by the test, the SP metric could almost be given units of decibels because for every 10 dB of change in the input signal the SP metric changes by 10. A value of zero in SP means that the radio cannot see the input signal. The SQ metric was scaled to a value from zero to ten. As can be seen in Figure 4.35, under good propagation path conditions, the SQ metric is around 7.7. Once the signal power gets to a point where the radio is struggling to receive a signal, the SQ metric quickly drops off. Also note that in the control experiment with the attenuator, the SQ metric actually increases for the first radio at an attenuation of 95 dB. In observing the radio output during times when a received signal was very weak and on the verge of being detectable, the SQ metric would jump around significantly. In truth the first radio was occasionally reporting seeing the second radio and occasionally reporting not seeing it. This is mentioned here, because this radio behavior could probably be seen in the five node scenario. This shows that it is radio behavior and not the DBES interfering with radio behavior because it happens when just attenuators connect the radios.

4.3.3.2 Two Node Divided Wall Scenario

This scenario used two nodes divided by a very thick wall with a gap in the middle. The goal was to show a simple on/off scenario, but it turned out to be a little more interesting than that. The scenario was generated by Wireless InSite. While it was not possible to generate a control for this test, it was possible to view how the DBES handles this scenario using a signal generator and spectrum analyzer. This scenario was first run by injecting a narrowband RF signal into the first channel and viewing the output of the second channel on the spectrum analyzer. This gives a good indication that the DBES is operating as it should; however, it is possible for things to be missed on a spectrum analyzer and so two Ad-Hoc Networking radios were used to confirm that the system will perform as expected in this scenario.

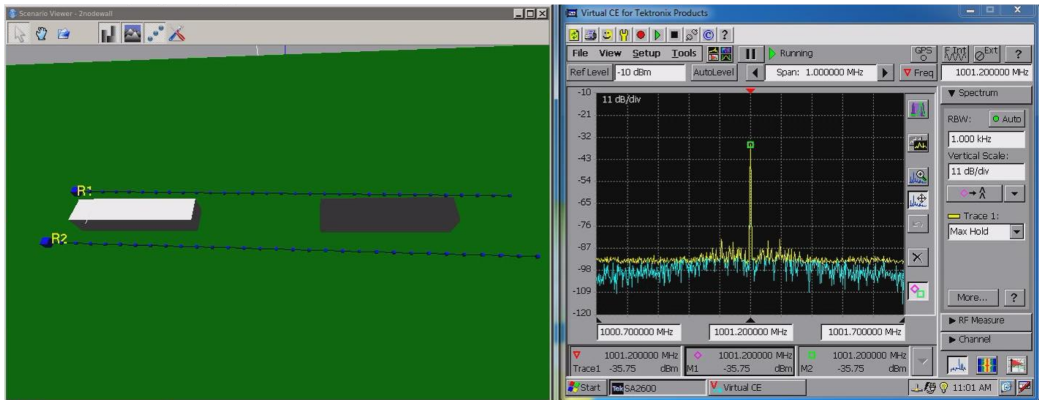
The scenario viewer in the DBES GUI was used to display the scenario and the output of the spectrum analyzer was placed along side. A freeware program called ActivePresenter was used to capture the image on the screen and produce a video. The synchronization between the two was not perfect as on close inspection is was off by approximately 200 milliseconds. There is inherent delay between the spectrum analyzer seeing the signal, producing the spectrum results and sending the screen image over ethernet. In addition the DBES GUI was operating remotely over an X-windows session which has delay as well. These factors prohibit exact timing; however, it is good enough to give confidence that the system is operating as expected.

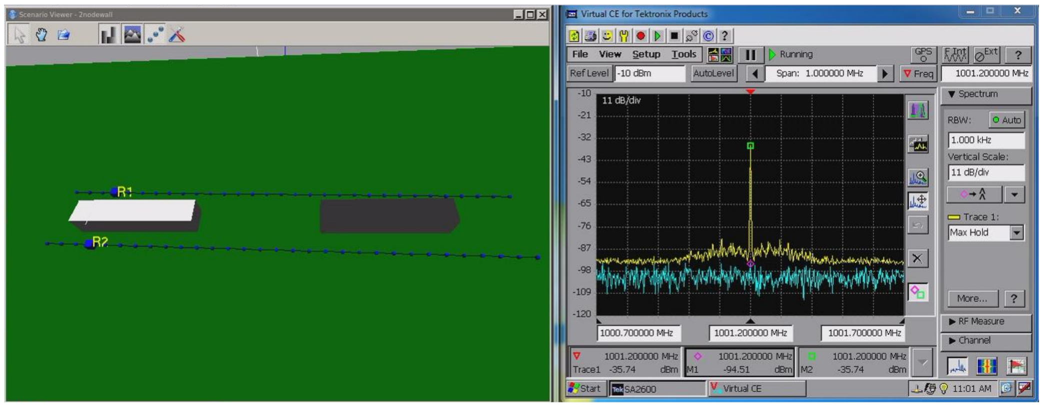
The results in Figure 4.36 show that the signal disappears when blocked by the wall as expected and reappears when the walls are not there. Additionally, the results show that the signal does not just disappear and reappear instantly as can been seen in Figure 4.36d. The scenario compiler found propagation paths that refracted and reflected off the walls so that a signal could be seen when the line of sight was lost. To ensure this was accurate, the parameters generated by the Wireless In-site were inspected closely and it was confirmed that there was a significant signal power coming through reflections when the radios came close to the edge of the wall. It is important for the DBES to be able to recreate such a situation as it is common in urban environments.

Next the same scenario was run with radios attached and the results are shown in Figure 4.37. This shows typical radio performance as expected for this scenario. The radio statistics show that they are unable to communicate behind the wall but as they emerge the link is formed and they begin to send data.

4.3.3.3 Two Node Run-by Scenarios

This scenario involves two radios in a high Doppler scenario. One radio was stationary and placed at the center of a flat open terrain. A second radio started at one end of the terrain and drove in a straight line at a constant velocity past the first radio and off to the end of the wall where the scenario ends. At the closest point, the radios were 100 meters apart. The idea behind this is that it would test Doppler shift capabilities and the change in Doppler shift. This scenario was created first with REMCOM but the actual parameters used were created with Matlab. The REMCOM scenario did not produce any multipath environment which means that the ground reflections were too small in comparison with the line of sight signal to be used. At the time of this dissertation the REMCOM software was taking too


(a) Start of scenario, signal is seen


(b) Behind first wall, signal not seen

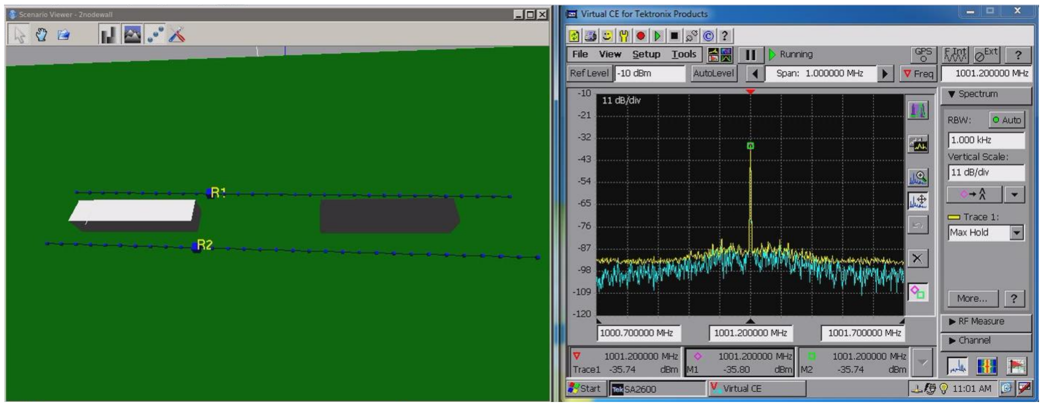
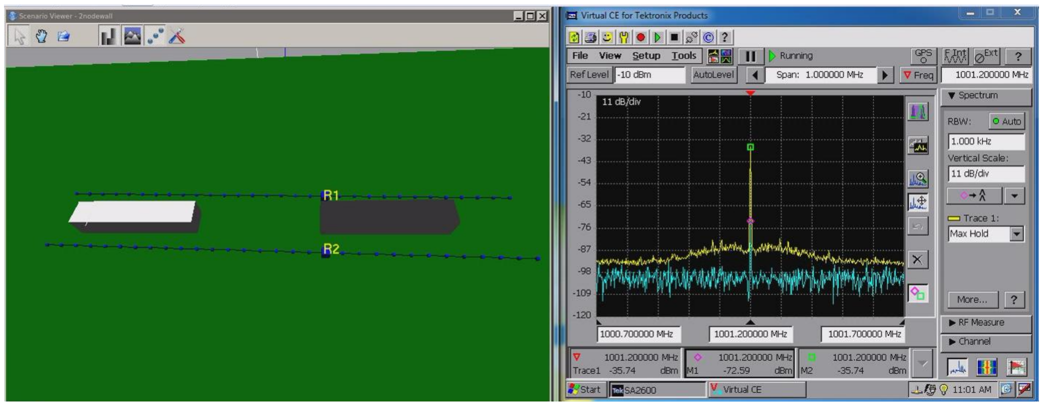
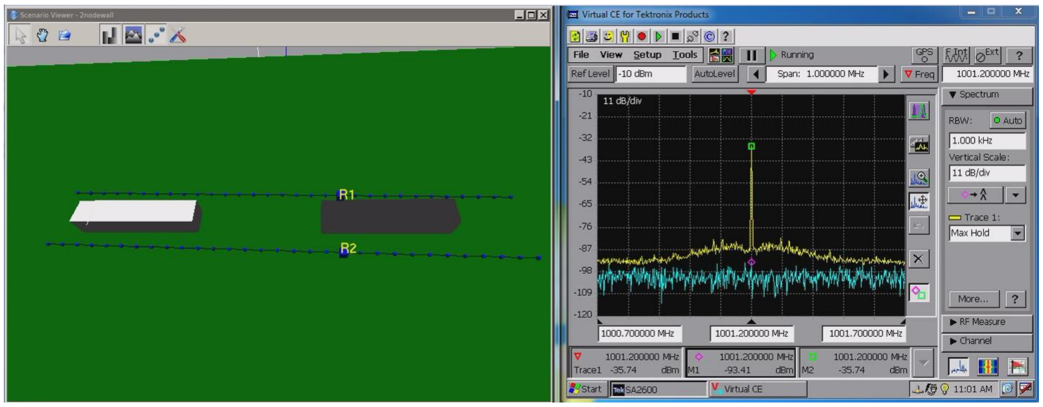

(c) Past first wall, signal seen

Figure 4.36: Two node two wall scenario with time progressing from (a) to (f). The purple M1 purple diamond marker denotes the current power on the blue trace that is somewhat hidden by the yellow trace which is a maximum hold. Figure continued on next page



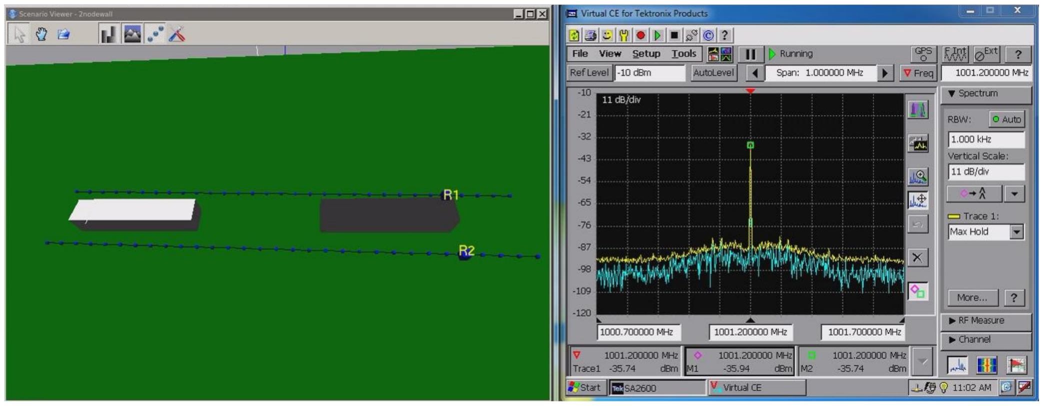
Ref Level: -10 dBm
Span: 1.000000 MHz
Freq: 1001.200000 MHz
11 dB/div
RBW: 1.000 kHz
Vertical Scale: 11 dB/div
Trace 1: Max Hold
1000.700000 MHz 1001.200000 MHz 1001.700000 MHz
Trace1 1001.200000 MHz -35.74 dBm
M1 1001.200000 MHz -72.59 dBm
M2 1001.200000 MHz -35.74 dBm

(d) Line of sight is lost but signal visible due to multipath



Ref Level: -10 dBm
Span: 1.000000 MHz
Freq: 1001.200000 MHz
11 dB/div
RBW: 1.000 kHz
Vertical Scale: 11 dB/div
Trace 1: Max Hold
1000.700000 MHz 1001.200000 MHz 1001.700000 MHz
Trace1 1001.200000 MHz -35.74 dBm
M1 1001.200000 MHz -93.41 dBm
M2 1001.200000 MHz -35.74 dBm

(e) Behind second wall, signal not seen



Ref Level: -10 dBm
Span: 1.000000 MHz
Freq: 1001.200000 MHz
11 dB/div
RBW: 1.000 kHz
Vertical Scale: 11 dB/div
Trace 1: Max Hold
1000.700000 MHz 1001.200000 MHz 1001.700000 MHz
Trace1 1001.200000 MHz -35.74 dBm
M1 1001.200000 MHz -35.94 dBm
M2 1001.200000 MHz -35.74 dBm

(f) After second wall, signal seen

Figure 4.36: Continued from previous page

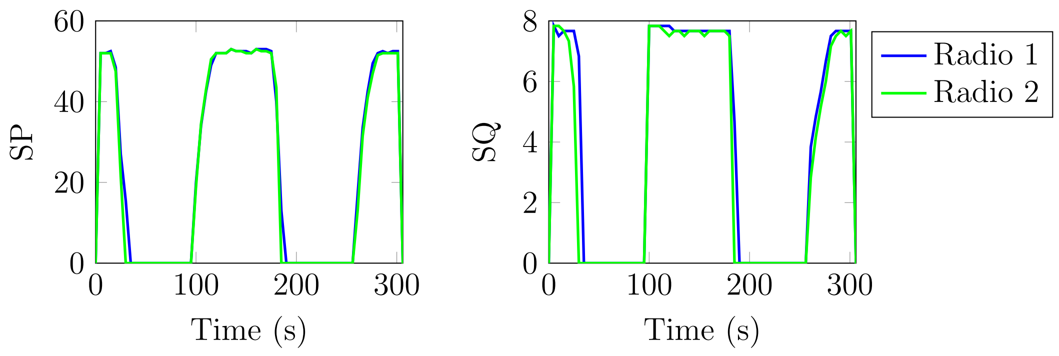


Figure 4.37: Two node divided wall scenario results with radio reporting SP and SQ metrics

long to run, so Matlab was used to generate the parameters over a larger distance. This scenario is similar the one mentioned in the use case Section 3.6, only the long wall was not used in this scenario. That section can be consulted to see how the parameters were generated.

Theoretically, this scenario should produce a Doppler shift and amplitude as shown in Figure 4.38. Notice how it is relatively constant positive Doppler shift at first but then quickly changes as the radios pass each other until it is a relatively constant negative Doppler shift. Figure 4.39 shows snapshots of the video recorded that demonstrates the theoretical behavior exactly. Notice that as the radios are passing each other, the Doppler shift changes so quickly that the Spectrum Analyzer does not scan fast enough to capture the change. The result is only a few data points are grabbed while the shift is transitioning. This is a deceiving result as the Doppler shift produced by the DBES is actually very consistent and does not happen in discrete steps as shown by the spectrum analyzer. The signal interpolation ensures that the signal remains continuous throughout the rapid Doppler change. Perhaps the best proof of this comes from the next part where actual radios are connected to the system and this scenario is run.

Now that the theoretical behavior was confirmed with the DBES, radios were connected to see how they would perform with such a Doppler shift and these results are

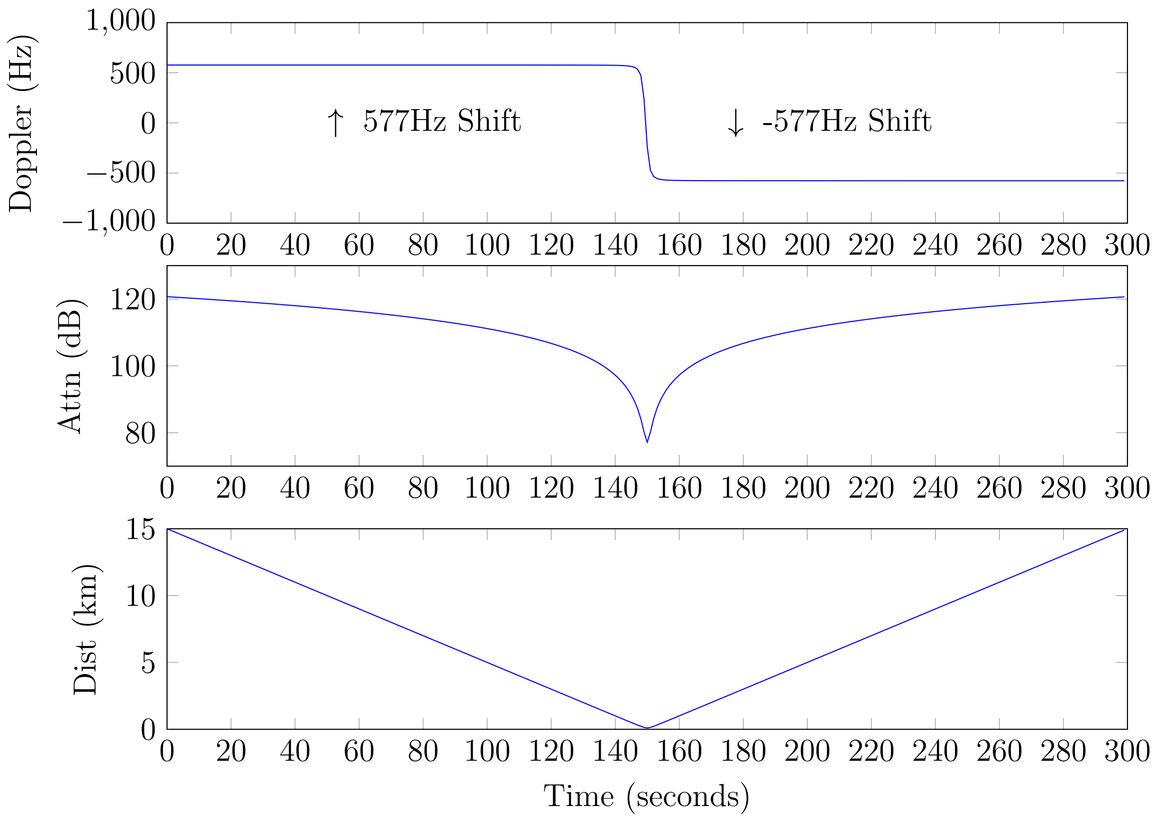
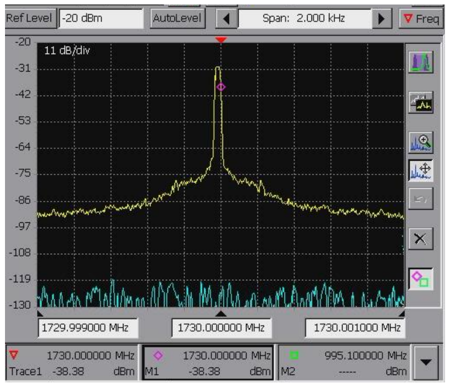
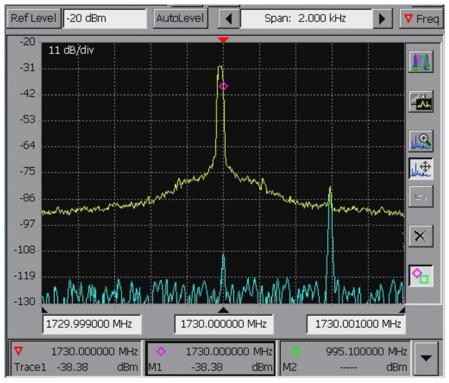


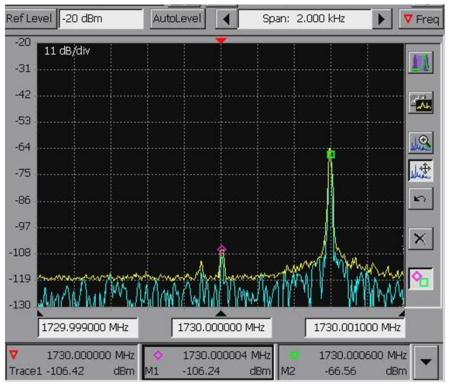
Figure 4.38: Theoretical Doppler shift and attenuation (FSPL) for the run-by scenario with distance between the radios also shown.


Ref Level -20 dBm AutoLevel Span: 2.000 kHz Freq
11 dB/div
1729.999000 MHz 1730.000000 MHz 1730.001000 MHz
Trace1 1730.000000 MHz -38.38 dBm | M1 1730.000000 MHz -38.38 dBm | M2 995.100000 MHz ----- dBm

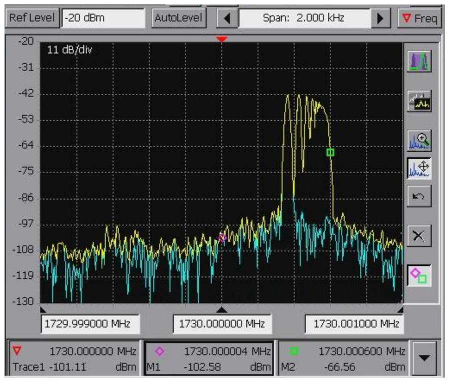
(a) 0:00 Prior to scenario start: yellow shows unity gain and blue shows zero gain


Ref Level -20 dBm AutoLevel Span: 2.000 kHz Freq
11 dB/div
1729.999000 MHz 1730.000000 MHz 1730.001000 MHz
Trace1 1730.000000 MHz -38.38 dBm | M1 1730.000000 MHz -38.38 dBm | M2 995.100000 MHz ----- dBm

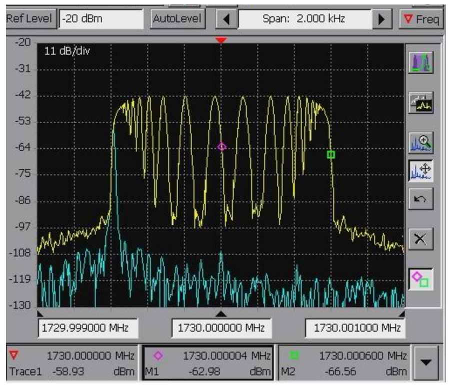
(b) 0:11 Scenario just stared, yellow line (maximum hold) reset shortly after


Ref Level -20 dBm AutoLevel Span: 2.000 kHz Freq
11 dB/div
1729.999000 MHz 1730.000000 MHz 1730.001000 MHz
Trace1 1730.000000 MHz -106.42 dBm | M1 1730.000004 MHz -106.24 dBm | M2 1730.000600 MHz -66.56 dBm

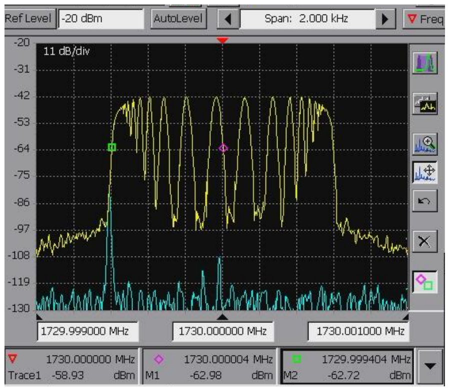
(c) 2:09 Heavy Doppler shift as radios approaching passing point


Ref Level -20 dBm AutoLevel Span: 2.000 kHz Freq
11 dB/div
1729.999000 MHz 1730.000000 MHz 1730.001000 MHz
Trace1 1730.000000 MHz -101.11 dBm | M1 1730.000004 MHz -102.58 dBm | M2 1730.000600 MHz -66.56 dBm

(d) 2:26 Shortly before the passing point showing rapidly changing Doppler


Ref Level -20 dBm AutoLevel Span: 2.000 kHz Freq
11 dB/div
1729.999000 MHz 1730.000000 MHz 1730.001000 MHz
Trace1 1730.000000 MHz -58.93 dBm | M1 1730.000004 MHz -62.98 dBm | M2 1730.000600 MHz -66.56 dBm

(e) 2:34 The radios have passed and there is a heavy negative Doppler shift


Ref Level -20 dBm AutoLevel Span: 2.000 kHz Freq
11 dB/div
1729.999000 MHz 1730.000000 MHz 1730.001000 MHz
Trace1 1730.000000 MHz -58.93 dBm | M1 1730.000004 MHz -62.98 dBm | M2 1729.999404 MHz -62.72 dBm

(f) 4:54 Scenario is almost over and the signal strength is lower

Figure 4.39: The two node run-by scenario testing with RF signal generator and spectrum Analyzer. The yellow line is a maximum hold and the blue line is the current signal.

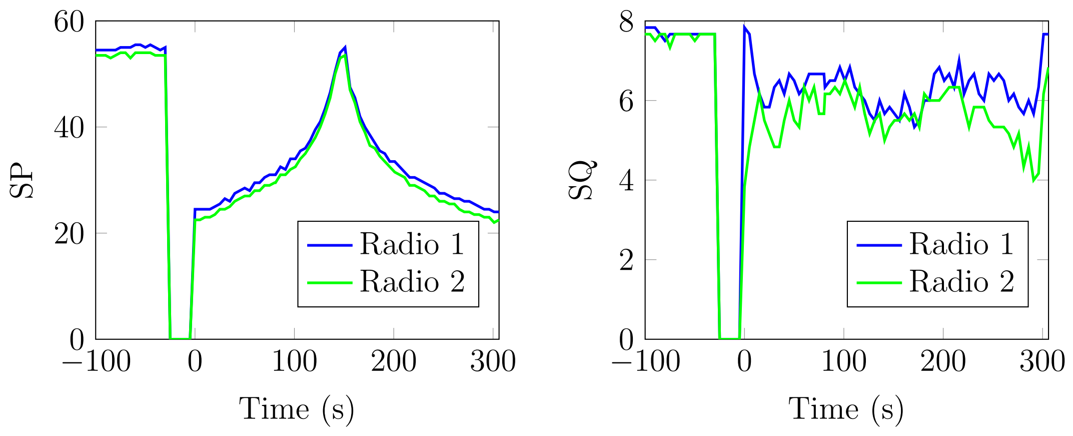


Figure 4.40: Two node run-by scenario results with radio reporting SP and SQ metrics

in Figures 4.40. These figures show a zero attenuation state and then full attenuation state prior to starting scenario. This was done and is shown to give a baseline to the radio behavior. The actual scenario starts at $t = 0$ on the graphs and proceeds for 300 seconds. At first glance these results appear to be relatively melodramatic as the radios maintained a constant state of connectivity throughout the scenario. This however, is exactly what the radios should do. Notice that the radios do not lose connectivity during the rapid Doppler shift and that the signal quality metric the number of packets received does not change significantly. This is how the radios are expected to perform and is strong evidence of the quality of the signal produced. If the DBES produced a non-continuous signal that discretely changed frequencies, it would be likely that significant errors would be seen in the radios.

Another interesting piece of information is that the signal quality metric has been diminished from a standard attenuation. The question is, did the DBES cause this in error or is this an actual effect that would be seen in the field? Unfortunately, this question cannot be definitively answered in this dissertation as it would be necessary to take the radios into the field by a racetrack and borrow some expensive cars. Notice that the SQ metric prior to starting the scenario is ideal and only when the scenario begins does the decrease in SQ occur. This proves that the DBES was set up correctly and the diminished SQ occurs because of channel effects produced by the DBES. Given that the signal interpolation noise was not seen when running this scenario with a signal generator and spectrum analyzer, and that the DBES was correctly setup prior to the scenario running, there is a lot of evidence supporting the theory that this is exactly how the radios would preform in the real world. This also demonstrates the power of the DBES by showing a scenario that is very complicated to reproduce in the real-world is relatively easy to perform in the lab with the DBES.

4.3.3.4 Five Node Scenario

The five node scenario is based off a location called Dupont Circle in Washington DC. This scenario was shown in the REMCOM Wireless InSite section (Figure 3.14) and is shown with a different view in Figure 4.41. The scenario area is approximately one half mile by one half mile. The Wireless In-site software was used to generate this scenario. This particular terrain and building information came with the software, which made it easy to simply place radios and move them on a particular route through the city. These particular routes were almost picked randomly. The goal was to spread the radios out and try to induce a lot of channel effects with the urban environment. The hope was that the radios would gain and lose connectivity because of the buildings. Three different runs were successfully completed with this scenario. This was done to see if the radios would display repeatable behavior and to have some comparison of performance. There is one other issue to discuss prior to showing the results. One of the radios did not have the equipment to gather metrics from them. Because the paths are generally symmetrical this information was filled in based on what the other radio said the link was. Also note that the radio without the data gathering ability was radio four in the first test and radio three in tests two and three. As described in the use case section the system was set up, the radios were connected, and the scenario was run. Before showing the results for every node, the results from a single signal path will be displayed as shown in Figure 4.42.

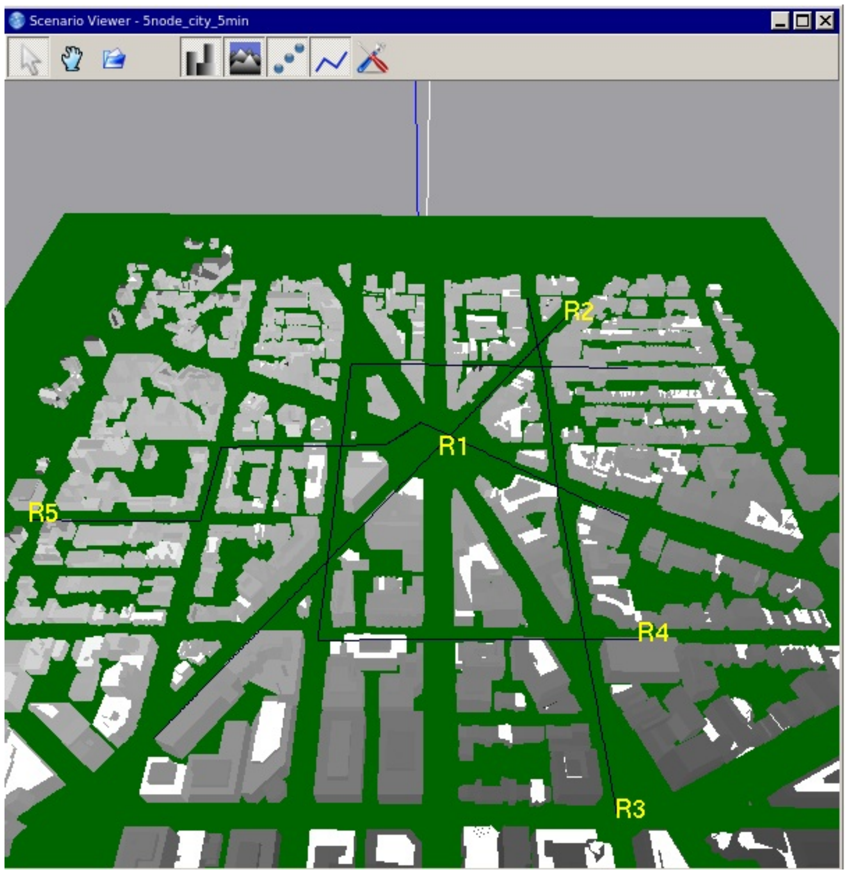


Figure 4.41: 3D model of a location in Washington DC along with the five different radio paths used in the five node scenario

Figure 4.42 shows the signal power metric for the signal path from radio three to radio one. This means that radio three was the transmitting radio and radio one was the receiving radio. The three different color lines each represent a run of this same scenario. If the repeatability of the test was perfect each of the lines would sit directly on top of one another. This is not exactly the case here; however, the lines trend together. The biggest difference is a standard offset in the power level from run to run. Each of these runs occurred on different days and the setup was slightly different each time. The setup was different because there was no automated way of getting the power settings just right. Because there was no automation, the result was different each time. This is one major lesson learned taken from this experiment

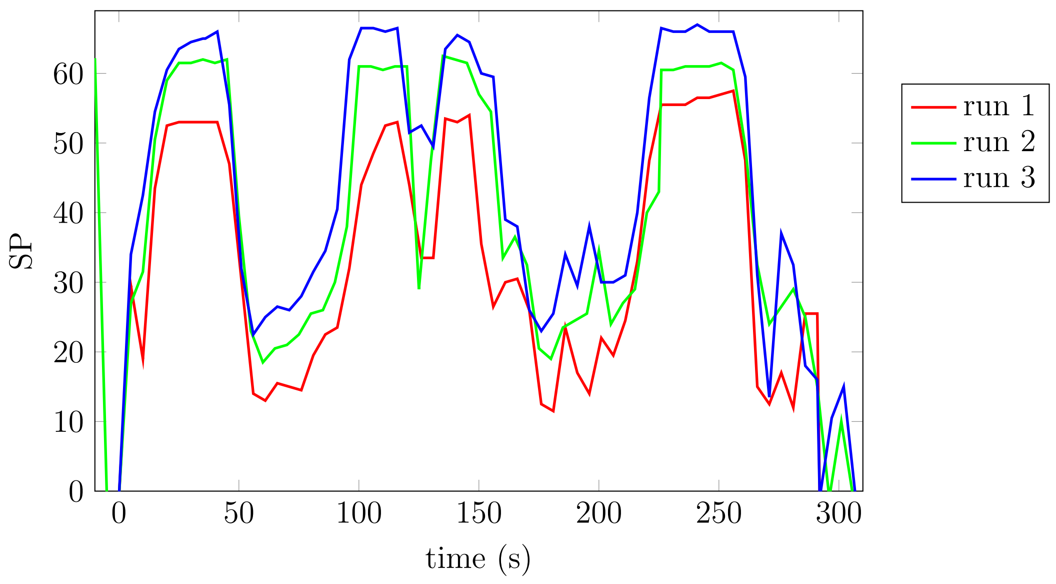


Figure 4.42: SP result for radio 3 to radio 1 in five node scenario

and development began on automating this process immediately. For these results however, the data will not line up perfectly. What is very good is that the data trends together exceptionally well. So while there is a SP offset, the SP for each day rises and falls in unison.

Next, the signal path between radio three and radio one will be examined thoroughly to determine if the DBES correlates with the scenario. Radio one is stationary and radio three moves North in a straight line one block over from radio one as shown in Figure 4.41. In the beginning, radio three has almost line of sight to radio one, until it drives past the first triangular block of buildings. This corresponds exactly with the SP results as this is the first large hump seen between 0 and 50 seconds. The time from 50 and 100 seconds is characterised by the loss of line of sight. The SP takes a large dip but is still connected due to the multipath around the building. The SP slowly increases as it nears the corner of the building until line of sight is reestablished at around 100 seconds. The time between 100 and 150 seconds is characterized by line of sight in the beginning and end with a small building blocking line of sight in the middle. The time between 150 and 220 seconds is similar to the time period between 50 and 100 seconds where there is a group of buildings blocking line of sight, but there is decent amount of RF reflections and diffractions that maintains connectivity between the radios. The time between 225 and the end of the scenario shows line of sight established at first but then as the radio moves up the street it loses line of sight and the signal level continues to drop as radio three moves away from radio one. This clearly shows an excellent correlation between the scenario and the SP results.

The signal quality metric was also plotted and Figure 4.43 shows the SQ metric between radios three and one for each of the three runs. Unfortunately, this shows another issue with one of the runs. For runs one and two the SQ metric is exactly as one would expect, the SP was good but occasionally dipped low and this kept the SQ high for the majority of the time. Run three however, shows an odd middle of the road and oscillating metric. Something was not quite set up correctly with this run. What this issue was is hard to say but a good guess is that the received signal power was brought in slightly too high resulting in occasional bad symbols as the power level peaked. This would result in a signal that was received at approximately the correct power level but with some of the information in the packet garbled because of the DBES. The radios still communicated and the power levels were still accurate; however, the fidelity of the third run is diminished because of this issue.

Next the signal power is plotted for all the signal paths and is shown in Figure 4.44. Each plot in this figure is created in the same manner as the plot shown in Figure 4.42. The first thing seen in this plot is that generally the signal power results correlate with themselves over time very nicely. Each of them has the same signal power offset as described previously, which does make sense; if one was off then likely each of them would be off. This dissertation will not describe in detail every signal path as was done in the previous paragraph; however, in spot checking each of them an excellent correlation between scenario and signal power can also be seen. There are

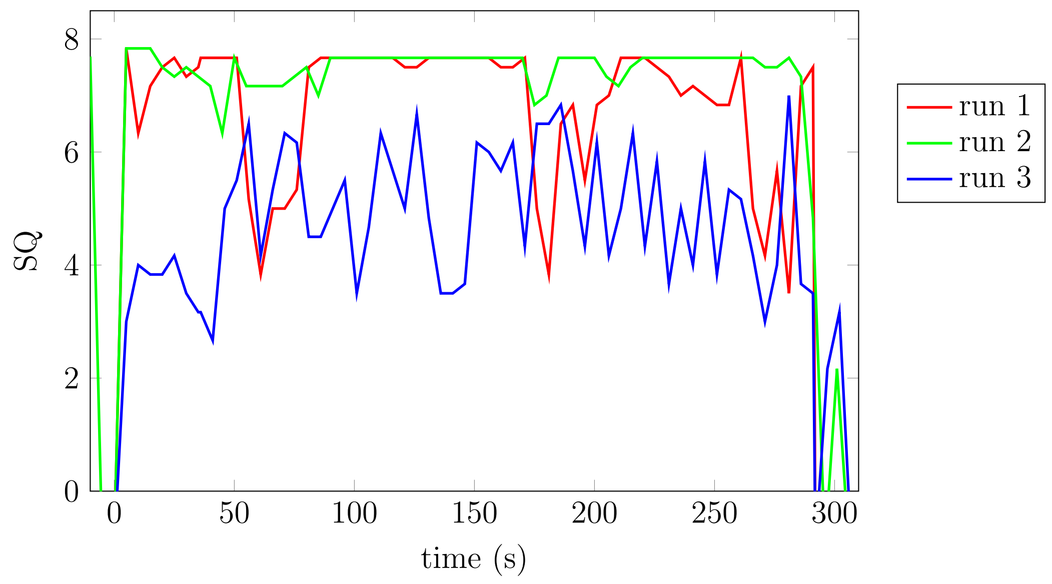


Figure 4.43: SQ result for radio 3 to radio 1 in five node scenario

a few exceptions and one of them is in the signal path from radio two to radio four. The first run does not correlate with the runs two and three for a small period of time. The data for that run was actually not collected by radio four because that radio did not have the data capturing hardware. That data was captured by radio two. If one looks at the signal path from radio four to radio two this leads to an interesting discovery. The signal paths from two to four and from four to two are not reciprocal from approximately 200 to 275 seconds in the scenario. This could be an issue with the scenario compiler or this could be accurate. Unfortunately there is no way of knowing for sure the cause, but it is an interesting result that requires investigation outside the scope of this dissertation.

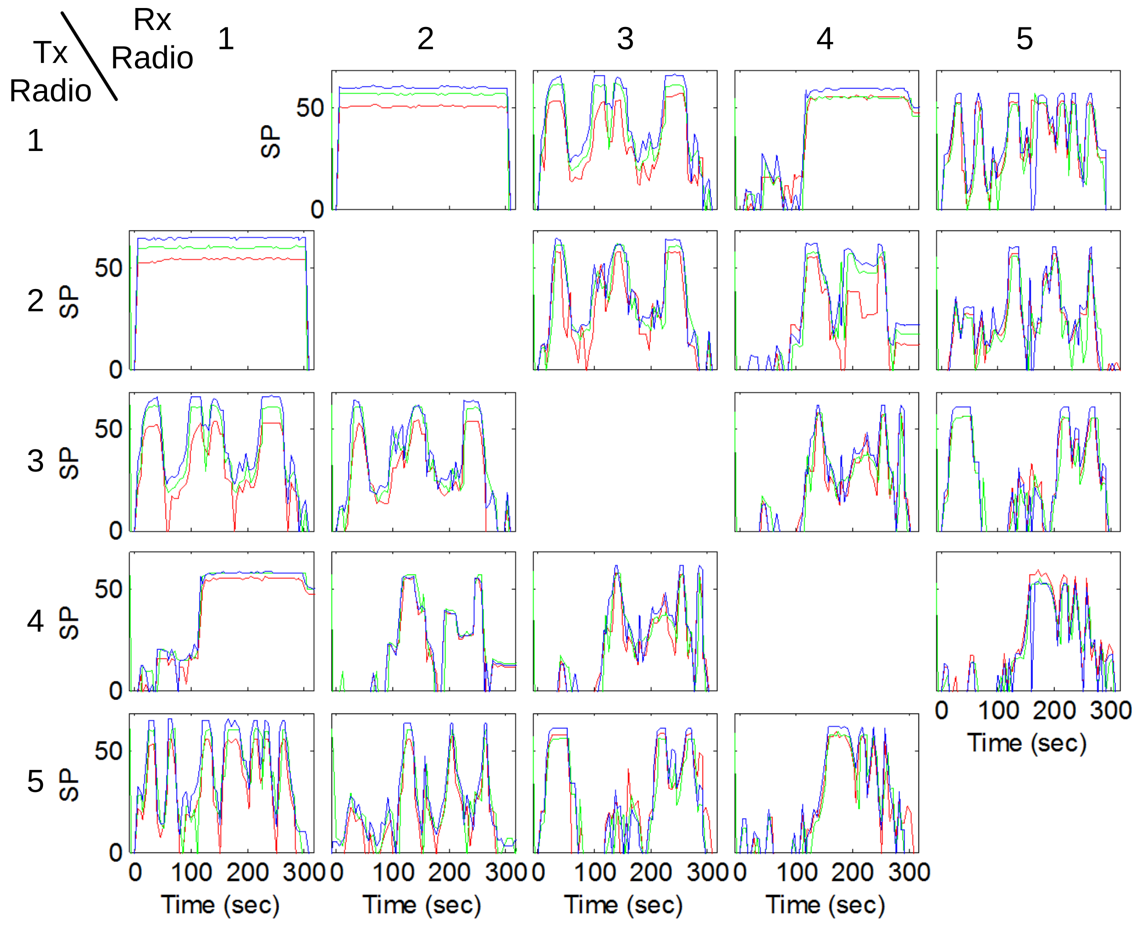


Figure 4.44: The signal power metric shown for three different runs of the 5 node urban scenario. The rows correspond to a given transmitting radio and the columns correspond to a receiving radio. With a few exceptions these results show excellent repeatability from run to run.

CHAPTER 5: CONCLUSION

This dissertation has described the research into and creation of a scalable digital wireless channel emulator dubbed the Digital Battlespace Environment Simulator. The thesis question for this dissertation is: constructing a DWCE with a distributed architecture provides superior scalability than previously documented systems while maintaining high fidelity. To confirm the thesis statement three questions were asked. The first question was: is it possible to build a DWCE utilizing a distributed architecture? To answer this question the feasibility study was performed and a two channel prototype was constructed and analyzed as shown in Section 4.1. Most of the basic components that were eventually used in the large scale DBES were first tested in this prototype. There were many uncertainties about how the system would behave and if it would work at all. The prototype addressed the majority of those uncertainties and showed that yes, it is indeed possible to construct a DWCE utilizing a distributed architecture.

The second question was: can this architecture provide a scalable and cost effective solution as compared to other testing alternatives? To answer this question the scalability study was performed as shown in Section 4.2. Using resource utilization information compiled from working DBES VHDL code, the required hardware was determined based on the number of channels, number of propagation paths, and signal bandwidth. The study showed that using the same (V,8,2)-network design and hardware that is in the working DBES it is possible to reach 102 node system and still maintain excellent signal bandwidth and number of propagation paths. In addition, using the same hardware but redesigning the network to a hypercube, it would be possible to reach a 1250 node system with 625 FPGAs. The largest hypercube system would have significantly limited signal bandwidth and only one propagation path per node; however, smaller hypercube systems could be viable solutions depending on requirements. For whatever sized system desired, the study showed exactly how many propagation paths and signal bandwidth would be achievable. In addition, new FPGAs are constantly being developed. This study only used the current hardware created, but this analysis could be performed with Virtex 7 or future technology that would increase the network bandwidth and available resources significantly. The algorithms used to determine these results could easily be re-run with the new specifications. This would produce an entirely new set of results that would allow for even larger DWCEs to be created. In addition to the ability to scale, the section on the hardware cost of the system showed that for less than half the cost of a field test, it was possible to build a DWCE for that test and then run that field test in the lab. So the answer to this question is a definite yes, it is a scalable and cost effective solution for testing wireless devices.

The final question is: is the fidelity of the system good enough to be a substitute for the alternative field test? It was discussed in the introduction that it was impossible to give a definitive yes to this question. There are too many variables having to do with different radios and different scenarios that all of them cannot be evaluated; however, a best attempt to determine the fidelity of the DBES was presented in Section 4.3. Any given performance metric for the DBES can be categorised as signal power fidelity, signal delay fidelity, or signal summation fidelity. The study conducted multiple theoretical and experimental tests for each of these categories of fidelity. Some tests showed that the system was not as accurate to the real world as may be desired. There was significant noise under heavy Doppler shifts, the frequency selective fading was not as accurate as it could be, and the noise generated from the AFE causes several limitations in scenarios. The system did however, perform very well in many cases as shown in by the accuracy of the attenuation and the Doppler shift for normal terrestrial vehicle speeds. The signal interpolation was a novel feature not previously investigated that proved to be very powerful; however, was also the cause of many of the fidelity issues with the system. If this signal interpolation was not done, it would not be possible to produce an accurate Doppler shift across the band of interest and it would not be possible to do frequency selective fading at all. In addition, it is believed that many of the issues found with the signal interpolation can be solved by improving the interpolation scheme and by over-sampling the signals. While some tests proved to not be as accurate as desired, the capability to even produce these effects in emulation is unprecedented in literature. The final part of the fidelity study was the observational tests with actual radios. These tests lend credence to the work done in the experimental and theoretical sections and affirms that some of the issues with the noise and accuracy do not matter, at least for the radios under test. It showed that the system could be used to produce repeatable results which is extremely important to radio designers and evaluators and represents a new modality for analyzing these systems that had not been present prior to the DBES. So the answer to this question is a qualified yes: as long as the wireless devices used with the DBES ignores the fidelity concerns presented, and the corner cases that the DBES cannot accurately reproduce are not key concern to the evaluator, then the DBES would be an excellent substitute for a real-world field test. The DBES was never intended to completely replace field testing, but it does have the ability to dramatically reduce the time spent in the field by conducting the majority of the testing in the lab, thus dramatically reducing the cost and time to conduct the test, and produce repeatable results that can more easily be evaluated by an engineer.

As the answer to each of the three thesis questions was a yes or a conditional yes, the thesis statement has been confirmed. Constructing a DWCE with a distributed architecture does provide superior scalability than previously documented systems while maintaining high fidelity.

The completion of this project allows for several new avenues of research and continued work with the system developed. First, the FPGA cluster created for this dissertation is unique and could be used to research many high performance computing topics. As mentioned in the hardware section, this cluster has roughly four times the bandwidth and ten times the computational capability of the previous SPIRIT cluster used by the RCS Lab. This will allow greater scale of research and the creation of designs not previously possible. In addition, there is much research that could be done to improve the quality of the emulator. This dissertation discussed the issues with the signal interpolation and this could be a focal point for further research which would improve many of the issues seen with Doppler and fading. Another major limitation found from this research is the AFE. New designs and improvements on the AFE could dramatically impact the fidelity of the system. This dissertation never spoke about using multiple channels to cover a larger frequency range which would improve the emulation for frequency hopping radios. In the design section the automatic gain control was discussed but never implemented and tested which could improve dynamic range considerably.

In addition to all the research topics mentioned in the previous paragraph, the simple utilization of the 32-node DBES that was constructed is novel. It is our goal to use 32 radios with the system in a short period of time and publish another article on the outcome of that evaluation. The goal is to then allow other universities and organizations to use the system. As planned from the beginning, all the hardware for this system (and Wireless In-Site) can be purchased by anyone. The code that runs the DBES is government purpose rights, which means that it can be distributed to universities and other organizations with good cause. In addition there is little need for a basic user to have the source code, so likely the government can hand out binaries and executables with little resistance. Hopefully the technology presented in this dissertation will allow many other people to improve on their wireless device design and open up new avenues of research not previously possible.

REFERENCES

[1] Amirhossein Alimohammad and Bruce F. Cockburn. Compact implementation of a sum-of-sinusoids rayleigh fading channel simulator. In *Signal Processing and Information Technology, 2006 IEEE International Symposium on*, pages 253 – 257, aug. 2006.

[2] et. al. Billy Zhong, Chris Niessen. The mitre tactical channel emulation system, 2013.

[3] K. Borries, Xiaohui Wang, G. Judd, P. Steenkiste, and D. Stancil. Experience with a wireless network testbed based on signal propagation emulation. In *Wireless Conference (EW), 2010 European*, pages 833 –840, april 2010.

[4] K.C. Borries, G. Judd, D.D. Stancil, and P. Steenkiste. Fpga-based channel simulator for a wireless network emulator. In *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pages 1 –5, april 2009.

[5] S. Buscemi, W. Kritikos, and R. Sass. A range and scaling study of an fpga-based digital wireless channel emulator. In *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*, pages 137–144, 2013.

[6] S. Buscemi and R. Sass. Design of a scalable digital wireless channel emulator for networking radios. In *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, pages 1858–1863, 2011.

[7] S. Buscemi and R. Sass. Design and utilization of an fpga cluster to implement a digital wireless channel emulator. In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pages 635–638, 2012.

[8] Chen Chang, Kimmo Kuusilinna, Brian Richards, and Robert W. Brodersen. Implementation of bee: a real-time large-scale hardware emulation engine. In *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays*, FPGA '03, pages 91–99, New York, NY, USA, 2003. ACM.

[9] D. Chizhik, J. Ling, P. Wolniansky, and R. Valenzuela. Multiple input multiple output measurements and modeling in manhattan. In *Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th*, volume 1, pages 107–110 vol.1.

[10] A. Dassatti, G. Masera, M. Nicola, A. Concil, and A. Poloni. High performance channel model hardware emulator for 802.11n. In *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*, pages 303 – 304, dec. 2005.

[11] Elektrobit (EB). Eb propsim radio channel emulator user reference, 2012.

[12] H. Eslami and A.M. Eltawil. A scalable wireless channel emulator for broadband mimo systems. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 2592 –2597, june 2007.

[13] Dan Gordon. La jolla covering repository. www.ccrwest.org/cover.html.

[14] Glenn Judd and Peter Steenkiste. Using Emulation to Understand and Improve Wireless Networks and Applications. In *NSDI 2005*, 2005.

[15] Kimmo Kuusilinna, Chen Chang, M. Josephine Ammer, Brian Richards, and Robert W. Brodersen. Designing bee: a hardware emulation engine for signal processing in low-power wireless applications. In *EURASIP Journal on Applied Signal Processing, special issue on Rapid Prototyping of DSP Systems*, 2003.

[16] R. Ohs S. Buscemi M. D. Schmiedekamp, A. J. Kuhlman. High fidelity modeling of spatio-temporally dense multi-radio scenarios. In *The Applied Computational Electromagnetics Society (ACES), 2013*, 2013.

[17] C. Mehlfuhrer, M. Rupp, F. Kaltenberger, and G. Humer. A scalable rapid prototyping system for real-time mimo ofdm transmissions. In *DSPenabledRadio, 2005. The 2nd IEE/EURASIP Conference on (Ref. No. 2005/11086)*, page 7 pp., sept. 2005.

[18] C. Mehlfuhrer, M. Rupp, F. Kaltenberger, and G. Humer. A scalable rapid prototyping system for real-time mimo ofdm transmissions. In *DSPenabledRadio, 2005. The 2nd IEE/EURASIP Conference on (Ref. No. 2005/11086)*, page 7 pp., sept. 2005.

[19] S. Mellers, B. Richards, H.K.-H. So, S.M. Mishra, K. Camera, P.A. Subrahmanyam, and R.W. Brodersen. Radio testbeds using bee2. In *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, pages 1991–1995, nov. 2007.

[20] P. Murphy, F. Lou, A. Sabharwal, and J.P. Frantz. An fpga based rapid prototyping platform for mimo systems. In *Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 1, pages 900 – 904 Vol.1, nov. 2003.

[21] Patrick Murphy, Feifei Lou, Ashutosh Sabharwal, and J. Patrick Frantz. An fpga based rapid prototyping platform for mimo systems. In *in Proc. 37th Asilomar Conference on Signals, Systems and Computers*, pages 900–904, 2003.

[22] Dan Noneaker. Recommended channel models for wnw 3.1 tests, May 2009.

[23] J.J. Olmos, A. Gelonch, F.J. Casadevall, and G. Fermenias. Design and implementation of a wide-band real-time mobile channel emulator. *Vehicular Technology, IEEE Transactions on*, 48(3):746 –764, may 1999.

[24] T. Ould Bachir and J.-P. David. Performing floating-point accumulation on a modern fpga in single and double precision. In *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*, pages 105 –108, may 2010.

[25] S. Picol, G. Zaharia, D. Houzet, and G. El Zein. Hardware simulator for mimo radio channels: Design and features of the digital block. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1 –5, sept. 2008.

[26] J.G. Proakis. *Digital Communications*. McGraw-Hill, 4th edition edition, 2001.

[27] T.S. Rappaport. *Wireless communications: principles and practice*. Prentice Hall communications engineering and emerging technologies series. Prentice Hall PTR, 1996.

[28] Ron Sass, William V. Kritikos, Andrew G. Schmidt, Srinivas Beeravolu, and Parag Beeraka. Reconfigurable computing cluster (rcc) project: Investigating the feasibility of fpga-based petascale computing. In *Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 127–140, Washington, DC, USA, 2007. IEEE Computer Society.

[29] Andrew G. Schmidt, William V. Kritikos, Rahul R. Sharma, and Ron Sass. Airen: A novel integration of on-chip and off-chip fpga networks. In *Proceedings of the 17th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'09)*. IEEE Computer Society, April 2009.

[30] Inc. Spirent Communications. Sr5500 wireless channel emulator user manual, 2009.

[31] C. Steger, P. Radosavljevic, and J. P. Frantz. Performance of IEEE 802.11b wireless LAN in an emulated mobile channel. pages 1479–1483, 2003.

[32] Xiaohui Wang, Kevin C. Borries, Eric Anderson, and Peter Steenkiste. Network-scale emulation of general wireless channels. In *VTC Fall*, pages 1–5. IEEE, 2011.

[33] J. Yackoski, B. Azimi-Sadjadi, A. Namazi, J.H. Li, Y. Sagduyu, and R. Levy. RFnest #x2122;: Radio frequency network emulator simulator tool. In *Military Communications Conference (MILCOM2011)*, pages 1882 –1887, nov. 2011.