EVALUATION OF SPACE-TIME BLOCK CODES UNDER CONTROLLED FADING CONDITIONS USING HARDWARE SIMULATION

A Dissertation

Submitted to

the Temple University Graduate Board

In Partial Fulfillment

of the Requirements for the Degree

DOCTOR OF PHILOSOPHY IN ENGINEERING

---

by

Leonard R Colavito

December 2009

Dr Dennis Silage Dissertation Advisor	Dr Saroj Biswas Committee Member	Dr Li Bai Committee Member
Dr Iyad Obeid Committee Member	Dr. Joseph Picone Committee Member	Dr. Kevin Buckley External Reviewer

ABSTRACT

Evaluation of Space Time Block Codes
Under Controlled Fading Conditions
Using Hardware Simulation

Leonard R Colavito
Doctor of Philosophy
Temple University, 2009
Doctor Advisory Committee Chair: Dennis Silage, PhD

Space time block codes (STBC) are a type of multiple input multiple output (MIMO) communications system that encode a block of information into a sequence of symbol sets that are simultaneously sent from multiple transmit antennas to one or more receive antennas. MIMO communications systems are of interest because of their potential for great channel capacity improvement in multipath digital communications environments. The STBC class of MIMO communications systems has the property of being easily decoded using linear combination of the received signals. These systems are resilient in the face of multipath channel effects.

These types of systems have traditionally been studied using theoretical analyses, software simulations based on probabilistic channel models and real signal based experiments. Probabilistic models simulate channel effects as random variables, but are only estimates of actual conditions. Real signal experiments seek to evaluate system performance under real-world conditions, but are not readily repeatable as the channel conditions are not easily controlled. Both of these modeling methods evaluate system performance in terms of the aggregate results.

This dissertation research presents an approach that introduces deterministic attenuation and delay to probabilistic channel models to allow the evaluation of MIMO system performance under extreme or unusual channel conditions. The approach is demonstrated by constructing a hardware accelerated STBC system model. The model is then used to evaluate the performance of the STBC under controlled path fading conditions.

The STBC system model utilizes a Xilinix® programmable gate array (PGA) device as a hardware accelerator. The model exploits the parallel processing capability of the PGA to simulate a nine path channel model and a three antenna rate 1/2 STBC. Each of the channel paths allows for deterministic adjustment of attenuation and delay in addition to the additive white Gaussian noise (AWGN) and multipath fading effects. The signal paths can be attenuated and delayed both individually and in combination. The model evaluates performance of the communications system in terms of bit error rate (BER) versus signal-to-noise ratio (SNR).

The use of hardware to accelerate the simulation greatly reduces the time required to obtain results. Reduced simulation time improves the use of the model and allows for evaluation under a greater number of conditions, greater number of points for each performance curve or evaluation of lower BER points. The processing rate of the hardware accelerated model is compared to that for an equivalent software model.

The system model provides an extensible platform for future research in communications theory. The system model can be extended by replacement of the STBC, the

implementation of more sophisticated channel models, addition of channel estimation or by increasing the number of signal paths.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1
INTRODUCTION

MIMO Systems

Space time block codes (STBC) are a class of multiple input multiple output (MIMO) systems that are of interest because of their potential for high channel capacity in multipath digital communications environments [1]. A simple STBC wireless system, illustrated in Figure 1-1, employs two transmit antennas and two receive antennas. The system transmits a data stream as a time sequence of symbols sets. During each symbol time, one symbol is transmitted from each antenna simultaneously. At the receive antennas, the STBC allows the original data stream to be reconstructed from the arriving signals [2].

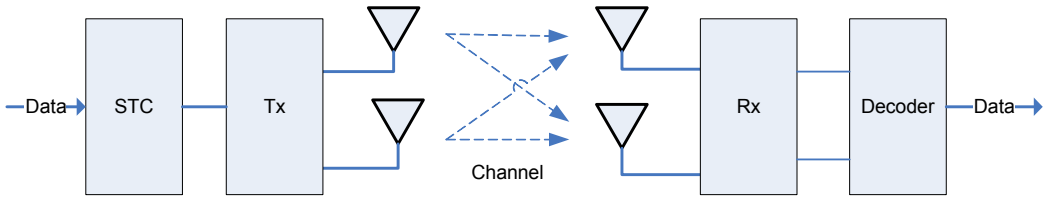

Figure 1-1 MIMO System Diagram

Space time block code wireless systems exploit the spatial diversity of the signal paths between each transmission antenna and each receiving antenna and the decorrelation of the signals that results due to multipath. For a given transmitter power and received signal-to-noise ratio (SNR), there is a theoretical maximum rate at which information can be sent over the channel without error [3]. Systems exploiting spatial diversity can achieve a greater error-free information rate than traditional single-input single-output

(SISO) systems. Thus, utilizing spatial diversity increases the channel capacity in the Shannon sense [4]. Furthermore, this channel capacity is proportional to the number of antennas utilized at the transmitter and receiver, so that even greater channel capacity can be achieved by utilizing more antennas [4].

The increased channel capacity can be utilized either to provide a greater data rate for a single user or to provide equivalent data rates to a greater number of users [5]. The former is of interest to telecommunication providers who want to offer "high bandwidth" services, such as video to mobile customers, while the later has been of interest for wireless networking application and is now under consideration for several emerging wireless standards [6].

The inherent potential of MIMO digital communications systems has prompted considerable study, consisting of theoretical analyses, simulations based on probabilistic channel models and real world experiments. Work by Telatar [4], Foshini [7] and Gans [8] provided the mathematical foundation that predicted the capacity improving potential of MIMO. Seshadri [3] provided methods and criteria for the design of optimal space-time codes. Alamouti [9] demonstrated how a second order (2 transmit antennas and 2 receive antennas) STBC could out perform a traditional (1 transmit and 1 receive antenna) transmission over a Rayleigh fading multipath transmission channel. Higher order codes, those utilizing three or more transmit and receive antennas, have also been proposed using the same technique [1] [10].

Evaluation and proof of the theoretical performance of STBC based communications systems has been attempted through simulation and experimentation. Simulations based on Rayleigh and Rican fading channel models where initially used to validate the predicted performance of STBC codes and MIMO systems [1] [9] [10] [11] [12] [13]. Furthermore, some real world experiments have also been attempted to verify the practical use of MIMO systems [14] [15]. In addition, several programmable gate array (PGA) hardware accelerator simulations have been constructed to allow for the rapid-prototyping of MIMO systems for the evaluation of new codes and implementations [16] [17] [18] [19] [20] [21].

However, all of these systems only consider the aggregate performance of the system under evaluation. One salient contribution of this dissertation research is to present a system model in which specific extreme or unusual conditions may be specifically created and evaluated. Such conditions include the total loss of one or more channel paths or additional phase delay experienced by some paths. This approach is demonstrated by constructing a model that allows such control of channel conditions.

Modeling MIMO Systems

The other salient contribution of this dissertation research is the construction of a PGA hardware accelerated model to study a three antenna STBC under such controlled channel conditions. The MIMO system simulation model is based on a third-order space-time block code. The model consists of a transmitter, a multipath channel, a receiver and a BER evaluation module as illustrated in Figure 1-2.

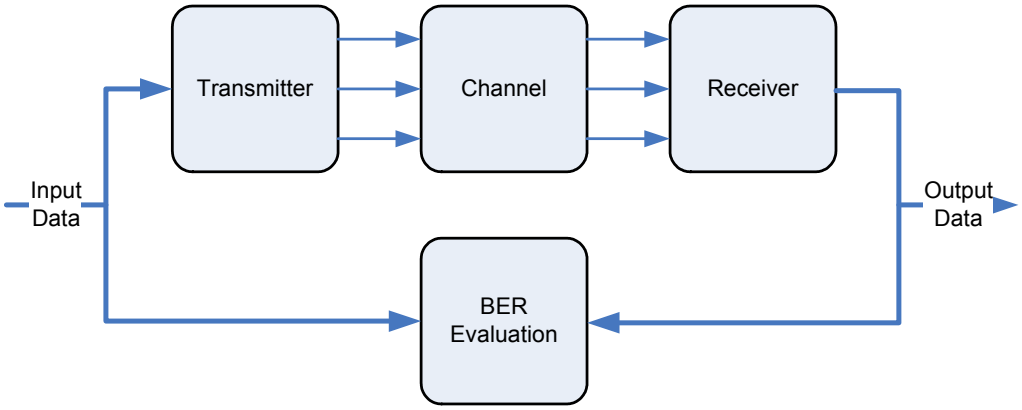

Figure 1-2 MIMO System Model

The transmitter, shown in Figure 1-3, employs quadrature phase shift keying (QPSK) as the base modulation scheme. The QPSK symbols inherently have equal symbol energy that facilitates evenly distributing the transmitter power across the antennas [1]. The transmitter accepts a data stream that is configured as two-bit codewords, which are then mapped to the QPSK symbols. The symbols are sent from the transmitting antennas as blocks by utilizing the STBC.

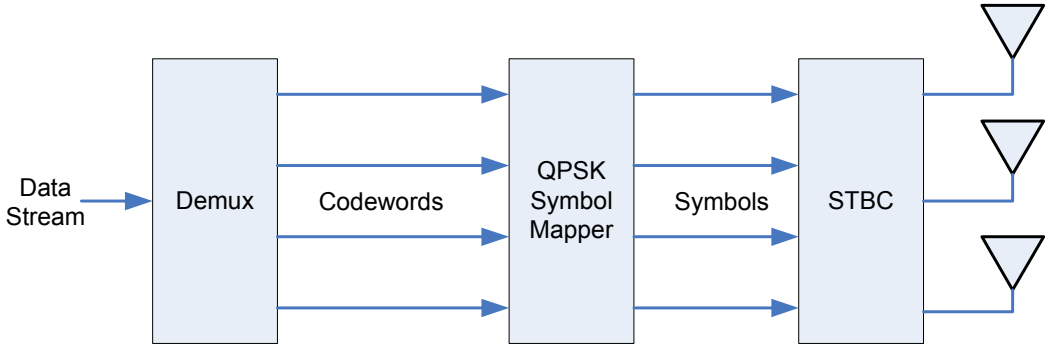

Figure 1-3 Transmitter Model

The channel model simulates a multipath fading environment with additive white Gaussian noise (AWGN) and allows attenuation and phase shift to be controlled for each signal path as shown in Figure 1-4. The channel model simulates nine signal paths between three transmit antennas and three receive antennas. Each signal path independently experiences random fading and phase delay. The channel model allows the variance of these effects to be controlled. The channel model also allows for an additional attenuation and phase delay for each signal path. These parameters allow for evaluation of the system under specific conditions such as the complete loss of one or more signal paths.

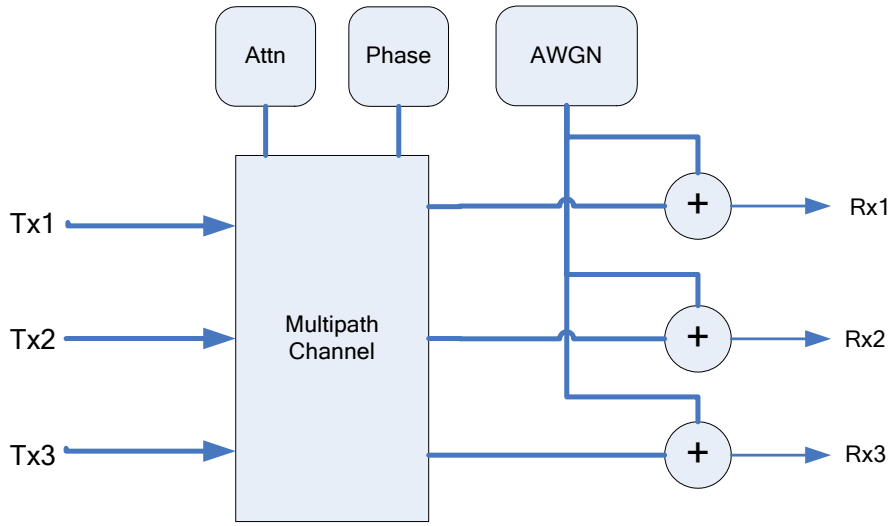

Figure 1-4 Channel Model

The channel model provides for the introduction of noise to the signals arriving at the receiver. Assuming a constant transmitter power and equal energy symbols, the variance of the noise is synonymous with the SNR. Typically, the performance of a digital communications system is evaluated in terms of the bit-error-rate (BER) verses the SNR. Thus, the model allows for the BER to be evaluated as the SNR is progressively changed.

Figure 1-5 illustrates the receiver that includes a decoder and a detector. The decoder computes estimates of the transmitted symbols using the STBC and knowledge of the channel characteristics. The receiver model assumes complete knowledge of the channel state information (CSI), that is, the decoder has perfect knowledge of the channel characteristics, including attenuation and phase shift, at every instant in time [1]. After the symbol estimates are computed, a maximum likelihood (ML) detector is used to

select the most probable symbols sent from the transmitter. The receiver then maps symbols into codewords and reconstructs the data stream.

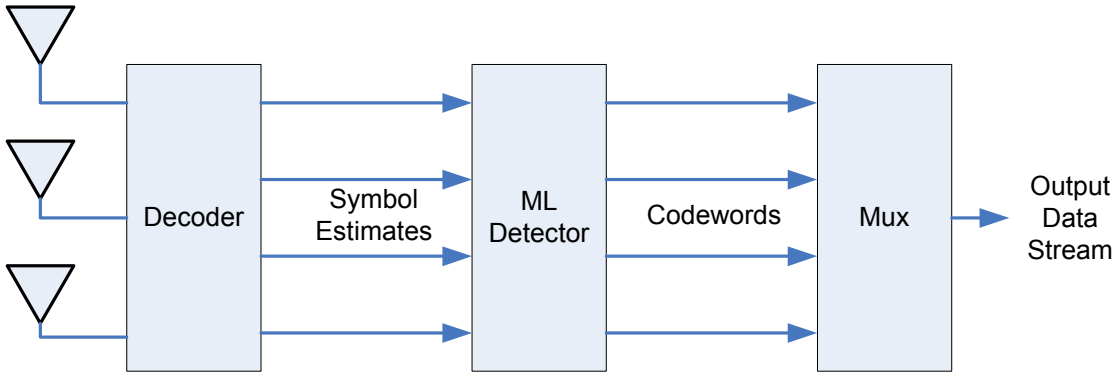

Figure 1-5 Receiver Model

The receiver model can be augmented in the future with a channel estimator so that the CSI need not be assumed. Furthermore, the detector may be replaced with another detection techniques such as maximum a posteriori probability (MAP) detection [32]. The performance of the STBC system is evaluated in terms of the BER verses the SNR under the given channel conditions. The transmitter input data bits are directly compared with the corresponding output bits from the receiver. The total bits transferred and the total bit errors detected are tallied. The BER can then be computed at any time by taking the ratio of these values.

Hardware verses Software Implementation

The inherent parallel structure of the STBC system model make it well suited for implementation in PGA hardware. Since each signal path is independent of all others, it is theoretically possible to compute all signal paths simultaneously. Thus, a parallel

computational hardware implementation can be beneficial for the evaluation of such systems considering the following:

1. In order to evaluate the system performance across changing conditions, it is necessary to compute a BER verses SNR curve for each set of channel conditions.

2. The statistical evaluation of a BER curve requires the repeated processing of a data set for a series of SNR values.

3. Low BER values, such as those achievable by MIMO systems, require the processing of a large number of bits in order to obtain sufficient precision. For example, for a BER on the order of 10^{-6} it is necessary to process at least 10^{8} data bits to achieve a precision of 1%.

Since evaluation of system performance across varying channel conditions can require a large number of computations, this research implements the system model in PGA hardware. A software based implementation is also created as a reference. The hardware implementation provides a computational speed advantage over software implementation, allowing for larger data sets and finer resolution of analysis points.

Development Tools

The software implementation of the system is written in C# using the Microsoft® Visual Studio 2008 integrated development environment (IDE). The Visual Studio IDE provides all the software tools and libraries required to write and debug the model. No specialized software libraries other than the standard .NET libraries provided with Visual

Studio 2008 are utilized to construct the model and no special effort is made to optimize the code. The model is executed on an AMD Athlon™ 64 Dual Core processor running at 2.66MHz.

The hardware implementation is primarily developed using the MATLAB/Simulink® programming environment [24] [25] in combinations with the Xilinx System Generator™ [26]. Simulink allows the construction of system models, such as the MIMO system of this dissertation research, using a graphical development environment in which functional blocks are interconnected with signal lines. Simulink solves the model at a series of time steps by evaluating each function block and propagating the results across the signal connections.

Simulink function blocks are generally equivalent to MATLAB functions. The Xilinx System Generator extends the Simulink block libraries with function blocks that can be automatically translated to Xilinx PGA hardware implementations. The System Generator libraries also include a block that allows a custom function to be defined using a subset of the MATLAB scripting language that is also automatically translated into a hardware implementation. The Simulink and System Generator blocks can be intermixed in a single model as long as special interface blocks are utilized.

Simulink supports the direct exchange of data with the MATLAB environment. This mechanism is utilized to set the model configuration before a simulation run, to monitor its progress and to retrieve the final results. The Simulink environment is also utilized to perform off-line computations such as parameter conversion and the computation of BER

from the final tallies of bits transmitted and errors, eliminating the need to implement division and transcendental functions in hardware.

System Generator supports the use of its block sets both in a software simulation and in a hardware assisted mode. In software simulation the System Generator blocks are processed just as any other block by Simulink and produce results that are equivalent to that produced by a hardware implementation. This mode can be time consuming for large models such as the MIMO model of this dissertation research, but is useful for detailed debugging.

In the hardware assisted mode, also known as "hardware-in-the-loop", a subsystem composed entirely of System Generator blocks is converted into an equivalent hardware implementation block [26]. The hardware block is then used to replace the subsystem in the Simulink model. At the start of a simulation, the hardware implementation is downloaded to target hardware device for execution. In the course of the simulation, data is passed to and from the hardware subsystem for processing. The hardware subsystem executes with the speed and parallel execution capabilities of the PGA target device greatly improving the performance of the simulation.

The PGA target device for the implementation is the Xilinx® Vertex-4 SX FPGA. The Vertex-4 is the fourth generation in the Xilinx Vertex family and features XtremeDSP™ Slices that can perform 18x18 bit multiplication and additions, Block RAM for lookup tables and intermediate data storage and configurable logic blocks (CLB) for state machine and control logic [15]. These PGA resources are well suited to implement the

complex number computations and random number generation required by the STBC model.

This research utilizes the Xilinx ML402 evaluation platform for the Vertex-4 SX FPGA. The ML402 platform is configured with the XC4VSX35 Vertex-4 FPGA and also provides an Ethernet port for downloading programming code and transferring data to and from the simulation [22].

Problem Statement

This dissertation research demonstrates a method for evaluation of STBC based MIMO communications systems under controlled channel condition by constructing a simulation model allowing individual control of signal path conditions in terms of attenuation and phase delay. The simulation model is based on a third order STBC and is implemented both in sequential software and as a Simulink model with a Xilinx FPGA hardware accelerator. Both model implementations are validated by confirming the predicted performance results for the STBC. The software implementation is utilized primarily as a reference of model performance, while the hardware implementation is used to demonstrate how MIMO system performance degradation can be examined under controlled channel conditions.

Additionally, the model itself provides a platform for future research in digital communications theory by allowing substitution of the base modulation, the STBC, the base channel model, the number of signal paths, the detector and the decoder.

Furthermore, the model is extensible so that components such as channel estimation can be in incorporated.

Outline of the Dissertation

This chapter provides an introduction to STBC systems and discusses the applicability of a PGA hardware assisted system model. Chapter 2 describes the historical precedent of the development of MIMO focusing on STBC systems. Chapter 3 provides the digital communications theory behind the system model constructed. Chapter 4 presents the general design of the system model implementation that applies to both the software and hardware models. Chapter 5 discusses details specific to the software model implementation. Chapter 6 discusses details specific to the hardware model implementation. Chapter 7 describes the method for validating the model. Finally, Chapter 8 discusses the results obtained from the simulation models, compares their performance and indicates the significance and extension of this dissertation research.

CHAPTER 2
HISTORICAL BACKGROUND

Multiple-input-multiple-out (MIMO) digital communications systems grew out of the desire to improve the performance of cellular wireless service [27][29]. Investigations leading to MIMO began with an examination of antenna diversity to overcome the effects of multipath encountered by mobile wireless users. Multipath results when the signal sent from a transmitter travels over several different spatial paths to a receiver as illustrated in Figure 2-1. Each path may impose a different attenuation, phase shift and time delay on the signal. At the receiver, the signals arriving via the different paths interfere with each other degrading the reception of the intended signal. In addition, the effect of each transmission path on the signal can vary with time.

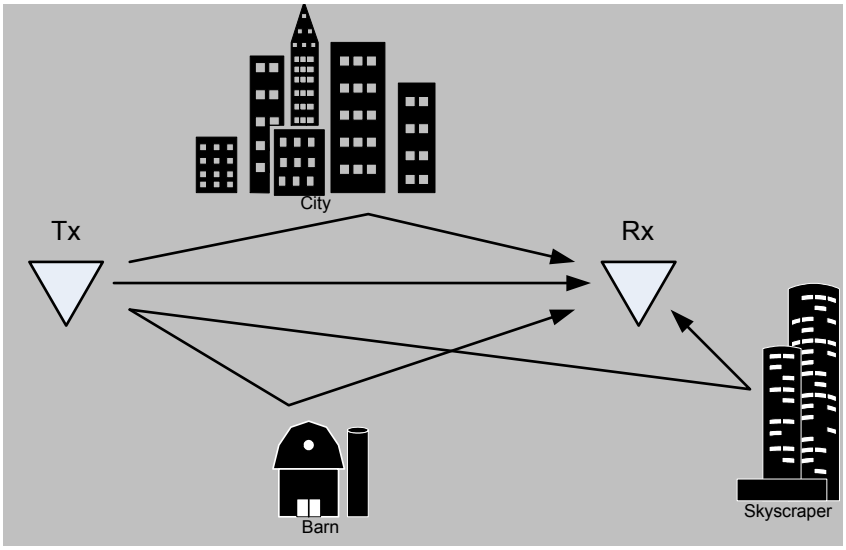

Figure 2-1 Multipath

Spatial diversity attempts to mitigate some of the multipath effect by utilizing two or more transmit or receive antennas. If two receive antennas can be placed far enough apart to ensure that the arriving signals are uncorrelated, each will see the signal affected differently by the multipath. Antenna diversity then attempts to utilize the best signal received, by switching between antennas or by combining the signals from two or more antennas [27]. However, unlike MIMO, antenna diversity does not utilize the multipath itself for a performance advantage.

The next step in the evolution of MIMO systems came from reversing the previous concept. If a signal is transmitted from several antennas, spaced far enough apart to ensure different signal paths, then the advantages of spatial diversity should be achievable while allowing the receiver to have only one antenna. This arrangement is somewhat more practical to implement since it is often easier to accommodate more antennas at a stationary transmission site than on a mobile receiver.

Such a system was presented by Wittneben [28] for applications, such as police radio, where a region is blanketed by identical signals transmitted from several sites at the same time. As illustrated in Figure 2-2, the transmit sites are chosen so that the signals cover different areas, but overlap at the boundaries. Wittneben proposed that reception could be improved in the overlapping regions by the use of optimally weighted combining. He proposed that optimal weighting be applied to the transmitted signals to improve combining at the receiver. Although Wittneben showed some improved performance with this method, this was still not a MIMO system since the weighting was applied only

to improve the combining of the signals at the receiver and not to utilize the additional capacity of the multiple spatial signal paths.

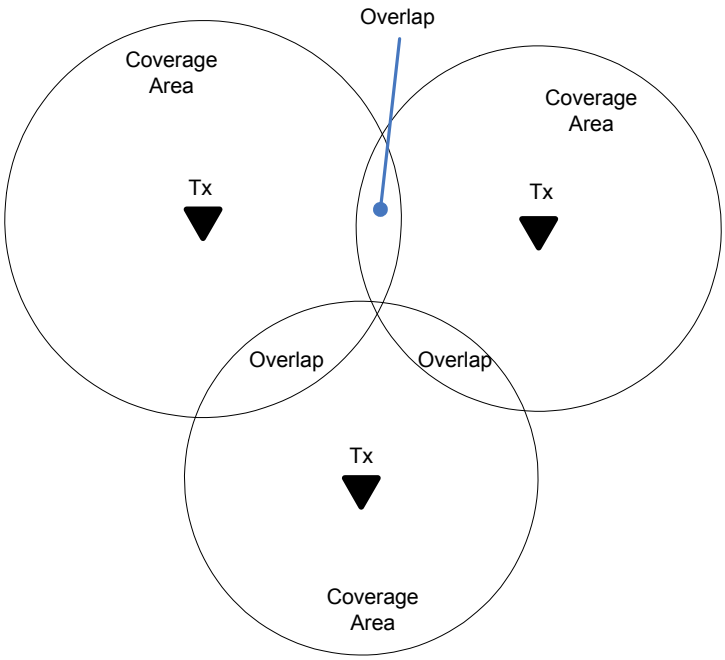

Figure 2-2 Simulcast Signal Coverage

Winters [29] took the next step in the evolution of MIMO systems and showed that the capacity of a wireless system could be increased if the signals from multiple receive antennas could be combined using an optimally weighted linear combination. He analyzed the case were several mobile users, each having one antenna, communicate to a base station having eight antennas. Winters showed that, with optimized combining of the received signals and assuming Rayleigh fading channels, interference to any one user due to the others could be canceled out. That is, he showed that as long as the number of base station antennas is greater than the number of mobile users, each user's bit error rate is the same as it would be if there where no other users. Thus, the capacity of the

wireless system to support multiple mobile users could be improved by increasing the number of base station antennas.

Seshadri and Winters [30] proposed two schemes that take advantage of the spatial diversity afforded by transmitting from multiple antennas. Each scheme involved encoding blocks of information bits into symbols. In the first scheme, a set of sequential symbols is transmitted simultaneously, with each symbol sent from a different antenna. In the second scheme, each antenna transmits the same symbol sequence, but introduces a delay of one symbol, such that the first symbol is transmitted from the first antenna during the first symbol time, from the second antenna during the second symbol time and so forth. They also considered Rayleigh fading and multipath effects on the ability to correctly decode the data, and showed that these two simple schemes provided better BER verses SNR performance than an uncoded transmission between single antennas.

Telatar [4] derived the information capacity C of a single user MIMO transmission over Rayleigh fading channels. If the channel introduces AWGN of zero mean and noise power of $P_N = 1$, then the total transmitter power P_T is synonymous with the signal-to-noise ratio SNR. Consider the channel capacity of a system employing m transmit antennas and n receive antennas. The total transmitter power P_T is distributed among the m transmit antennas. At the receiver, the signal from each of the n receive antennas is combined. All antennas are assumed to be far enough apart so that the signals are essentially uncorrelated. Telatar found three interesting results.

1. Receive Diversity: For $m = 1$, $C \rightarrow C = \log (1 + n\ SNR)$ as $n \rightarrow \infty$.

2. Transmit Diversity: For $n = 1$, $C \rightarrow C = \log(1 + SNR)$ as $m \rightarrow \infty$.

3. Combined Diversity: For $n = m$, C increase approximately linearly with n or m.

At about the same time, Foschini and Gans [8] derived the same expressions for the channel capacity of MIMO channels. This work lead immediately to work by Foschini [7] in which he proposed the use of the MIMO channel capacity by multiple users. He described a method by which a primitive data stream is broken into a number of sub-streams to be transmitted simultaneously using a MIMO coding. At the receiver, each data stream is recovered using a recursive method that later became known as the Diagonal Bell Laboratories Layered Space-Time (D-BLAST) method. Afterward, the more computationally efficient Vertical BLAST (V-BLAST) method was proposed [31] and has become one of the main areas of MIMO research.

In the BLAST method, the data streams are transmitted simultaneously from a set of antennas. The matrix $\boldsymbol{H}$ of channel characteristics for the signal paths is assumed to be known and is usually determined using a training sequence. At the receiver, the data streams are detected one-at-a-time. Given N data streams, the first is detected in the presence of $N - 1$ interferers. This is accomplished by computing a weighting vector $\boldsymbol{w}$ based on $\boldsymbol{H}$ that nulls the contributions from the interferers. Once the first data stream is obtained, it can be subtracted from the received signals effectively canceling its interference with the remaining data streams. The same process is then repeated for the second data stream, which only need be detected among the remaining $N - 2$ interferers.

Although Foschini provided a MIMO architecture with BLAST, he did not propose a method of coding. His work, however, was followed by a number of papers that presented criteria for the design and optimum performance of space-time codes [11] [12] [13]. This work concluded that trellis codes provided better space-time codes than block codes in terms of diversity gain, code rate and computational complexity.

Alamouti [9] proposed one of the first space-time block codes (STBC) that used two transmit antennas, two receive antennas and maximum-ratio receiver combining (MRRC). He showed marked improvement in terms of BER verses SNR over a 1-dimensional transmission. Alamouti's method, although suboptimal compared to trellis codes, lowered decoding complexity over trellis decoding by requiring only linear processing at the receiver. He showed that STBC is a viable alternative to trellis codes for MIMO.

Following the introduction of Alamouti's method, Tarokh, Jararhani and Calderbank [1] [10] showed how optimal STBCs could be designed. They described how STBCs could yield the performance achieved by Alamouti. They presented criteria for the design of optimal STBCs and determined the performance of such codes. Using these criteria, they derived several new STBCs utilizing three and four antennas. One of these codes is the basis for the models implemented by this dissertation research.

CHAPTER 3
THEORY OF OPERATION

This research constructs a MIMO system model employing a three antenna rate ½ orthogonal space-time block code (STBC) as presented by Tarokh et al. [1]. As described in Chapter 1 and illustrated in Figure 1-2, the system consists of a transmitter, a multipath channel, a receiver and a BER evaluation module. This chapter provides the theory behind the model's design and operation.

Data Stream

The input data consists of a continuous stream of arbitrary 8-bits data words that are provided by a uniform pseudo-random number generator (PRNG). The data stream is assumed to have no error correction coding, although an error correction code could be added to the model. The data word is demultiplexed into 4 two-bit codewords from the codeword space $C = \{ c_0 , c_1 , c_2 , c_3 \}$ where each codeword maps to a unique two-bit sequence defined in Table 3-1.

Table 3-1 Definition of Codewords

Codeword	Bit Pattern
c_0	00
c_1	01
c_2	10
c_3	11

Base Modulation

The base modulation for the data transmission here is quadrature phase shift keying (QPSK), which provides four equal energy complex symbols from the symbol space $S = \{ s_0 , s_1 , s_2 , s_3 \}$. Each symbol from S can be decomposed into the sum of a real or in-phase (I) component and an imaginary or quadrature (Q) component as expressed by (1) [32]. Here s_{iI} and s_{iQ} are the coefficients of the in-phase and quadrature components respectively. The functions $\varphi_I(t)$ and $\varphi_Q(t)$ are known as basis functions and are given by (2) and (3) respectively, where T_S is the symbol period and f_c is the carrier frequency [32].

$$s_i = s_{iI}\varphi_I(t) + s_{iQ}\varphi_Q(t) \tag{1}$$

$$\varphi_I(t) = \sqrt{\frac{2}{T_S}}\cos(2\pi f_c t) \tag{2}$$

$$\varphi_Q(t) = \sqrt{\frac{2}{T_S}}\sin(2\pi f_c t) \tag{3}$$

Each symbol s_i is distinguished by its in-phase and quadrature coefficients. The coefficients for each symbol are given in Table 3-2, where E_s is the symbol energy and is the same for all symbols.

Table 3-2 Base Modulation Symbol Coefficients

Symbol	s_{iI}	s_{iQ}
s_0	$+\sqrt{E_s/2}$	$+\sqrt{E_s/2}$
s_1	$-\sqrt{E_s/2}$	$+\sqrt{E_s/2}$
s_2	$+\sqrt{E_s/2}$	$-\sqrt{E_s/2}$
s_3	$-\sqrt{E_s/2}$	$-\sqrt{E_s/2}$

A modulator or symbol mapper maps each codeword to a specific symbol according to Table 3-3, such that a given 8-bit data word maps to a vector of codewords that in turn map to a vector of symbols. For example, the data word 00101110_2 corresponds to the codeword vector [c_0, c_2, c_3, c_2], which in turn leads to the symbol vector [s_0, s_2, s_3, s_2].

Table 3-3 Codeword to Symbol Mapping

Codeword	Symbol
c_0	s_0
c_1	s_1
c_2	s_2
c_3	s_3

Space-Time Block Code

The model uses a three antenna rate ½ orthogonal space-time block code proposed by Tarok et al. [1]. As given in (4), the matrix **G** describes the STBC, which encodes the

transmission of a symbol vector $\mathbf{X} = [\, x_0, x_1, x_2, x_3 \,]$ over eight transmission periods. Here, the elements of **X** are taken from the symbol space *S* such that the vector **X** corresponds to a data word as described in the previous section. The encoding defines how the elements of **X** are transmitted during each of eight symbol periods or timeslots. Since, the transmission of the entire code block requires eight symbol periods and the block carries only four codewords, the code is rate ½.

$$\mathbf{G} = \begin{pmatrix} x_0 & x_1 & x_2 \\ -x_1 & x_0 & -x_3 \\ -x_2 & x_3 & x_0 \\ -x_3 & -x_2 & x_1 \\ x_0^* & x_1^* & x_2^* \\ -x_1^* & x_0^* & -x_3^* \\ -x_2^* & x_3^* & x_0^* \\ -x_3^* & -x_2^* & x_1^* \end{pmatrix} \tag{4}$$

In addition to assigning symbols to antennas, the encoding also applies the operations of negation and complex conjugation (indicated by *). These operations are applied to the basic symbols during specific timeslots. It can be observed that, for the chosen base modulation (QPSK), applying negation and/or complex conjugation to any symbol conveniently transforms that symbol into another symbol in *S*. Thus, all of these operations can be accomplished by sign inversion of the real, imaginary or both the real and imaginary parts of the complex symbol.

Each row of the code matrix **G** specifies the symbols to be transmitted during one timeslot. Here the columns of the matrix correspond to each of the transmission antennas $\{ t_0, t_1, t_2 \}$. Thus, the elements of a row determine the symbol transmitted from each antenna and the operation applied to the symbol during the timeslot. For example, during the first transmission period, the symbol x_0 is transmitted from antenna t_0, x_1 from t_1 and x_2 from t_2. During the second transmission period, $-x_1$ is transmitted from t_0, x_0 from t_1 and $-x_3$ from t_2. The process is repeated until all the rows of **G** have been sent, completing the transmission of the block.

Transmitter

A transmitter with output power P_T excites the three transmitting antennas. The transmitter power is distributed among the antennas such that the total transmitted power remains equal to P_T. Holding transmit power constant makes the resulting performance measurements directly comparable to a 1-dimensional transmission utilizing the same output power. In MIMO digital communications systems where the channel state information (CSI) is known at the transmitter, it is possible to distribute the power optimally over the antennas [5]. As considered here, the transmitter is assumed to have no knowledge of the CSI, in which case it is optimal to distribute the power evenly to each of the antennas.

The transmit antennas are assumed to be spaced far enough apart to ensure that the transmitted signals are uncorrelated, in other words, the transmit antennas do not constitute a beam forming array that delivers one signal, but instead provide multiple

independent signals [33]. The independent signals provide the spatial diversity from which the advantages of MIMO are derived.

Channel

Each transmitted signal traverses an independent fading path with channel characteristic given by (5). Here α_{ik} is the attenuation, θ_{ik} is the phase delay and h_{ik} is the channel characteristic for the path from the i^{th} transmit antenna to the k^{th} receive antenna. The multipath channel introduces Rayleigh fading so that α_{ik} is a Rayleigh distributed random variable and θ_{ik} is a uniformly distributed random variable [7].

$$h_{ik} = \alpha_{ik} \exp(j\theta_{ik}) \tag{5}$$

The channel characteristics, h_{ik} , are independent identically distributed (iid) time varying complex random variables that form the matrix **H** according to (6). The matrix **H** is assumed to be quasi-static, that is **H** is assumed to be constant for the time required to transmit a code block.

$$\mathbf{H} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \tag{6}$$

This dissertation research extends the definition of h_{ik} as in (7). Two new variables have been introduced, A_{ik} and Φ_{ik}, which are chosen to establish the conditions of the simulation and remain constant throughout the evaluation of a BER curve. The first variable, A_{ik} provides a specific attenuation factor for a transmission path. The second variable, Φ_{ik} provides a bias to the phase delay.

$$h_{ik} = \mathrm{A}_{ik}\alpha_{ik}\exp(j[\theta_{ik} + \Phi_{ik}]) \quad (7)$$

In addition to the transfer characteristics of each signal path, complex additive Gaussian white noise (AWGN) n_k is introduced at each receive antenna. In the model, n_k is generated with a mean $\mu=0$ and a variance $\sigma^2=1$. The variance is then scaled to obtain the required SNR.

Receiver

The receiver utilizes linear combining of the signals obtained from the three receive antennas. Here r_k is the signal observed at the k^{th} receive antenna during a single transmission period. The observed signal at each antenna is then the sum of the signals arriving from each of the transmitting antennas plus noise, as in (8).

$$r_k = \sum_{i=1}^{m} h_{ik} s_i + n_k \quad (8)$$

Decoding of the STBC requires that the observed signals at each receive antenna be collected for all of the periods comprising the transmission of a code block. Here $r_k^{(\tau)}$ is the signal observed at the k^{th} antenna during the τ^{th} transmission period. In addition, assume that the receiver has perfect knowledge of the channel transfer characteristics **H** during the block transmission time. The receiver applies linear combining to obtain estimates of the transmitted symbols, followed by maximum likelihood detection as described in the following sections.

Decoding

The STBC utilized by the simulation model of this dissertation research given by (4) transfers the symbol vector **X**. The decoder utilizes knowledge of the STBC and the channel transfer characteristics **H** to arrive at an estimate of the elements of **X**. This method was developed by Tarok et al. [10], where (9) through (12) give the estimates.

$$\tilde{x}_0 = \sum_{k=0}^{2}\left[r_k^{(0)} h_{0k}^* + r_k^{(1)} h_{1k}^* + r_k^{(2)} h_{2k}^* + \left(r_k^{(4)}\right)^* h_{0k} + \left(r_k^{(5)}\right)^* h_{1k} + \left(r_k^{(6)}\right)^* h_{2k} \right] \tag{9}$$

$$\tilde{x}_1 = \sum_{k=0}^{2}\left[r_k^{(0)} h_{1k}^* - r_k^{(1)} h_{0k}^* + r_k^{(3)} h_{2k}^* + \left(r_k^{(4)}\right)^* h_{1k} - \left(r_k^{(5)}\right)^* h_{0k} + \left(r_k^{(7)}\right)^* h_{2k} \right] \tag{10}$$

$$\tilde{x}_2 = \sum_{k=0}^{2}\left[r_k^{(0)} h_{2k}^* - r_k^{(2)} h_{0k}^* - r_k^{(3)} h_{1k}^* + \left(r_k^{(4)}\right)^* h_{2k} - \left(r_k^{(6)}\right)^* h_{0k} - \left(r_k^{(7)}\right)^* h_{1k} \right] \tag{11}$$

$$\tilde{x}_3 = \sum_{k=0}^{2}\left[- r_k^{(1)} h_{2k}^* + r_k^{(2)} h_{1k}^* - r_k^{(3)} h_{0k}^* - \left(r_k^{(5)}\right)^* h_{2k} + \left(r_k^{(6)}\right)^* h_{1k} - \left(r_k^{(7)}\right)^* h_{0k} \right] \tag{12}$$

Detection and Multiplexing

The receiver uses maximum likelihood (ML) detection to recover the elements of **X** based on the estimates computed by the decoder [1] [10]. This is accomplished by computing the Euclidian distance between each estimate $\tilde{x}_n$ and each of the symbols in S as expressed in (13). The symbols corresponding to the minimum distances are used to form the detected symbol vector $\mathbf{Y} = [\, y_0, y_1, y_2, y_3 \,]$, where $y_n = s_i$ such that s_i has the

smallest distance to $\tilde{x}_n$. The elements of **Y** are then mapped back to codewords and these codewords are multiplexed to recover the data word.

$$\text{for } n = 0,1,2,3 \qquad d(\tilde{x}_n) = \min_{i=0,1,2,3} \left(\left| \tilde{x}_n - s_i \right|^2 \right) \tag{13}$$

BER Evaluation

The bit error rate of the recovered data stream is determined by direct comparison to the original data stream. In the simulation model, the BER evaluation maintains running totals of the number of bit errors and the total number of bits processed. The quantitative value for BER is then computed by taking the ratio the total errors to the total bits processed.

CHAPTER 4
GENERAL IMPLEMENATION

This chapter describes the architecture and operation of the simulation models and provides details of the functional blocks. The model is implemented both as a C# program on a Microsoft® Windows based desktop computer and as hardware-in-the-loop on a Xilinx Vertex-4 FPGA. Both implementations have architecture as illustrated in Figure 4-1. A Host environment provides the configuration and control for the simulation model. Prior to each simulation, the Host sets the parameters such as SNR, path attenuations, and the number of bits to process. The Host environment controls the starting and stopping of the simulation and monitors its progress. At the completion of the simulation, the total bits and error counts are read and used to compute BER. The Host environment can also automatically process a series of simulations with incrementally changing conditions so that BER verses SNR curves may be evaluated.

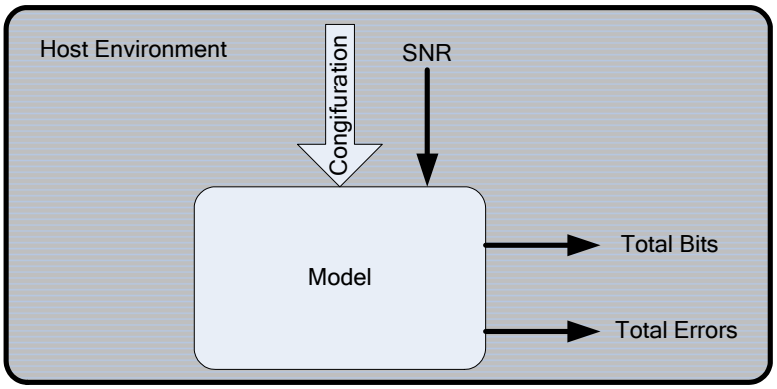

Figure 4-1 Overview of Simulation Model

Inputs and Outputs

The model is designed to compute BER verses SNR curve for the STBC system under specific conditions. These conditions are provided to the model at the start of a simulation and remain constant throughout. The conditions include the mean μ_m and variance μ_m of the multipath, attenuation and phase delays for the individual signal paths, the SNR and the total number of bits to process. Data is generated within the simulation model and accumulators track the total number of bits processed as well as the total bit errors detected. The simulation is complete when the total number of bits processed equals or exceeds the quantity specified for the simulation

Multipath

The multipath characteristics are defined by the mean μ_m and variance μ_m of the fading. These are real valued constants that are supplied once prior to the start of the simulation. These parameters are utilized in the generation of random variables α_{ik} and θ_{ik} utilized in (7). A single value of mean and variance define the fading conditions for all channel paths.

Path Attenuation and Delay

The path attenuation A_{ik} and phase delay Φ_{ik} characteristics define the attenuation and bias phase delay for each communications channel signal path. Nine attenuation and delay pairs define the conditions for the nine signal paths of the communications channel. Each attenuation and delay pair is expressed as a single complex value z_{ik} in rectangular form as given by (14).

$$z_{ik} = A_{ik} \exp(j\Phi_{ik}) = \mathrm{Re}(z_{ik}) + j\,\mathrm{Im}(z_{ik}) \quad (14)$$

Signal-to-Noise Ratio

The model generates channel noise from Gaussian pseudorandom number sources having zero mean and unit variance. A required noise variance is computed from the specified SNR that is then used to scale the output of the Gaussian sources.

Data

Eight-bit data words are generated by a pseudorandom number generator (PRNG) within the simulation model. Data words are uniformly distributed over the range [0, 255]. Each data word is processed through the model and the output is compared to the input to determine bit errors.

Bit-Error Counters

The total number of bits and the total number of bit errors are accumulated within the model for each simulation. The accumulators are sampled periodically during the simulation in order to monitor progress. At the completion of the simulation, the content of the accumulators are used by the Host environment to compute the BER value.

Architecture

Figure 4-2 illustrates the architecture of the simulation model. Each block represents a functional element. Data input to the demultplexer block (Demux) is split into two-bit segments and used to form the codeword vector to be transmitted. The codeword vector is mapped to a symbol vector **X** by the Symbol Mapper as described in Chapter 3. The symbol vector is input to the transmitter block (Tx) that utilizes the STBC to encode the

symbols and pass them through the channel. The noise vector **n** and the channel path characteristics matrix **H** are generated using pseudorandom number generators. The channel (Chn) applies the signal path characteristics and noise to the symbols.

The channel output is the receive vector **R,** which is utilized by the Decoder (Decode) to compute symbol estimates. The detector (Detect) chooses the most likely transmitted codewords by selecting the symbols having the minimum Euclidian distance from the estimates. Finally, the multiplexer block (Mux) reconstructs the output data word. The output data words are then supplied to the BER block for evaluation. A first-in first-out (FIFO) buffer compensates the reference data path for delay introduced by the channel processing.

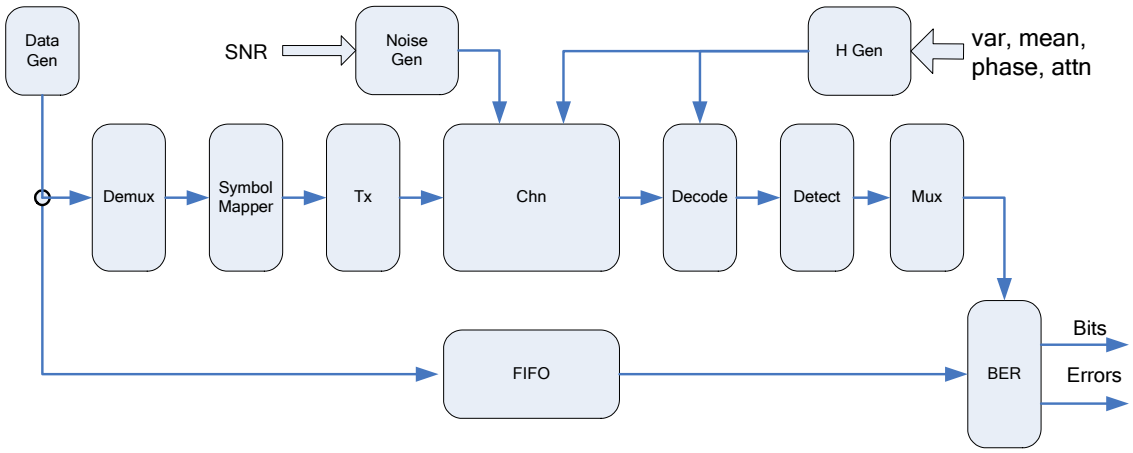

Figure 4-2 Simulation Model System Architecture

Demultiplexer

The Demux block converts the eight-bit input data word *D* into the codeword vector **c** = [$\gamma_0, \gamma_1, \gamma_2, \gamma_3$]. This is accomplished by assigning two-bit segments of the data word *D*

to the values of γ_i according to Table 4-1, such that each γ_i takes on the one of the values from the codeword space *C*.

Table 4-1 Input Data Bit to Codeword Mapping

Bit Position	Codeword
0,1	γ_0
2,3	γ_1
4,5	γ_2
6,7	γ_3

Symbol Mapper

The symbol mapper converts the codeword vector **c** to the symbol vector **X**. Each symbol is represented as a complex value with energy $E_s = 1$. The energy of each symbol is then divided by the number of transmitting antennas used for the simulation so that the total transmitted power is $P_T = 1$ and is evenly distributed to the antennas.

Transmitter

The transmitter encodes the symbols of **X** according to the STBC and sends them through the channel. The STBC defines how the four symbols of **X** are transferred over eight time slots as explained in Chapter 3. The STBC requires the operations negation and complex conjugation that are easily accomplished by sign inversion of the real, imaginary or both the real and imaginary parts of the complex symbol value as given by (15) through (18).

$$x_i = \mathrm{Re}(x_i) + j\,\mathrm{Im}(x_i) \tag{15}$$

$$-x_i = -\mathrm{Re}(x_i) - j\,\mathrm{Im}(x_i) \tag{16}$$

$$x_i^* = \mathrm{Re}(x_i) - j\,\mathrm{Im}(x_i) \tag{17}$$

$$-x_i^* = -\mathrm{Re}(x_i) + j\,\mathrm{Im}(x_i) \tag{18}$$

Channel

The channel applies the path characteristics and noise to the transmitted symbols and computes the signal arriving at each receive antenna for each symbol time. Path characteristics are provided by the H-Generator (HGen) and remain constant for a block time. A noise vector **n** is supplied by the Noise generator (Noise Gen) every symbol time. The channel block computes (8) for each receive antenna to produce the receive vector **R**. A new receive vector is provided to the decoder by the channel block during each time slot.

Noise Generator

The noise generator (Noise Gen) is a set of three Gaussian pseudorandom number generators (GPNG) that supply three complex white noise values to the channel block during each symbol time. One noise value is added at each of the three receive antennas. The noise generator is of zero mean and unit variance. The basic PRNG produces uniformly distributed random values that are transformed using the inversion method to produce a Gaussian distribution [50].

The noise generator variance is scaled to produce the required SNR for the simulation. The required noise variance is computed assuming that a single symbol carries all of the transmitted energy, $E_s=1$, to allow for comparison of the MIMO simulation results with that of a single channel QPSK transmission [10].

Path Characteristic H-Generator

The H-generator (HGen) creates the nine element path characteristic matrix **H**. Each element h_{ik} is computed based on (7), which can be divided into two parts, the random multipath characteristic Ψ_{ik} and the path attenuation z_{ik} as given by (19). The attenuation and phase delay z_{ik} are provided by the Host prior to execution of the simulation and remain constant throughout. The multipath characteristic Ψ_{ik} is generated for each coded block of symbols.

$$h_{ik} = (z_{ik})(\psi_{ik}) = \left[A_{ik} \exp(j\phi_{ik})\right]\left[\alpha_{ik} \exp(j\theta_{ik})\right] \tag{19}$$

Although the magnitude of Ψ_{ik} is Rayleigh distributed and the phase of Ψ_{ik} is uniformly distributed, in rectangular form the real and imaginary parts of Ψ_{ik} are both Gaussian distributed [32]. Thus Ψ_{ik} is created by generating a complex pair of Gaussian random variables using the same technique used to generate the channel noise. However in this case, both the mean and the variance of the GPNG are adjusted to provide the multipath mean μ_m and variance σ_m^2 specified for the simulation. The value for h_{ik} is then computed by multiplying Ψ_{ik} by z_{ik} for each signal path.

Decoder

The Decoder (Decode) estimates the elements of transmitted symbol vector **X** using (9) through (12). The Decoder utilizes knowledge of the channel path characteristics **H** and receive vector **R** from the channel. The Decoder requires eight consecutive receive vectors to produce the symbol vector estimate. One receive vector is obtained from the channel during each time slot. Each receive vector is utilized immediately by the Decoder to compute all terms of (9), (10), (11) and (12) in which it is involved. The terms are then accumulated for each of the estimated symbols. After eight symbol times the estimates are then ready for the detector.

Detector

The Detector (Detect) selects the maximum likelihood symbol vector **Y** using the minimum distance computation of (13). The detector converts **Y** into the corresponding codeword vector $\tilde{\mathbf{c}}$.

Multiplexer

The Multiplexer (Mux) reconstructs the transmitted data word from the recovered codeword vector $\tilde{\mathbf{c}}$. The two-bit codewords are mapped back to an eight-bit data word according to Table 4-1. The output data word is then available for BER evaluation and can be read by the Host system.

Bit Errors

The BER block counts the total number of bits processed through the system model and the number of resulting bit errors. Bit errors are detected by direct comparison of the

input data word and the corresponding output data word. A first-in first-out (FIFO) buffer holds input data words to compensate for the system's processing delay.

CHAPTER 5 SOFTWARE MODEL

This chapter describes the software based implementation of the MIMO system model used in this dissertation research. The MIMO system model described in the pervious chapters was implemented here using an object oriented sequential programming approach. The software is written as a Microsoft Windows® application that is intended to execute on a desktop or laptop computer.

The software based model has several purposes. First, the software implementation serves as a test bed for the MIMO system model design. The software development tools allow for detailed debugging and quick turnaround of design changes. The software based model relies on standard libraries to avoid problems that may be introduced when implementing fundamental algorithms as programmable gate array (PGA) hardware. The software based model also uses floating-point number representation to avoid the limitations on the range and precision of values inherent to the PGA implementation.

Second, the software based model is used to confirm results obtained by the PGA hardware based model. The use of standard libraries and floating point numbers in the software implementation provides independently derived results that are free of certain limitations inherent to the hardware. In addition, beyond the validation cases, comparison of results from the two implementations allows the results of both to be confirmed.

Finally, the software implementation provides a performance reference to which the hardware implementation is compared. Performance, in this dissertation research, is

defined by the time required to produce a BER verse SNR analysis for a given set of conditions and a number of points. The execution time required by the software implementation is compared to the execution time required by the PGA hardware implementation.

The remainder of this chapter describes the software implementation of the MIMO system model, its general operation and relevant details. This implementation generally follows the design of Chapter 3 and Chapter 4. The implementation of standard functional blocks is straight forward and details are not provided. Details are given where the implementation is not evident or can have significant impact on results.

Development Tools

The software based model is written in the C# programming language using the Microsoft Visual Studio 2008 integrated development environment (IDE). No special effort is made to optimize the software execution other than that general compiler optimization is enabled. No special libraries are utilized other than those provided by Microsoft .NET 3.5 Framework and HydeSoft Computing DPlot. The DPlot libraries only support the post-plotting of analysis curves and are not utilized during simulation. The .NET 3.5 Framework provides a Math class that is utilized by this model. The Math class provides square root, trigonometric and exponential functions based on IEEE 754 standard 64-bit floating-point number representation [44].

Interface

The software model provides a graphical user interface (GUI) that allows setting the simulation conditions, controlling the simulation process, monitoring the simulation progress, and viewing the simulation results. The GUI is split into two panes as shown in Figure 5-1. The left pane is fixed and shows the overall simulation conditions. The right pane has two tabbed pages. The first tabbed page, shown in Figure 5-1, allows the antenna configuration to be defined and allows the attenuation and delay of each path to be specified. The second tabbed page, shown in Figure 5-2, allows control of simulation, monitors the simulation progress and displays the results. The individual simulation controls are described in the following sections.

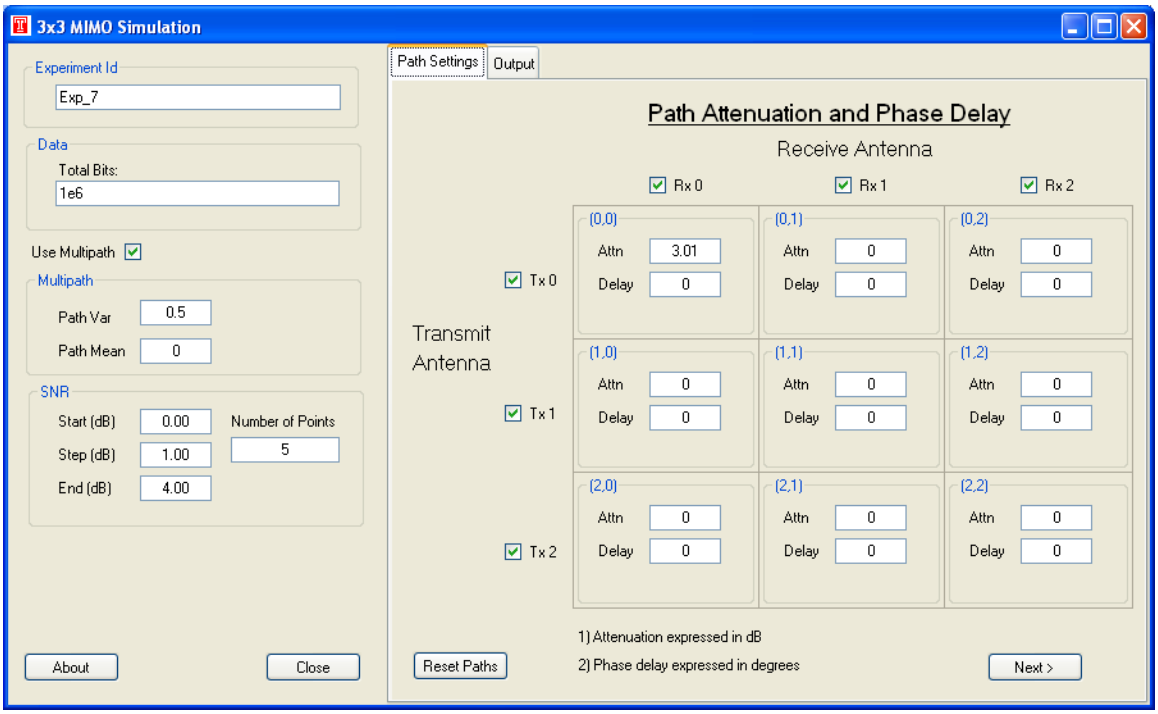

Figure 5-1 Software Based MIMO Model Configuration Page

Experiment Identifier

The experiment identifier (Experiment Id) allows for a unique name to be given to each simulation. The name is used to identify the specific results on the display and in the results file.

Data Bits to Process

The Total Bits define the minimum number of bits to process during the simulation. Each data point is evaluated by simulating no less than defined number of bits.

Multipath Parameters

Multipath can be enabled or disabled for each simulation. Disabling multipath allows the QPSK validation case to be evaluated. When multipath is enabled, the mean and variance of the random component of the channel characteristic may be set.

Signal to Noise Ratio

SNR values for the simulation are defined by a starting value, an ending value and a step value or count. SNR values are expressed in dB and are always evenly distributed over the defined range and include the start and end values.

Path Characteristics

Path Characteristics are defined by the antenna configuration and the attenuation and delay for each transmission path. Transmit and receive antennas used in the simulation are selected using the check boxes. Any combination of transmit and receive antennas is possible. Total transmit power is automatically distributed among the selected

transmission antennas. Unselected receive antennas provide no input to the decoder, but the decoder operation is not changed.

Selection of the antennas also defines the channel signal paths for the simulation. Path attenuation in dB and phase delay in degrees may be specified for each available signal path. When multipath is enabled, each signal path will also have a random component so that the resulting path characteristic h_{ik} will be as define by (19).

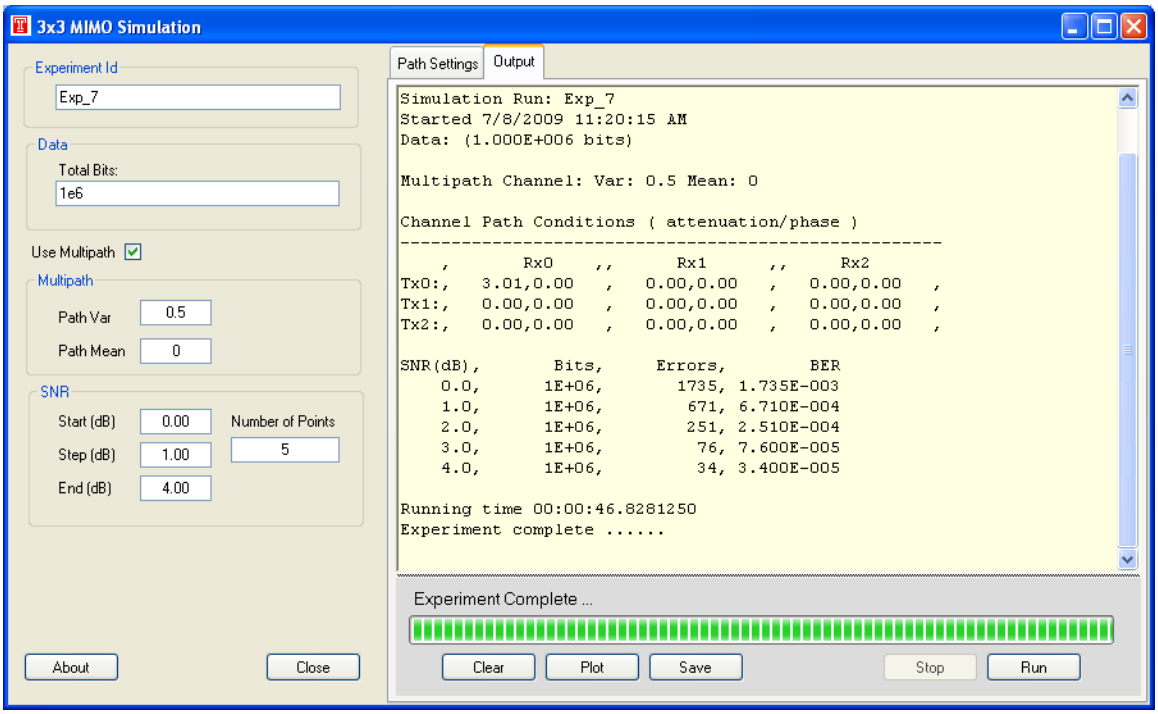

Figure 5-2 Software Based MIMO Model Results Page

Simulation Control

The simulation is controlled with buttons located at the bottom of the results page as shown in Figure 5-2. The Run button starts the simulation, while the Stop button allows

termination at any time. The relative completion status is displayed graphically with the progress bar. Intermediate results are displayed just above the progress bar and are updated periodically throughout the simulation.

Results

The simulation results are displayed in the text box just above the progress bar area shown in Figure 5-2. The results start with a header indicating the conditions of the simulation and include the experiment identifier, time and date of the simulation, the number of data bits to process at each SNR, the antenna configuration and the path characteristics. The header is followed by a table of results obtained at each SNR. The table lists the SNR, actual number of bits processed, the number of bit errors detected and the computed BER. The results table is followed by a time value indicating the time required to compute the simulation.

The simulation results may be saved to a comma separated variable (csv) formatted file that can be imported to a spreadsheet or into MATLAB for analysis. The results may also be plotted as shown in Figure 5-3. The plot allows for two or more BER verses SNR analysis curves to be shown together for comparison.

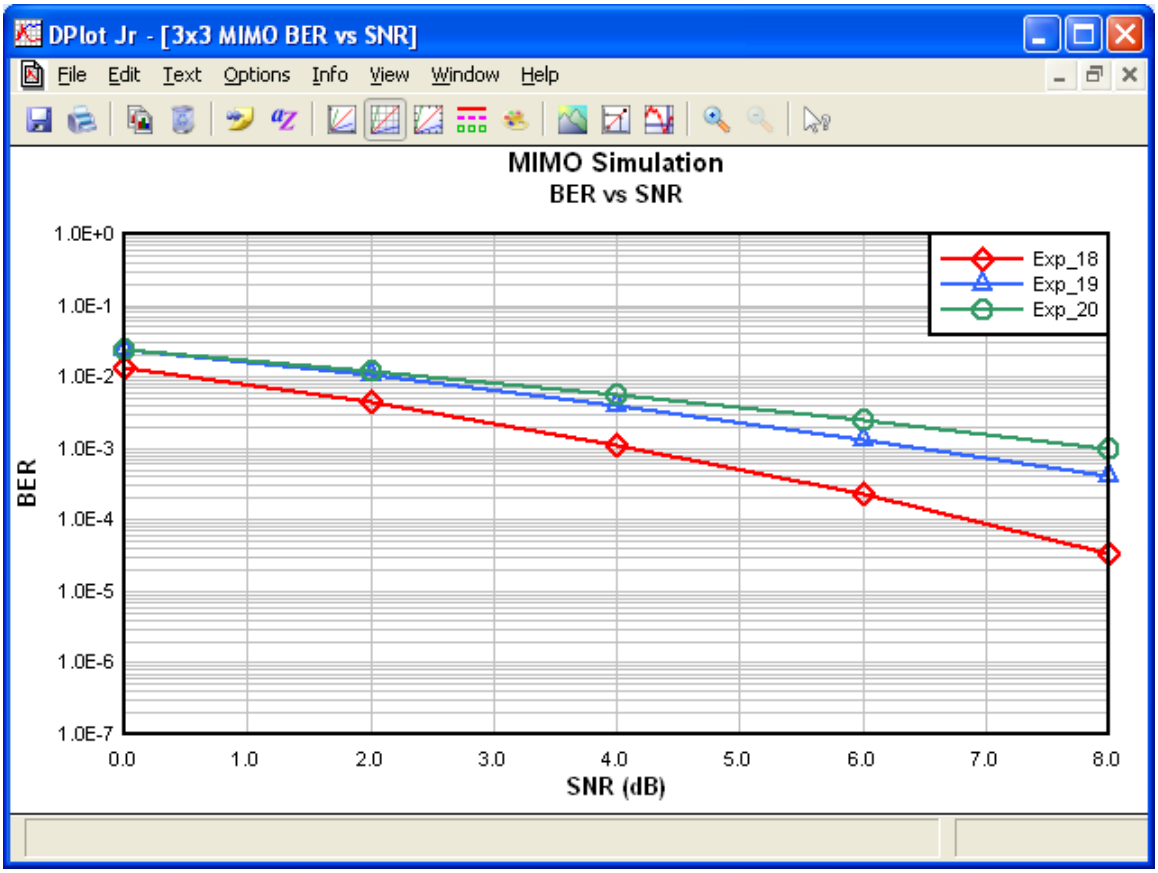

Figure 5-3 Software Based MIMO Model Results Plot

Operation

The Run button initiates the simulation using the conditions defined through the interface. For each SNR value, the simulation model is executed until the required number of data bits has been processed. A BER value is then computed based on the number of errors detected and the actual number of bits processed. The SNR, BER, bits processed and bit errors are recorded and the entire process is repeated using the next value of SNR. The simulation ends when all of SNR points have been evaluated. The simulation time is the real elapsed time from the start to completion.

For each SNR point, a core execution sequence implements the MIMO system model as described in Chapters 3 and Chapter 4. This portion of the software consists of two nested loops. The outer loop is executed once for each data byte (8-bits) processed. An inner loop is executed once for each of the 8 time slots defined by the STBC.

The outer loop has the following steps that map directly to the model of Figure 4-2.

1. (Data Gen) Generate a new random data word *D*.
2. (Demux) Covert the data word into codeword vector **c**.
3. (Symbol Mapper) Map the codeword vector to a base transmission symbol vector **X**.
4. (H Gen) Generate an instance of the channel characteristic matrix **H**.
5. Process the inner loop to produce symbol estimates.
6. (Detect) Select the maximum likelihood symbol vector **Y** and map it to the recovered codeword vector $\tilde{\mathbf{c}}$.
7. (Mux) Reconstruct the data word from recovered codeword vector.
8. (BER) Evaluate bit errors and accumulate totals

The inner loop has the following steps that map directly to the model of Figure 4-2.

1. (Tx) Encode symbols for transmission during each time slot according **G**

2. (Noise Gen) Generate an instance of the channel noise vector **n**.

3. (Chn) Compute the receive vector **R** for the time slot.

4. (Decode) Incrementally perform decoding of the received symbols.

Software Model Implementation

The following sections provide additional details on the implementation of significant sections of the software based MIMO model.

Gaussian Random Number Sources

The generation of the channel noise and the channel path characteristics all require independent Gaussian random numbers. All random numbers in this model are generated using the .NET Random class [45]. The Random class produces uniformly distributed random numbers in the range of 0.0 to 1.0. The required Gaussian distributed random numbers are derived from the values generated by the Random class using the inverse transform method [43].

The inverse transform method produces a random variable X with a desired cumulative probability distribution function (cdf), in this case Gaussian, from a uniformly distributed random variable Y by applying the a transformation function according to (20).

$$X = F^{-1}(Y) \tag{20}$$

Here the transformation function $F^{-1}(Y)$ is the inverse of the Gaussian cdf. The Gaussian cdf is given by (21) for zero mean and unit variance.

$$F(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} \exp\left(-\frac{u^2}{2}\right) du \tag{21}$$

Although (21) is difficult to invert analytically, the goal may be achieved by utilizing an identity that makes use of the inverse of complimentary error function *erfc*$^{-1}(\cdot)$ [46] according to (22).

$$F^{-1}(y) = -\sqrt{2} \bullet erfc^{-1}(2y) \tag{22}$$

Furthermore, by applying the identities $erfc^{-1}(y) = erf^{-1}(1-y)$ and $erf^{-1}(-y) = -erf^{-1}(y)$ where *erf*$^{-1}$*(y)* is the inverse error function, $F^{-1}(y)$ may be expressed as in (23).

$$F^{-1}(y) = \sqrt{2} \bullet erf^{-1}(2y-1) \tag{23}$$

The inverse error function can be approximated by series expansion. In this dissertation research, the approximation function of (24) proposed by Winitzki [41][42] is utilized since it is easily implemented. This method relies only on the mathematical functions available in the .NET Math class [47] and produced good results during for the validation test cases.

$$erf^{-1}(y) \approx \left[-\frac{2}{0.147\pi} - \frac{\ln(1-y^2)}{2} + \sqrt{\left(\frac{2}{0.147\pi} + \frac{\ln(1-y^2)}{2}\right) - \frac{\ln(1-y^2)}{0.147}} \right]^{1/2} \tag{24}$$

MIMO Decoding

The MIMO decoding is performed using an incremental approach as described in [48]. Following this approach (9) through (12) are first manipulated into (25) through (28), where each estimate is a series of summation terms that depend only on the symbols received during a single time slot.

$$\begin{aligned}\tilde{x}_0 = &\sum_{k=0}^{2} r_k^{(0)} h_{0k}^* + \sum_{k=0}^{2} r_k^{(1)} h_{1k}^* + \sum_{k=0}^{2} r_k^{(2)} h_{2k}^* \\ &+ \sum_{k=0}^{2} \left(r_k^{(4)}\right)^* h_{0k} + \sum_{k=0}^{2} \left(r_k^{(5)}\right)^* h_{1k} + \sum_{k=0}^{2} \left(r_k^{(6)}\right)^* h_{2k}\end{aligned} \tag{25}$$

$$\begin{aligned}\tilde{x}_1 = &\sum_{k=0}^{2} r_k^{(0)} h_{1k}^* - \sum_{k=0}^{2} r_k^{(1)} h_{0k}^* + \sum_{k=0}^{2} r_k^{(3)} h_{2k}^* \\ &+ \sum_{k=0}^{2} \left(r_k^{(4)}\right)^* h_{1k} - \sum_{k=0}^{2} \left(r_k^{(5)}\right)^* h_{0k} + \sum_{k=0}^{2} \left(r_k^{(7)}\right)^* h_{2k}\end{aligned} \tag{26}$$

$$\begin{aligned}\tilde{x}_3 = &\sum_{k=0}^{2} r_k^{(0)} h_{2k}^* - \sum_{k=0}^{2} r_k^{(2)} h_{0k}^* - \sum_{k=0}^{2} r_k^{(3)} h_{1k}^* \\ &+ \sum_{k=0}^{2} \left(r_k^{(4)}\right)^* h_{2k} - \sum_{k=0}^{2} \left(r_k^{(6)}\right)^* h_{0k} - \sum_{k=0}^{2} \left(r_k^{(7)}\right)^* h_{1k}\end{aligned} \tag{27}$$

$$\begin{aligned}\tilde{x}_3 = &-\sum_{k=0}^{2} r_k^{(1)} h_{2k}^* + \sum_{k=0}^{2} r_k^{(2)} h_{1k}^* - \sum_{k=0}^{2} r_k^{(3)} h_{0k}^* \\ &- \sum_{k=0}^{2} \left(r_k^{(5)}\right)^* h_{2k} + \sum_{k=0}^{2} \left(r_k^{(6)}\right)^* h_{1k} - \sum_{k=0}^{2} \left(r_k^{(7)}\right)^* h_{0k}\end{aligned} \tag{28}$$

The following definitions are now introduced.

$$p_i^{(t)} = \sum_{k=0}^{2} r_k^{(t)} h_{ik}^* \tag{29}$$

$$q_i^{(t)} = \sum_{k=0}^{2} \left(r_k^{(t)} \right)^* h_{ik} \tag{30}$$

Using the definitions of (29) and (30), (25) through (28) can be written as (31) through (34).

$$\tilde{x}_0 = p_0^{(0)} + p_1^{(1)} + p_2^{(2)} + q_0^{(4)} + q_1^{(5)} + q_2^{(6)} \tag{31}$$

$$\tilde{x}_1 = p_1^{(0)} - p_0^{(1)} + p_2^{(3)} + q_1^{(4)} - q_0^{(5)} + q_2^{(7)} \tag{32}$$

$$\tilde{x}_2 = p_2^{(0)} - p_0^{(2)} - p_1^{(3)} + q_2^{(4)} - q_0^{(6)} - q_1^{(7)} \tag{33}$$

$$\tilde{x}_3 = -p_2^{(1)} + p_1^{(2)} - p_0^{(3)} - q_2^{(5)} + q_1^{(6)} - q_0^{(7)} \tag{34}$$

During each iteration of the inner loop sequence values are computed for $p_i^{(t)}$ and $q_i^{(t)}$ from the received data according to (29) and (30). Values for the symbol estimates $\{\tilde{x}_0, \tilde{x}_1, \tilde{x}_2, \tilde{x}_3\}$ are accumulated over eight successive iterations of the inner loop corresponding to the eight time slots of the STB transmission. The resulting estimated symbol vector is passed to the detector and becomes the received codeword vector.

Path Characteristic Matrix Generation

The channel path characteristic matrix **H** is generated once for each STB transmission. The path characteristics remain constant during the 8 transmission time slots and during the decode processing. The path characteristics are generated based on the defined multipath characteristics, the antenna configuration and the attenuation and delay factors defined for each path.

Each path characteristic value starts with a pair of random values generated by a Gaussian random number generator as described above. The Gaussian random values are generated with zero mean and unit variance; after which, the values are scaled and offset to achieve the defined multipath mean and variance. The random pair is then used to form a complex value to which the attenuation and delay for the path is applied according to (7).

The antenna configuration and multipath selection determine how **H** matrix is populated. When multipath is enabled and the i^{th} transmit antenna and the k^{th} receive antenna are selected, the value of h_{ik} is populated as described. When either a transmitter or a receiver antenna is not selected, the paths associated with that antenna are all assigned a value of zero. This action effectively removes the unused paths from the MIMO model without requiring conditional changes in the operation of the remaining model components.

CHAPTER 6 HARDWARE MODEL

This chapter describes the hardware based MIMO system model used in this dissertation research. The hardware based model is implemented on a Xilinix Vertex-4 FPGA device and utilizes the MATLAB/Simulink environment for interface and control of the simulation. The hardware based model is the main focus of the simulation work of this dissertation research because of its high processing performance.

With some overhead for setup and data transfer, the hardware based model is able to process bits at a much greater rate than the software base model. This processing rate advantage allows the total number of bits processed at each SNR point to be several orders of magnitude larger than what could be processed by the software based model in the same amount of time. This allows for greater accuracy in the determination of BER especially as SNR increased.

MIMO communications systems are of interest primarily because they can achieve better BER verse SNR performance than simpler schemes. This means that the number of bit errors decreases more rapidly as SNR increases than for other systems. Conversely, this characteristic requires that many more bits be processed to accurately determine the BER at any SNR.

For example, consider the MIMO system model validation cases, described in Chapter 7, used in this dissertation research. In order to achieve a BER precision of 0.1% for first seven SNR points, the first validation case requires $8x10^5$ bits to be processed, the second validation case requires $3.4x10^6$ processed bits and the third validation case requires that

$3x10^7$ bits be processed. As a practical mater, the ability to process and evaluate bits at a high rate is important to the study of MIMO communications systems performance.

Another reason for implementing the MIMO system model in PGA hardware in this dissertation research is to evaluate its practicality. Mathematics and computer simulation can prove the performance characteristics that are possible using MIMO systems, but in order for the technique to be truly useful, it must be possible to achieve that performance using currently available computational technology. The MIMO system model implementation on the Xilinx Vertex-4 FPGA, a commercially available hardware device, shows that the practical use of MIMO is achievable if the predicted BER verse SNR performance can be achieved under the resource and precision constraints imposed by the hardware.

The remainder of this chapter describes the hardware implementation of the MIMO system model, its operation and relevant details. Like the software based model, this implementation follows the design of Chapter 3 and Chapter 4. Details are given where implementation is not straight forward or can significantly affect the results.

Development Tools

The hardware implementation of the MIMO system model used in this dissertation research is accomplished using two primary development tools, MATLAB/Simulink by The MathWorks and the Xilinx System Generator. MATLAB provides the main development environment supporting general computation, data analysis, and scripting. MATLAB is also the environment from which Simulink is invoked. Simulink provides a

graphical system modeling environment that supports the construction of the MIMO system model as interconnected functional blocks corresponding to that of Figure 4-2. The Xilinx System Generator provides block libraries for Simulink that translate directly to PGA hardware. System Generator also extends the functionality of the Simulink to support hardware synthesis, configuration and data transfer to and from the PGA hardware.

Number Representation

Numeric values are handled using fixed-point format throughout the hardware implementation. A fixed-point value is represented by a number of bits *b* of which the *r* lowest order bits represent the fractional portion and the *b-r* highest order bits represent the integer portion according to (35), where *N* is the value being represented and a_i is the value of the bit in the i^{th} position.

$$N = a_b 2^{b-r-1} + \Lambda + a_{b-r+1} 2^1 + a_{b-r} 2^0 + a_{b-r-1} 2^{-1} + \Lambda + a_0 2^{-r} \qquad (35)$$

Fixed-point number representation limits the range and precision of the values. The range of values that can be represented by unsigned fixed-point numbers is limited to $[0, 2^{b-r} - 2^{-r}]$ while the range of values that can be represented by signed fixed-point numbers is limited to $[-2^{b-r-1},\ 2^{b-r-1} - 2^{-r}]$ [72]. The precision with which fixed-point numbers can be represented is 2^{-r} regardless of the whether the representation is signed or unsigned.

Throughout the simulation model, the number of bits allocated to any particular value is chosen to make efficient use of the available PGA hardware resources. In each case, the number of bits *b* allocated to a value is chosen such that its range is sufficient to accommodate the worst case value excursion and that its precision is sufficient to support the required accuracy.

Nested Construction

The PGA hardware based MIMO system model is constructed in a nested configuration of scripts and models as illustrated by Figure 6-1. MATLAB supplies the outer layer and the encompassing environment in which the simulation model executes. The MATLAB environment holds the parameters that define the conditions for a simulation, such as SNR, antenna configuration, and path attenuation and delay. The MATLAB environment also collects and retains the results produced by a simulation: BER, bits processed and error count. MATLAB also supports scripting that allows a sequence of simulations to be configured and executed automatically so that the data points for an entire BER verses SNR curve can be computed. Finally, MATLAB is used to analyze the data generated by the simulations.

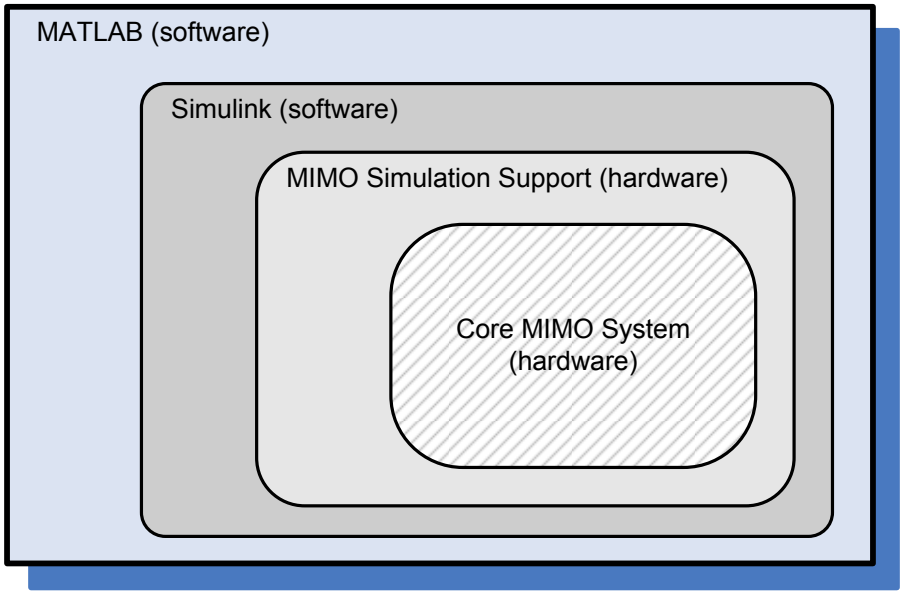

Figure 6-1 Nested Construction of the Hardware MIMO Model

Simulink is launched from within the MATLAB environment and provides the primary means for constructing and executing the MIMO system model. Blocks from the standard Simulink libraries support the interchange of data between the simulation model and the MATLAB environment. Blocks from the Xilinx libraries provide the interface between Simulink and those portions of the model implemented on the PGA hardware. In addition, by utilizing the Xilinx libraries, Simulink now supports hardware simulation, synthesis and debugging.

Beyond the interface components, the MIMO system model is constructed entirely of blocks from the Xilinx System Generator libraries. These blocks translate to a hardware implementation and execute in PGA hardware during simulation. System Generator provides a means to automatically download the PGA configuration and exchange data

with the PGA hardware during simulation. The "hardware-in-the-loop" feature of System Generator allows the hardware based portions of the model to work seamlessly with the Simulink components. In this dissertation research a network based Ethernet connection provided the link between Simulink and the hardware implementation executing on a Xilinx ML402 Vertex-4 FPGA evaluation board.

Within the hardware portion of the model, a MIMO Simulation Support layer provides the interface for exchange of condition parameters and results between the hardware and the software environment. This layer encapsulates the Core MIMO System model. All of the components shown in Figure 4-2 are contained within the Core MIMO System except for the Data Generator, FIFO and BER blocks which reside within the MIMO Simulation Support layer. Data generation and performance monitoring are handled by the Simulation Support layer.

The Core MIMO System layer implements the processing chain of Figure 4-2. This portion of the simulation model takes the conditions and data as input, processes the data through the MIMO system model and returns output data to the MIMO Simulation Support layer for evaluation. Noise and path characteristic are generated within the Core MIMO System.

Overview

This section provides an orientation to the inner layers of the MIMO system simulation model. Figure 6-2, Figure 6-3 and Figure 6-4 show these layers as they appear in the Simulink environment. Figure 6-2 is the outermost Simulink model layer and provides

the interface to the MATLAB environment. Figure 6-3 and Figure 6-4 respectively are the MIMO Simulation Support and the Core MIMO System described above.

With the exception of the MIMO_Simulation block, all of the blocks in Figure 6-2 are executed in the Simulink software environment during simulation. The input blocks SNR, TxAnt, RxAnt, MpMean, MpVar, UseMp, Atn(1) through Atn(9) and Dly(1) through Dly(9) are seen in the left half of Figure 6-2. The input blocks access simulation configuration values from the MATLAB workspace and import them into the MIMO simulation. Signals from these input blocks lead to the intermediate subsystems that transform the input values into a format that is usable by the PGA hardware. These transformed input values are then loaded to transfer registers from which they are sent to the PGA hardware.

The output blocks Errors, BER and TotalBits are seen in the right half of Figure 6-2. Output blocks take signals from intermediate subsystems that retrieve information from transfer registers. The BER is computed from TotalBits and Errors in the Simulink environment. Finally, Simulink output blocks make the results data available to the MATLAB workspace.

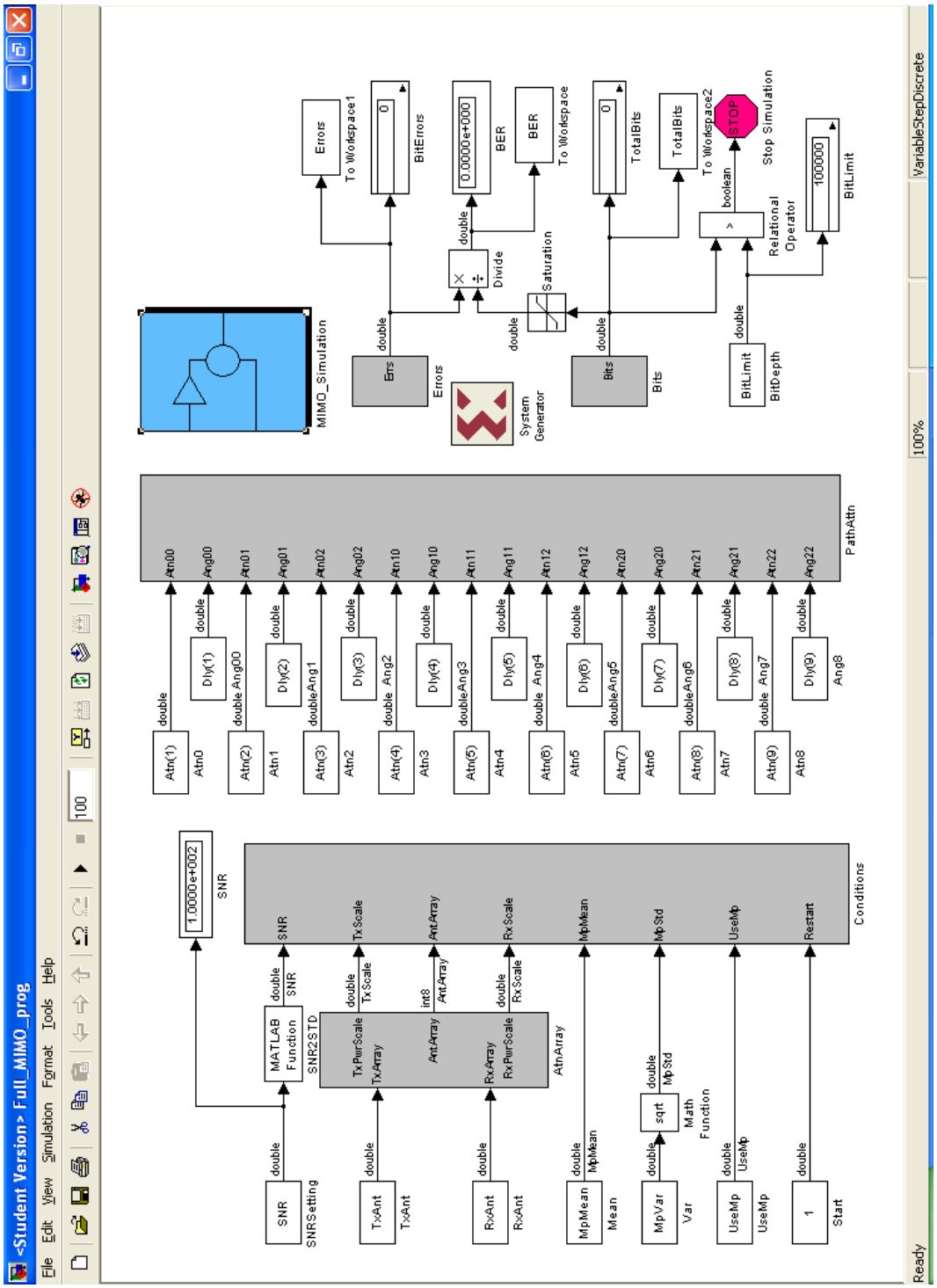

Figure 6-2 MIMO System Model Simulink Layer

During simulation the contents of the transfer registers are passed to and from the hardware by the System Generator hardware-in-the-loop functionality. The MIMO Simulation Support layer shown in Figure 6-3 interfaces the transfer registers to the PGA hardware portion of the model. The design of the interface ensures synchronous load and initiation of the PGA hardware. In order to avoid false error detection, a hold-off counter prevents the accumulation of bits and errors until the MIMO simulation pipeline has been filled. During the simulation, the bit and error counters are periodically sampled and returned to the outer layers of the model so that the execution progress may be monitored.

In Figure 6-3 the Data Generator, FIFO and BER blocks correspond directly to those of Figure 4-2. The remainder of the system is contained within the Core MIMO System block as shown in Figure 6-3. In Figure 6-3 these blocks have a "drop shadow" to assist with indentifying them. Figure 6-4 shows the Core MIMO System layer. Here again the blocks that directly correspond to Figure 4-2 are identified with a drop shadow. The remaining blocks in Figure 6-4 provide timing delays and simulation rate change but do not otherwise affect the processing.

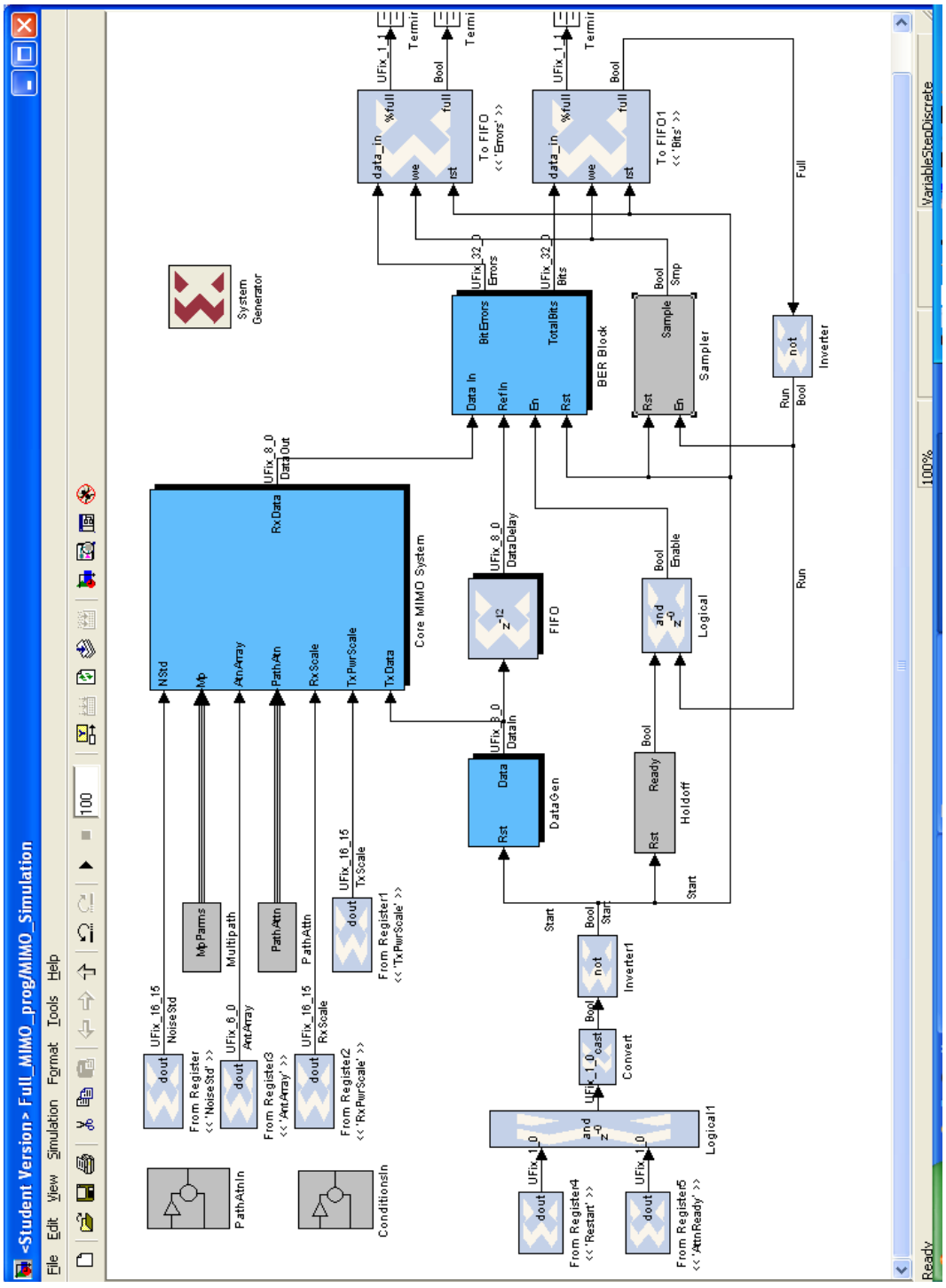

Figure 6-3 MIMO Simulation Support Hardware Layer

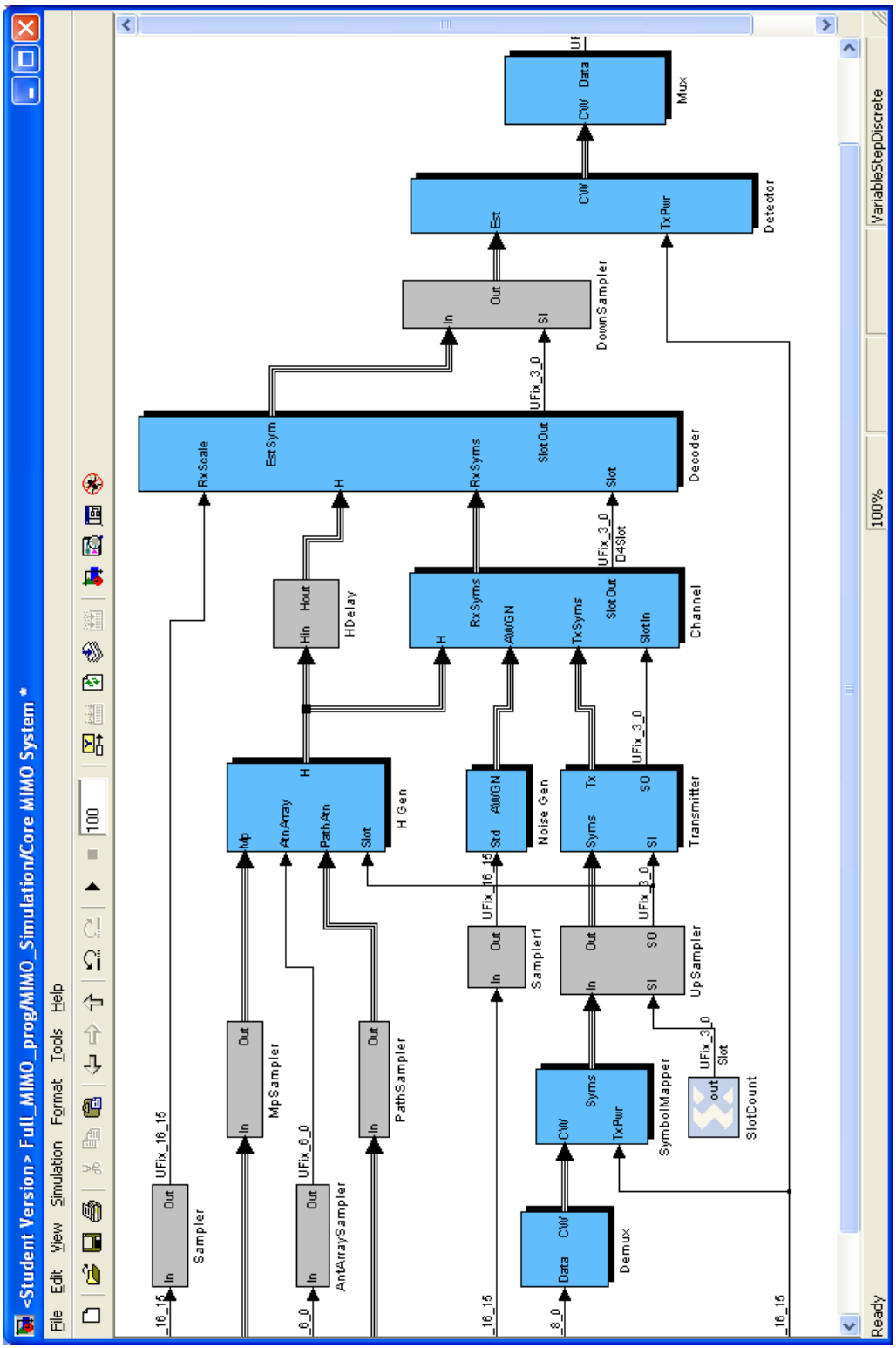

Figure 6-4 Core MIMO System

Interface

For the purposes of this dissertation research, the primary interface for the PGA hardware based MIMO system model is the MATLAB environment. Simulation conditions are set by assigning variables in the MALAB workspace before initiating a simulation. The simulation is then started from MATLAB. Following the completion of a simulation, the results are available in MATLAB, which allows the data to be analyzed and plotted.

The ability to utilize MATLAB scripts was also convenient in this dissertation research. MATLAB scripts simplify the configuration, execution and the collection of results especially where simulations must be repeated or executed unattended. Scripts support the sequencing of incremental SNR evaluations in the production of a BER verses SNR curve. Scripting capability is especially important during the validation, which requires repeating an evaluation of 15 BER verses SNR curves each consisting of 9 SNR data points.

Results Output

The results of a PGA hardware simulation are written to a comma separated variable text file. An example of a results file is shown in. The file includes the same information as that produced by the software base MIMO model of Chapter 5. The file contents include a header that provides an experiment identifier, time and date of the simulation, the minimum number of bits to process for each SNR data point and the multipath mean μ_m and variance σ_m^2. The header is followed by a table of channel path conditions, providing attenuation in dB and phase delay in degrees for each path. The channel path conditions table is then followed with the results for each SNR data point. Results

include the SNR in dB, total bits processed, error count and the BER. The file concludes with the elapsed simulation time in seconds, the total bits processed for all SNR values and the mean processing rate expressed in nanoseconds per bit. The file format allows the results data to be easily imported into a spreadsheet or into MATLAB for further analysis.

```
Simulation Run: HW-Case1-Tx0-Rx0
Started 9/7/2009 14:56:43
Minimum bits: 3.000000e+007

Multipath Channel: none
Channel Path Conditions (attenuation/phase)
-------------------------------------------
,      Rx0    ,,     Rx1    ,,    Rx2
Tx0:,  3.01,  0.00, -----,----- , -----,----- ,
Tx1:, -----,----- , -----,----- , -----,----- ,
Tx2:, -----,----- , -----,----- , -----,----- ,
SNR(dB),        Bits,  Errors,     BER
0.0,  3.001e+007, 2350879,7.835e-002
1.0,  3.001e+007, 1681490,5.604e-002
2.0,  3.001e+007, 1120005,3.733e-002
3.0,  3.001e+007,  682912,2.276e-002
4.0,  3.001e+007,  371572,1.238e-002
5.0,  3.001e+007,  176501,5.882e-003
6.0,  3.001e+007,   70853,2.361e-003
7.0,  3.001e+007,   22563,7.520e-004
8.0,  3.001e+007,    5632,1.877e-004
9.0,  3.001e+007,    1011,3.369e-005
Run time(sec),  173.277
Run bits     , 3.001e+008
Ave (nsec/bit) ,  577.481

Experiment complete ...
```

Figure 6-5 Example of Hardware Simulation Result File

Operation

Execution of the MIMO system model produces results for a single SNR data point. A simulation is initiated from MATLAB. Before starting a simulation, the conditions are

set, which include the minimum bits to process, the SNR, the use of multipath, the multipath mean μ_m and variance σ_m^2, the antenna configuration and the attenuation and delay for each path.

The simulation conditions are prepared by the Simulink layer for use by the PGA hardware. For each transmission path, the attenuation, A_{ik}, expressed in dB, and the delay, φ_{ik}, expressed in degrees, are converted to a single complex attenuation value, z_{ik}, according to (19). The square root of the multipath variance σ_m^2 is multipath standard deviation σ_m that is used within the simulation model for generation of the multipath characteristic matrix **H**. The SNR, expressed in dB, is converted to a value of noise standard deviation according to (36), where σ_{Noise} is the noise standard deviation and the SNR based on the symbol energy $E_s=1$.

$$\sigma_{Noise} = \frac{1}{\sqrt{4 \bullet SNR}} \tag{36}$$

The prepared condition parameters are stored to transfer registers from which they are passed to the PGA hardware. Once the simulation conditions are transferred to the PGA hardware, they are stored in working registers where they are accessible during simulation.

The process of loading the simulation conditions to the PGA hardware requires a finite period of time before which the simulation output is not valid. To prevent the accumulation of false errors, counters are initialized to zero and are held in this state until the loading of the simulation conditions is complete and the PGA processing pipeline has

properly filled. A Start flag is transferred with the simulation conditions. The Start flag is the final register transferred during the loading of the simulation conditions. It signals that the condition parameters are ready and enables the simulation processing. Since several clock cycles are required before the PGA hardware pipeline is filled and valid data begins to emerge from the simulation, the hold off counter prevents accumulation of the bit and error measurements until a predetermined number of clock cycles have elapsed.

Referring to Figure 6-3, the 8-bit input data words are generated in the hardware model using a 31-bit maximal length linear feedback shift register (LFSR) [52]. The input data words are supplied to the Core MIMO System as well as to the FIFO. The FIFO delays a data word for the precise number of hardware clock cycles required for the same data word to be processed through the Core MIMO System.

Data words emerging from the Core MIMO System and the FIFO are supplied to the BER block which then directly compares the two data words; counting the errors and the total bits. A sampler periodically stores the counter values to transfer buffers to be sent back to the Simulink environment.

Sampling the counter values in this way minimizes data transfer and prevents the PGA hardware from stalling should the buffers become full, while at the same time providing sufficient information to allow monitoring the progress of the simulation. However, since the termination condition is evaluated in the Simulink layer of the model, the sampling technique slightly affects the resolution with which the bit limit can be

specified. The simulation always processes at least the minimum number of bits specified, but may process several thousand additional bits due to this sampling resolution.

Simulation results for a SNR data point include the actual bits processed, the errors detected and the BER. The total bits and error counts are stored as 32-bit integer values. The 32-bit integers allow for counts in excess of $4x10^9$. This limit could be increased in future work by increasing the bit width of the counters and FIFOs. The BER value is computed in floating point format at the Simulink layer from the total bit and error counts.

Platform

In this dissertation research the Xilinx ML402 Vertex-4 FPGA evaluation board is utilized for the hardware execution platform. The ML402 is designed around the Xilinx XC4VSX35 FPGA, a member of the Vertex-4 FPGA family [22]. The hardware portion of the MIMO communications system model runs on the ML402 board. The hardware model is loaded to the Vertex-4 FPGA via Ethernet connection by System Generator at the start of a simulation. Once loaded to the Vertex-4 FPGA, the model is started and allowed to run asynchronously to the MATLAB/Simulink software portion of the model. This achieves the greatest processing rate but requires that synchronizing elements discussed earlier be included to allow coordination of the MATLAB/Simulink software and PGA hardware portions of the model.

Processing Rate

The ML402 circuitry supplies a 100 MHz clock signal to the Vertex-4 FPGA allowing a minimum clock cycle time of 10 nsec. However, for this dissertation research the minimum clock cycle was set to 15 nsec to relieve certain timing constraint issues that prevented successful hardware synthesis of the model. In addition, internal PGA resource limitations of the Vertex-4 FPGA prevented a complete parallel implementation of the channel. Therefore, only the transmitter, channel, decoder, noise generator and **H** matrix generator are processed at the maximum clocking rate. The remainder of the model is processed at 1/8 the maximum rate so that one data word transits through the model every 120 nsec for a maximum processing rate of 66.7×10^6 bits/second. However, the effective rate is slightly reduced by the data transfer time and the necessity to synchronize operation of the hardware and software portions of the model. As given in Chapter 8, processing rates of up to 123 nsec per byte or 65.0×10^6 bits/second are achieved using the hardware-based model.

Hardware Model Implementations

The following sections provide details on specific areas of the PGA hardware model whose implementation is unique or significant to the results obtained.

Pseudorandom Number Generator

Pseudo-random number generators (PNG) are used throughout the PGA hardware MIMO system model for data, channel noise and channel characteristic generation. The generation of channel noise n_i and channel characteristics h_{ik} are of great importance to obtaining accurate simulation results. These generators are configured using instances of

the same Gaussian PNG. The MIMO system hardware model utilizes 10 instances of this generator. This section provides an overview of the Gaussian PNG design and details are described in the sections to follow.

Like the software based MIMO system model, the PGA hardware based MIMO simulation generates Gaussian distributed random values using the inverse transform method. The hardware implementation first generates uniformly distributed random values using a linear feedback shift register (LFSR) employing a skip-ahead technique to whiten the generated values [49]. A composite look up table is then used to transform the linearly distributed values into Gaussian distributed values [50]. In addition, both the LFSR and the composite look up table designs minimize the required hardware resources while producing values at a rate of one per hardware clock cycle. This innovative architecture for the PNG design is a salient component of this dissertation research.

Linear Feedback Shift Register

The linear feedback shift register (LFSR) is a method of implementing a multiplicative congruential generator (MCG) that is readily accomplished in hardware. The MCG is a basic mathematical algorithm for the generation of uniformly distributed random numbers [53]. The LFSR uses a set of n single bit storage registers and a feedback logic network that implements a generating polynomial. The generating polynomial may be expressed in an algebraic form as in (37) where $q_i^{(t)}$ represents the content of the i^{th} bit of an n-bit register at time t, w_i is the binary weighting factor applied to the i^{th} bit and the symbol $\oplus$ is the binary XOR function.

$$q_{n-1}^{(t+1)} = w_{n-1}q_{n-1}^{(t)} \oplus w_{n-2}q_{n-2}^{(t)} \oplus \Lambda \oplus w_1 q_1^{(t)} \oplus w_0 q_0^{(t)} \tag{37}$$

The generating polynomial is responsible for the characteristics of the pseudorandom number sequence produced by the LFSR and many polynomials have been extensively studied [51]. It is generally desirable to have a generating polynomial that produces a number sequence that repeats only after 2^n-1 values, known as a maximal length sequence. The generating polynomial utilized by the MIMO system PGA hardware model of this dissertation research, was obtained from Xilinx application note DS257 [52] and is expressed in (38). This generating polynomial produces a maximal length sequence with a 55-bit register providing a sequence of more than $3.6\text{x}10^{16}$ values.

$$q_{54}^{(t+1)} = q_{24}^{(t)} \oplus q_0^{(t)} \tag{38}$$

This length of the pseudorandom number sequence allows the same polynomial to be utilized for all generators while maintaining a unique number sequence for each. This is accomplished by seeding each generator with a different starting value. The seeds are obtained by sampling a single instance of the generator every 2^{32} cycles. Thus the generators are ensured to produce non-overlapping sequences for the first $4.3\text{x}10^9$ hardware cycles.

All MCG have a deficiency in that a small correlation exists between successive values in the generated sequence [53][54]. This deficiency is insignificant in some applications, such as for the data generator used in this model, but can be detrimental to the noise and channel characteristic generation. The correlation in the pseudorandom sequence

produced by a LFSR using the generating polynomial of (38) can be observed in a plot of the autocovariance as shown in Figure 6-6. Patterns can also be observed when complex numbers, formed by taking successive values from the LSFR, are plotted on the complex plane as shown in Figure 6-7. This correlation also imparts a low-pass characteristic to the power spectral density as seen in Figure 6-8 [54]. In terms of the LFSR this correlation can be intuitively understood by noting that for any single step in the sequence, the generating polynomial alters only a few of the bits in the storage registers, while the remainder of the bits are simply shifted by one position [55].

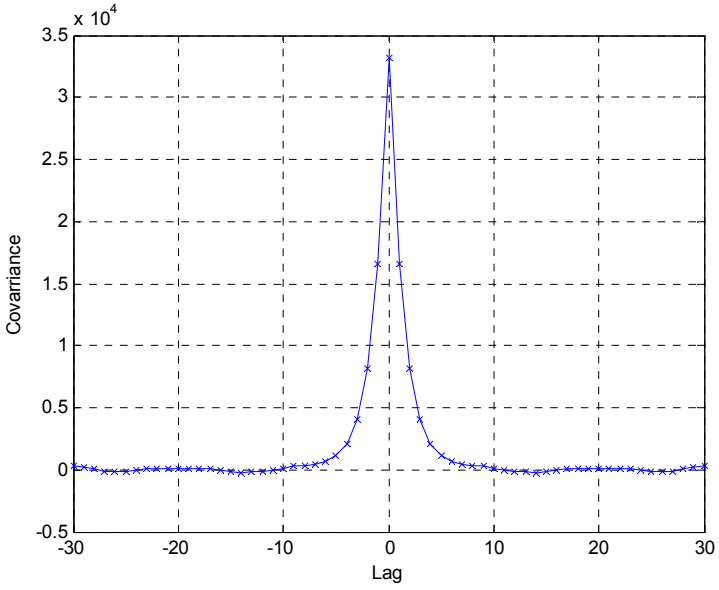

Figure 6-6 Autocovariance of Basic LFSR
Sample of 10^5 sequential values show correlation around 0 lag

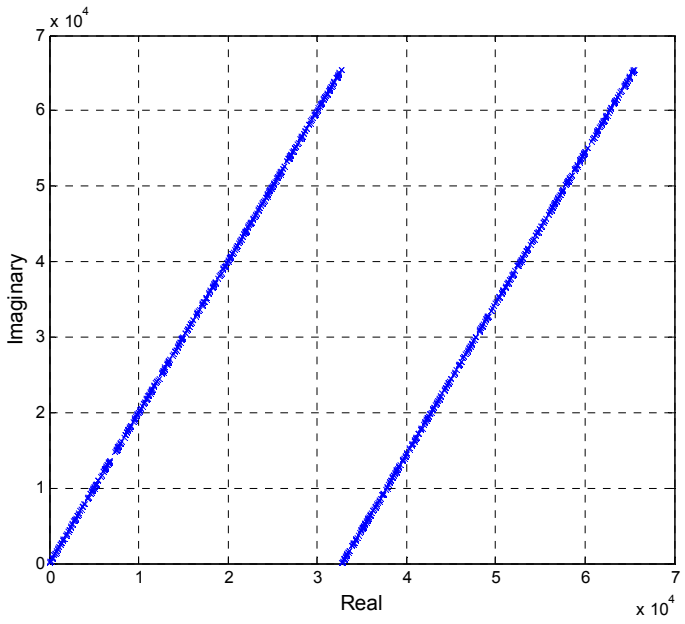

Figure 6-7 Complex pairs generated by the basic LFSR

Patterns are seen when values are plotted as complex pairs

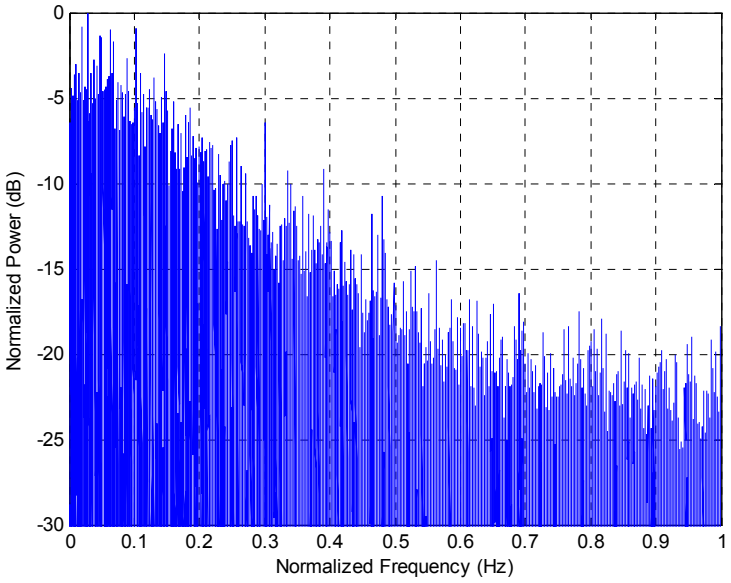

Figure 6-8 Power spectral density of basic LFSR

Sample of 10^5 sequential values shows a lowpass characteristic.

Many techniques have been proposed to mitigate this correlation [54][55]. However, most techniques either are not conducive to PGA hardware implementation or require too great a demand on such hardware resources to be of use in this dissertation research. Therefore, the multiple-bit skip-ahead technique was utilized.

The fundamental concept of the skip-ahead technique is to obtain a value from the LFSR and then advance the LFSR by k cycles before taking another value [55]. It may be observed from (37) that only a single new bit is generated during each cycle of the LFSR. The remaining bits are simply shifted by one position. The skip-ahead technique ensures that k new bits have been generated between successive samplings of the LFSR.

A disadvantage of the skip-ahead technique is that k cycles are required to produce each new random value. If this basic form of skip-ahead where utilized here, the MIMO system model throughput would be slowed by a factor of $1/k$. The multi-bit skip-ahead technique advances the LFSR k states in a single cycle maintaining the system throughput while achieving the decorrelating property of the basic skip-ahead technique.

The generating polynomial of (37) expresses how q_{n-1} at time $t+1$ is computed from the states of the q_i at time t. The remaining bits, q_{n-2} through q_0, are simply shifted by one bit from time t to time $t+1$ so that the operation of the LFSR over one clock cycle is expressed by the (39).

$$\left\{\begin{array}{l} q_{n-1}^{(t+1)} = w_{n-1}q_{n-1}^{(t)} + w_{n-2}q_{n-1}^{(t)} + \Lambda + w_1 q_1^{(t)} + w_0 q_0^{(t)} \\ q_{n-2}^{(t+1)} = q_{n-1}^{(t)} \\ \quad\quad \mathrm{M} \\ q_1^{(t+1)} = q_2^{(t)} \\ q_0^{(t+1)} = q_1^{(t)} \end{array}\right\} \tag{39}$$

Using matrix notation (39) can be expressed as (40) [49].

$$\mathbf{q}^{(t+1)} = \mathbf{A} \bullet \mathbf{q}^{(t)} \tag{40}$$

Where:

$$\boldsymbol{A} = \begin{bmatrix} w_{n-1} & w_{n-2} & \Lambda & w_1 & w_0 \\ 1 & 0 & & & 0 \\ 0 & 1 & & & 0 \\ \mathrm{M} & & \mathrm{O} & & \mathrm{M} \\ 0 & 0 & \Lambda & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{q} = \begin{bmatrix} q_{n-1} \\ q_{n-2} \\ \mathrm{M} \\ q_1 \\ q_0 \end{bmatrix}$$

Here **q** is the register state vector and **A** is the transition matrix for one execution cycle, from *t* to *t+1*. The first row of the transition matrix **A** is the coefficients of the generating polynomial and the remaining rows represent a single bit shift operation.

It is now possible to compute the register state vector **q** at time *t+k* for any arbitrary *k* using (41). Furthermore, it may be seen from (41) that matrix $\mathbf{A}^k$ allows the computation of $\mathbf{q}^{(t+k)}$ directly from $\mathbf{q}^{(t)}$. The matrix $\mathbf{A}^k$ is computed by raising the matrix **A** to the power of *k*, which is easily accomplished numerically using MATLAB. Therefore, by implementing $\mathbf{A}^k$ with the LFSR feedback network multi-bit skip-ahead is achieved.

$$\mathbf{q}^{(t+k)} = \mathbf{A} \bullet \mathbf{q}^{(t+k-1)} = \mathbf{A} \bullet \mathbf{A} \bullet \mathbf{q}^{(t+k-2)} = \mathbf{A}^{k} \bullet \mathbf{q}^{(t)} \tag{41}$$

A consequence of the multi-bit skip-ahead technique is an increase in the PGA hardware resources required to implement the feedback network. This increase is dependent on the generating polynomial and the value of *k*. One contribution of this dissertation research is to develop a set of rules that express conditions under which the resulting feedback network requires a minimum increase in PGA hardware resources [49].

Consider a 4-bit LFSR with the generating polynomial given in (42) having coefficients $\{ w_3, w_2, w_1, w_0 \} = \{ 1, 0, 0, 1 \}$.

$$q_3^{(t+1)} = q_3^{(t)} + q_0^{(t)} \tag{42}$$

The transition matrix $\mathbf{A}_{1001}{}^3$ for this LFSR at $k = 3$ as given by (43).

$$A_{1001}^{3} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^{3} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{43}$$

Here the matrix $\mathbf{A}_{1001}{}^3$ represents the set of binary equations (44) that involve up to four terms each.

$$\begin{Bmatrix} q_3^{(t+1)} = q_3^{(t)} \oplus q_2^{(t)} \oplus q_1^{(t)} \oplus q_0^{(t)} \\ q_2^{(t+1)} = q_3^{(t)} \oplus q_1^{(t)} \oplus q_0^{(t)} \\ q_1^{(t+1)} = q_3^{(t)} \oplus q_0^{(t)} \\ q_0^{(t+1)} = q_3^{(t)} \end{Bmatrix} \tag{44}$$

Now consider another 4-bit LFSR with generating polynomial (45) having coefficients $\{ w_3, w_2, w_1, w_0 \} = \{ 0, 0, 1, 1 \}$.

$$q_3^{(t+1)} = q_1^{(t)} + q_0^{(t)} \tag{45}$$

The transition matrix $\mathbf{A}_{0011}{}^3$ for this LFSR at $k = 3$ as given by (46) produces a simpler set of binary equations (47) each involving no more than two terms.

$$A_{0011}^3 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{46}$$

$$\begin{Bmatrix} q_3^{(t+1)} = q_3^{(t)} \oplus q_2^{(t)} \\ q_2^{(t+1)} = q_2^{(t)} \oplus q_1^{(t)} \\ q_1^{(t+1)} = q_1^{(t)} \oplus q_0^{(t)} \\ q_0^{(t+1)} = q_3^{(t)} \end{Bmatrix} \tag{47}$$

The significant difference between (43) and (46) is the w_3 coefficient of the generating polynomial. From (39) it can be seen that, for $k = 1$, $q_{n-1}{}^{(t+1)}$ is the only bit that is dependent on more than a single previous bit value. If $w_{n-1} = 1$, as in (42), then $q_{n-1}{}^{(t+1)}$ is dependent on its own previous value that, in turn, is computed from more than one prior bit value. Thus, as k increases the feedback equation for $q_{n-1}{}^{(t+1)}$ is compounded.

Using the generating polynomial of (42), (48) expresses the compounding at $k = 2$. Note how a 2 cycle skip turns the two term equation into a three term equation.

$$q_3^{(t+2)} = q_3^{(t+1)} \oplus q_0^{(t+1)} = (q_3^{(t)} \oplus q_0^{(t)}) \oplus q_1^{(t)} \tag{48}$$

However, when $w_3 = 0$ as in (45), the generating polynomial does not contain a q_{n-1} term and does not compound with increasing k. Using the generating polynomial of (45), (49) shows that after a 2 cycle skip the expression still contains only two terms.

$$q_3^{(t+2)} = q_1^{(t+1)} \oplus q_0^{(t+1)} = q_2^{(t)} \oplus q_1^{(t)} \tag{49}$$

Without compounding, as k increases the first row of the transition matrix $\mathbf{A}^k$ is only shifted left and therefore does not have more non-zero elements than terms in the generating polynomial. Eventually, however, a non-zero element will transition into the w_{n-1} position and compounding will occur. Notice from (46) that compounding will occur in the second example when $k = 4$. Generalizing, if w_j is the highest order non-zero coefficient of the generating polynomial, then compounding will not occur so long as $k \leq (n - j)$.

One further observation is that for each incremental increase in k the rows of $\mathbf{A}^k$ are propagated downward. Since all but the first row of $\mathbf{A}$ represents a 1-bit shift operation, all of the rows of $\mathbf{A}^k$ beyond the k^{th} row retain this characteristic with the generalization that they are a k-bit shift operation. This can be illustrated by considering the transition matrix $\mathbf{A}_{00111}{}^2$ for the 5-bit LFSR with the generating polynomial coefficients $\{w_4,w_3,w_2,w_1,w_0\} = \{0,0,1,1,1\}$ at $k = 2$ given in (50).

$$A^{2}_{00111} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{2} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (50)$$

These results can be summarized by the following rules:

I. The number of non-zero elements in each row of $\mathbf{A}^k$ is no greater than the number of terms of the generating polynomial as long as $k \leq (n - j)$, where w_j is the highest order non-zero coefficient of the generating polynomial.

II. Given the conditions of Rule I, only the first k rows of $\mathbf{A}^k$ will have more than one non-zero element.

III. All of the rows of $\mathbf{A}^k$ beyond the k^{th} row represent k-bit right-shifts operations.

The generating polynomial (38) for the 55-bit LFSR used in this dissertation research is selected because it is a two term equation that produces a maximal length sequence. Since 16-bit values are taken from the LFSR, a skip of $k = 16$ is utilized. From Rule I none of the rows of $\mathbf{A}^k$ will have more than two non-zero elements. In this case, Rule I is satisfied since $k = 16$, $n = 55$, $j = 24$ and $16 \leq (55 - 24) = 31$.

From Rule II, only the first 16 rows of $\mathbf{A}^k$ have more than one non-zero element and, from Rule III, the remaining 39 rows each represent a 16-bit logical right-shift operation.

Figure 6-9, Figure 6-10 and Figure 6-11 show the analysis of the pseudorandom number sequence generated by the LFSR employing skip-ahead. It can be seen in Figure 6-9 that the autocovariance for the sequence is a delta function at 0 lag indicating there is no detectable correlation between adjacent values. Plotting value pairs on the complex plane, Figure 6-10, reveals no discernable patterns. The power spectral density shown in Figure 6-11 now demonstrates a white noise spectrum and no longer has the significant lowpass rolloff. It should be noted that the generating polynomial is the same as that used by the LSFR of Figure 6-6, Figure 6-7 and Figure 6-8.

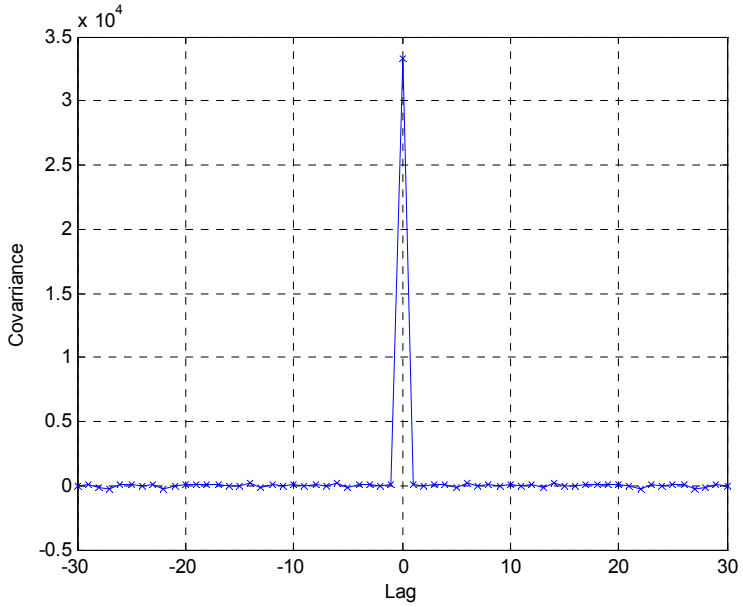

Figure 6-9 Autocovariance of LFSR with skip-ahead
Sample of 10^5 sequential values shows no correlation between values

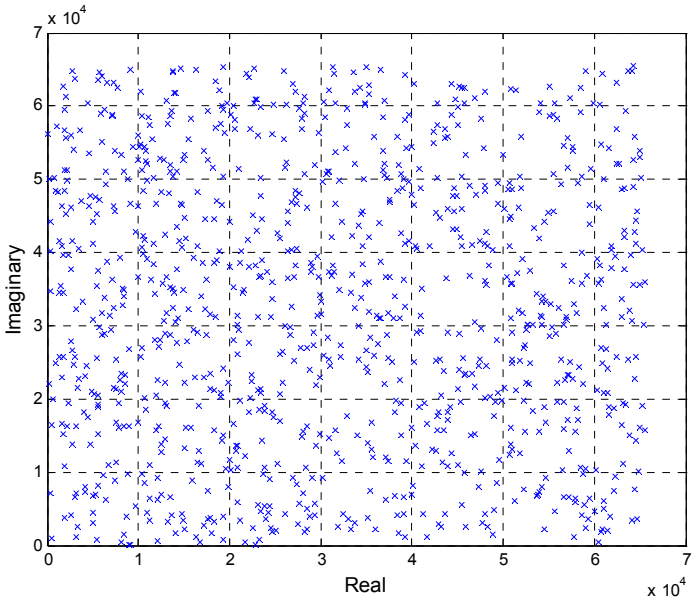

Figure 6-10 Complex pairs generated by LFSR with skip-ahead

No patterns seen in values when plotted as complex pairs

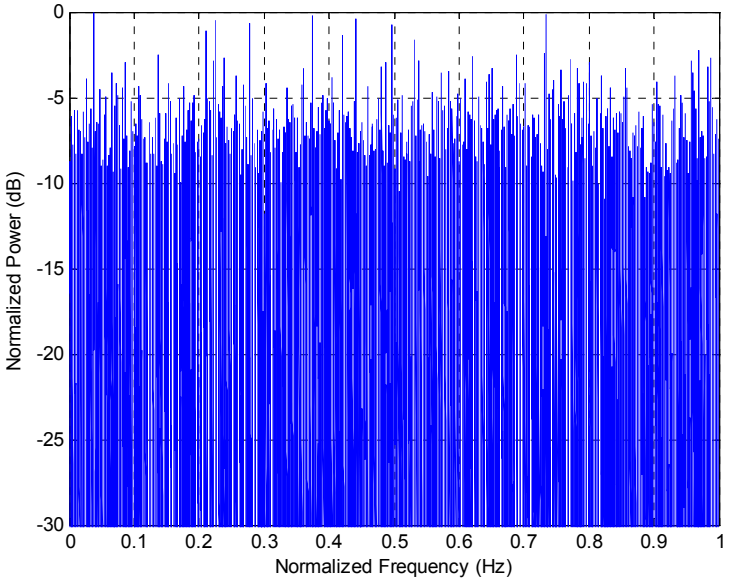

Figure 6-11 Power spectral density of LFSR with skip-ahead

Sample of 10^5 sequential values shows white noise spectrum

Composite Look-up Table Transform

Like the software based model, the PGA hardware based model utilizes the inverse transform method to produce Gaussian distributed pseudorandom values from the uniformly distributed values supplied by the LFSR described in the previous section. However, in order to reduce the complexity and required PGA hardware resources, the inverse Gaussian cumulative distribution function (IGCDF) is implemented with a lookup table (LUT). The memory resources required for the LUT depends upon the both the resolution and precession of the IGCDF representation. Consequently, the number of Gaussian sources required and the available memory resources limit the fidelity of the IGCDF and in turn limit the accuracy of the results that could be obtained from the MIMO system model. To overcome these resource limitations a composite LUT technique is developed as part of this dissertation research that reduces the resource requirements while allowing for improved IGCDF resolution and precision [50].

The PGA hardware based MIMO simulation model utilizes 10 Gaussian pseudorandom number generators (PNG). In addition, the throughput rate of the simulation is of great concern. Consequently, the following characteristics are important to the PNG implementation.

1. Minimum hardware resource requirements.

2. Consistent output rate.

3. Sufficient accuracy and range.

The composite LUT technique requires a tradeoff be made between the reduction in the size of the LUT and loss of fidelity to a Gaussian distribution. However as will be discussed, this loss of fidelity is confined to that part of the distribution curve that least affects the simulation results. The contribution of this inaccuracy is demonstrated to be minimal.

The basic PGA hardware architecture for a Gaussian PNG employing the inverse CDF method is illustrated in Figure 6-12. In the PGA hardware simulation model of this dissertation research the uniform PNG is a 55-bit maximal length LFSR from which 16-bits words are taken to obtain a uniformly distributed random number sequence. From each 16-bit word, 15 bits form an unsigned random integer *x* and the remaining bit *s* is used for the sign. A LUT provides a positive value *y* of the inverse Gaussian CDF (IGCDF) from *x*. The value *y* is then multiplied by either +1 or -1 as determined by *s* to yield the output value *n* that consists of positive and negative Gaussian distributed values.

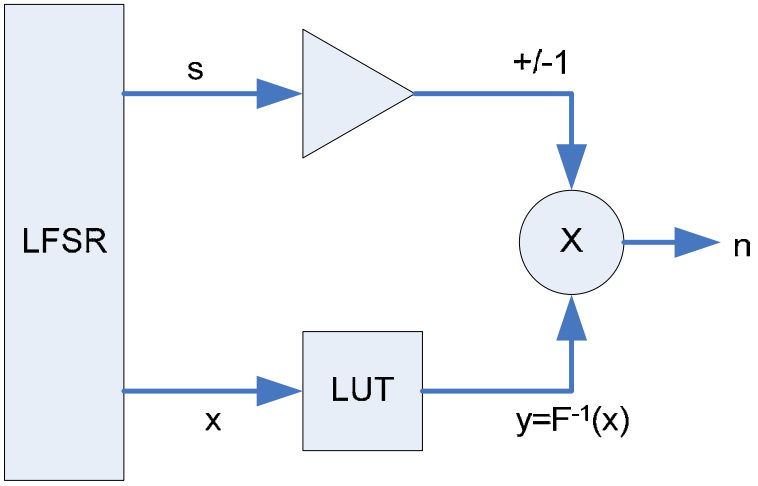

Figure 6-12 Basic architecture for a Gaussian PGN

The symmetry of the IGCDF curve is exploited by storing only the positive half of the transform in the LUT [61][63]. Thus, the unsigned integer x is transformed to a positive fixed point value y by the LUT and s provides the sign value independent of the LUT. Since s is uniformly distributed, the output value n is evenly distributed between positive and negative values.

This implementation has the advantage of requiring only minimal logic and is able to produce one new pseudorandom value for every clock cycle. The LUT contains fixed point values that can be made arbitrarily precise by allocating an appropriate number of bits to the LUT output width. The disadvantage of this implementation is that the LUT must hold 2^{15} values, one for each value of x.

Consider now the positive half of the IGCDF for a normal distribution of zero mean and unit variance, *N(0,1)* as shown in Figure 6-13. This curve represents the function to be implemented by the LUT. Note that the curve is approximately linear below a cumulative probability of 0.9, asymptotic near 1.0 but highly nonlinear in between.

Next, consider that since the IGCDF curve is monotonic, there is a unique relationship between the LUT input x and output y. If all possible values for x, or N_x, are mapped such that they are evenly spaced over [0.5, 1.0) then the IGCDF curve can be considered to be effectively sampled by x at intervals of $\Delta_x = 0.5 / N_x$.

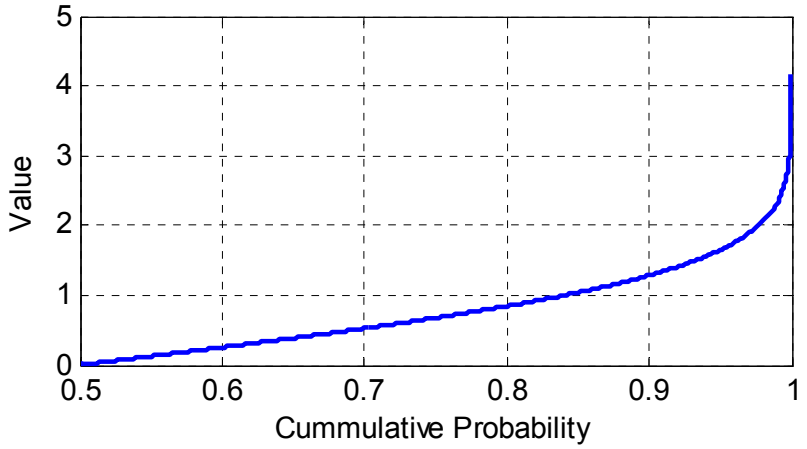

Figure 6-13 Positive half of IGCDF with zero mean and unit variance.

One manner in which the LUT size could be readily reduced is by increasing the size of the sampling interval or equivalently by decimating the LUT. However, to produce a value of y for each value of x, more than one x must be mapped to each value of y.

If w is the input to a LUT that is now intentionally decimated by a factor of m, then $\Delta_w = m\Delta_x$ and $N_w = N_x / m$. Now all the values of x that fall within a single interval Δ_w yield the same value of y, as given in (51) where $F^{-1}(x)$ is the IGCDF. The result of such a decimation procedure is that the LUT now represents a stair-step approximation of the IGCDF curve.

$$y = F_x^{-1}\left(w_1 \mid w_1 \le x \le w_2\right) \equiv F_w^{-1}(x) \tag{51}$$

Reducing the size of the LUT in this manner introduces two factors for a loss in the fidelity of output Gaussian distribution. The first loss factor ε_x results from the stair-step approximation of the IGCDF. This error is expressed by (52) for any value of x.

$$\varepsilon_x = F_w^{-1}(x) - F_x^{-1}(x) \tag{52}$$

This loss factor will introduce the least error in the flattest portion of the IGCDF curve, the region below a cumulative probability of 0.9, and the most error in the steepest part of the curve, near 1.0.

The second loss factor introduced by the decimation of the LUT is due to truncation of the IGCDF range. Since the smallest value of x is mapped to a cumulative probability of 0.5, the largest value of x maps to a point that is essentially one sampling interval short of 1.0 as given in (53), where the symbol → is to be interpreted as "maps to".

$$x_{\max} \rightarrow 0.5 + \Delta_w(N_w - 1) = 1 - \Delta_w = 1 - m\Delta_x \tag{53}$$

The result is to limit the extent of the "tails" of the Gaussian distribution. This effect is exasperated by the decimation factor m, as seen in (53). In addition, the high rate of change in the asymptotic portion of the IGCDF occurring near a cumulative probability of 1.0 implies that the amount of the Gaussian tail range that is unavailable is sensitive to the size of Δ_w and can thus be significant.

It can now be stated that there are two extremes to the LUT design of Figure 6-12. A high-resolution (small m) LUT provides high fidelity but has greater memory requirements, while a low-resolution (large m) LUT requires less memory but has lower fidelity. The innovative composite LUT design utilized in this dissertation research provides a compromise solution that utilizes the advantages of both while allowing control for the associated disadvantages [50].

The composite architecture of Figure 6-14 combines a low-resolution LUT with a high-resolution LUT in a single PNG. The design follows the essential principles of Figure 6-12 but utilizes two LUTs. A multiplexer selects between y_{Lo}, the output of the low-resolution LUT, and y_{Hi}, the output of the high-resolution LUT. A decoder controls the multiplexer selection so that every value of x is mapped either to the value of y_{Lo} or y_{Hi}. Moreover, the decoder decision is such that the range of x is effectively split into two ranges at a point that shall be designated x_{split}.

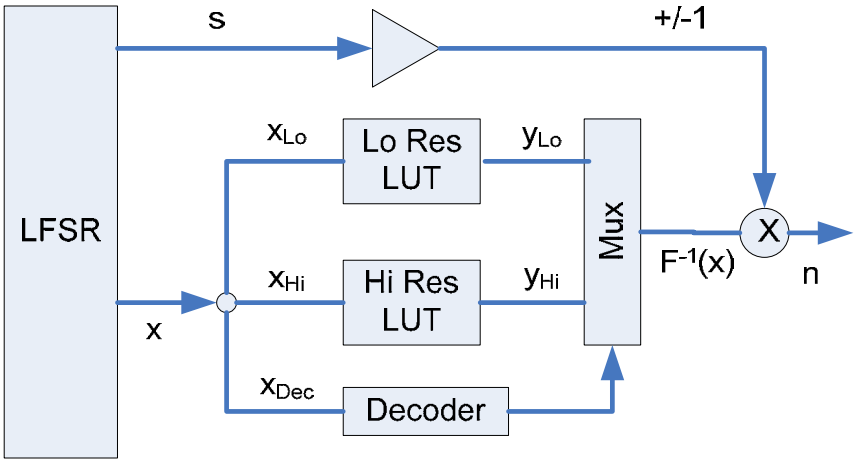

Figure 6-14 Composite LUT hardware architecture for Gaussian PGN

The low range x values ($x < x_{split}$) are mapped to y_{Lo}, and the high range x values ($x \geq x_{split}$) to y_{Hi}. This design maps the most non-linear portions of the IGCDF curve to the high-resolution LUT, while the more linear region of the curve is represented with the low resolution LUT. It is desirable to represent the non-linear portions of the IGCDF curve with high-resolution since this region has the highest rate of change and is responsible for the critical tail region of the output Gaussian distribution.

The flatter portion of the IGCDF curve with a cumulative probability of less than 0.9 introduces less of the first loss factor ε_x when the resolution is lowered. This portion of the IGCDF curve is responsible for the less critical central region of the output Gaussian distribution. A reduction in resolution for this portion of the IGCDF curve will result in the least loss of fidelity and in the least critical region of the resulting output distribution.

Since one of the objectives of this dissertation research is to minimize complexity as well as hardware resource requirements, an efficient decoder scheme is desired. Consider that x is a b-bit binary word as expressed by (54) where $x^{(i)}$ represents the value of the bit of x in the 2^i place.

$$x = x^{(b-1)}2^{b-1} + \ldots + x^{(1)}2 + x^{(0)} \tag{54}$$

Now consider that a high-resolution LUT holds one value y for every value of x and would therefore require an input word of b-bits. Next, further consider that a low-resolution LUT is a decimated version of a high-resolution LUT. The size of a low-resolution LUT is equal to the size of a high-resolution LUT divided by m. If m is constrained to a power of 2, then the number of bits p required for the input word to a low-resolution LUT is given by (55).

$$p = b - \log_2(m) \tag{55}$$

Moreover, the input to a low-resolution LUT would consist of exactly the p highest order bits of x as in (56).

$$x_{Lo} = x^{(b-1)}2^{b-p-1} + \ldots + x^{(b-p-2)}2 + x^{(b-p-1)} \tag{56}$$

In the composite LUT design of Figure 6-14, the value x_{split} defines a value on the range of x below which all values are taken from the low-resolution LUT and above which the high-resolution LUT is utilized. Therefore, the low-resolution LUT in the composite design need only hold the values of y_{Lo} for $x < x_{split}$ while the high-resolution LUT need only hold values of y_{Hi} for $x \geq x_{split}$.

In order to facilitate a seamless transition between the two tables, x_{split} is constrained to a value of x_{Lo}. Furthermore, x_{split} can also be constrained so as to require that its binary value consists of all 1's for the r highest order bits and 0's for all other bits. The decoder is then simply designed to select y_{Hi} whenever the r highest order bits of x are all ones. Thus the decoder is implemented as a simple r-input AND-gate.

A fortuitous consequence of this decoding scheme is that all of the values in the high-resolution LUT correspond to values of x for which the r highest order bits are 1's. Thus, the width of the input word to the high-resolution LUT x_{Hi} need be only $q = b - r$ bits and the value x_{Hi} is given by the q lowest order bits of x as in (57).

$$x_{Hi} = x^{(q-1)}2^{q-1} + \ldots + x^{(1)}2 + x^{(0)} \tag{57}$$

One further consequence of this design is that the effective composite LUT output $F^{-1}(x)$ retains the range of the high-resolution LUT. Note that $F^{-1}(x)$ follows y_{Lo} with interval Δ_w

up to x_{split}. Beyond x_{split} each value of y_{Lo} is replaced by m values of y_{Hi} with interval Δ_x such that $x_{\max} \to 1 - \Delta_x$.

The characteristics of the composite-LUT architecture inferred from the discussion and Figure 6-14 are summarized as follows:

1. The decoder consists of an r-bit AND-gate whose inputs are the r highest order bits of x.

2. The low-resolution LUT is a subset of a decimated high-resolution LUT with reduction factor m, which is a power of 2.

3. The low-resolution LUT holds only those values of y corresponding to $x < x_{split}$ and has no more than $N_w = N_x / m$ elements.

4. The input to low-resolution LUT, x_{Lo} , consists of the p highest order bits of x where p is determined by (55).

5. The value of x_{split} always corresponds to a value of x_{Lo} such that its r highest order bits are all 1's and all other bits are 0's.

6. The high-resolution LUT holds only those values corresponding to $x \geq x_{split}$ and its size is 2^q.

7. The composite LUT produces values of y_{Lo} for values of $x < x_{split}$ according to (51), and values of y_{Hi} for every value of $x \geq x_{split}$

8. The range of the composite LUT is equal to the range of a high-resolution LUT.

Values for m and r must be chosen such that the required hardware resources are minimized while the loss of fidelity to the output Gaussian distribution is not severely impacted. Note that the hardware resource requirements for the composite LUT architecture are primarily due to the memory needed. If M is the total memory required for both the high and the low-resolution LUT, then M may be used as a cost factor of the composite implementation.

If the interval size Δ_x of the high-resolution LUT is chosen so that the architecture of Figure 6-12 provides the necessary fidelity for the intended application, then the y_{Hi} can be used as a reference for this loss of fidelity. Deviation from the reference by the y_{Lo} values can then also be quantified as ε_x and correspond to the loss in fidelity due to the LUT size reduction. More specifically, the worst case error ε_{max} occurring anywhere over the range x is of the most interest. Therefore, the squared worst case error ε^2_{max} is taken as a second cost factor for the composite LUT implementation.

Since there is a limited number of combinations of m and r for any given word size of x, a complete search can be performed for values of M and ε^2_{max}. Figure 6-15 illustrates how the two cost factors vary with the possible selections of m and r. The points have been sorted in order of increasing ε^2_{max} in Figure 6-15. The combinations are numbered sequentially so that a cross reference is required to determine the values of m and r for each case.

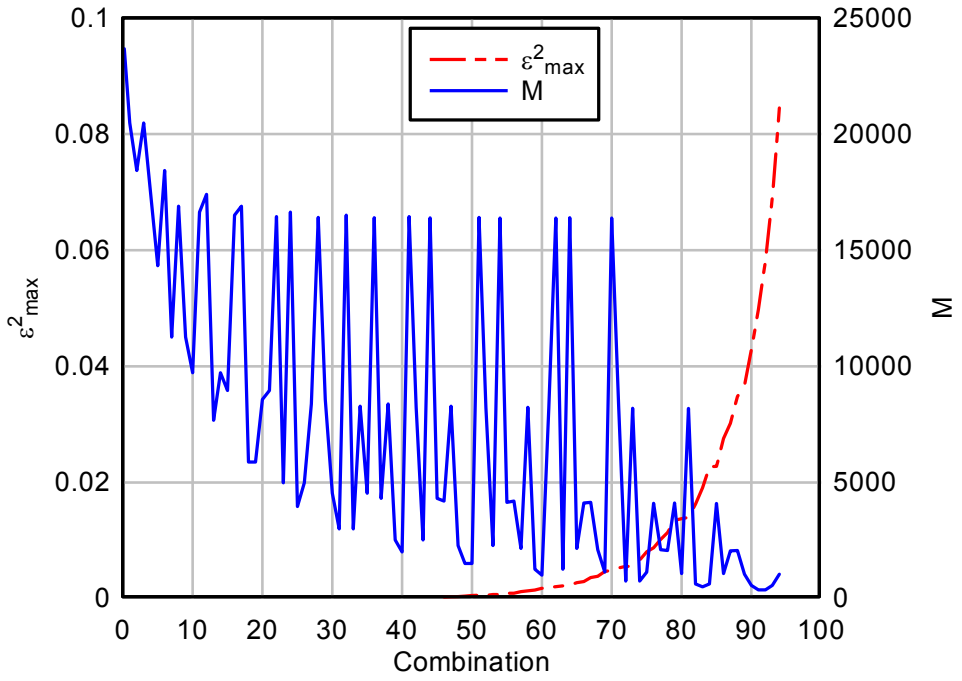

Figure 6-15 Cost factor plot for composite LUT:
Memory requirement, M, and error, ε^2_{max}, for combinations of m and r when $b = 15$.
Points are sorted in order of increasing ε^2_{max}.

It can be seen from Figure 6-15 that the memory cost M varies widely across the range. Therefore, the approach here is to first choose an acceptable value for ε^2_{max} and then find a combination of m and r that yields the minimum M while not exceeding this desired ε^2_{max}.

Table 6-1 Cost factor of composite LUT with ε^2_{MAX} near 2^{-12}

m	r	ε^2_{max}	M
8	7	8.22×10^{-5}	4320
128	2	8.77×10^{-5}	8384
16	6	1.10×10^{-4}	2528
32	5	1.42×10^{-4}	2016
256	1	1.49×10^{-4}	16448
4	9	1.80×10^{-4}	8240
64	4	1.82×10^{-4}	2528
2	11	2.37×10^{-4}	16392
128	3	2.42×10^{-4}	4320
8	8	2.78×10^{-4}	4208

In the hardware based MIMO system model of this dissertation research, a fixed point binary precision of 2^{-6} or $\varepsilon^2_{\max} = (2^{-6})^2 = 2^{-12} = 2.44x10^{-4}$ is required for the IGCDF tables in order to achieve the results reported in Chapter 7 and Chapter 8. Table 6-1 lists combinations of *m* and *r* that yield ε^2_{max} in the vicinity of the desired value. From Table 6-1 the combination $m = 128$ and $r = 3$ satisfies the error requirement while using only 4320 storage elements, where one storage element is the hardware resources required to hold one value of *y*. However, searching further through Table 6-1 shows that the $m = 32$ and $r = 5$ also satisfies the error requirement but uses only 2016 storage elements. Therefore, these parameters are utilized by the MIMO system model.

In the MIMO simulation model, the initial implementation of the Gaussian PNG used the architecture of Figure 6-12 and required $M = 32768$ storage locations for the single LUT.

The final implementation for the Gaussian PNG uses the improved composite LUT architecture of Figure 6-14 with $m = 32$ and $r = 5$ with a total required memory $M = 2016$ storage elements of the same width and precision as the original single LUT design. Although there is a 94% reduction in the required memory, there is no measurable increase in supporting digital logic and no loss of execution rate. The simulation performance utilizing the composite LUT for the Gaussian PNG remained equivalent to that obtained with the original single LUT design.

Path Characteristic Generation

The transmission of each data byte through the MIMO system model requires the generation of nine channel path characteristics values $\{h_{00}, h_{01}, h_{02}, h_{10}, h_{11}, h_{12}, h_{20}, h_{21}, h_{22}\}$. Each h_{ik} is a complex number created from two Gaussian pseudorandom values. In addition, conditional processing may constrain the value of each h_{ik} based on antenna configuration and whether multipath is applied.

Initially, a fully parallel implementation of the path characteristic generator was attempted. This implementation sought to produce all nine h_{ik} values in one hardware clock cycle. This implementation required nine instances of the basic generator, including a total of 18 Gaussian PNGs. This approach proved to be impractical due to the hardware resource limitations of the Xilinx Vertex-4 FPGA. Therefore, a semi-parallel approach is adopted.

The MIMO system model assumes that the channel characteristics **H** are qausi-static, that is, **H** is constant over any single block transmission. In the MIMO system model of this

dissertation research, one block transmission transfers one byte (8-bits) of data and requires 8 time periods or time slots. In the PGA hardware based MIMO simulation, each time slot is processed in one hardware clock cycle and blocks are processed in 8 clock cycles each. Therefore, only 8 clock cycles are available for the generation of the 9 h_{ik} values.

In order to produce the required 9 h_{ik} values in 8 clock cycles, the h_{ik} are organized into two groups with a path characteristic generator utilized by each group. The direct path group consists of 3 path characteristics $\{h_{00}, h_{11}, h_{22}\}$, representing the transmission paths between corresponding transmit and receive antennas. The cross path group consists of 6 path characteristics $\{h_{01}, h_{02}, h_{10}, h_{12}, h_{20}, h_{21}\}$, representing the transmission paths between non-corresponding transmit and receive antennas.

The generators for the direct and cross path characteristics are essentially identical. Each generator produces one path characteristic value per clock cycle. A time slot count is used to select the specific path characteristics produced. The path characteristic values are stored to double buffered latches for use during the simulated block transmission.

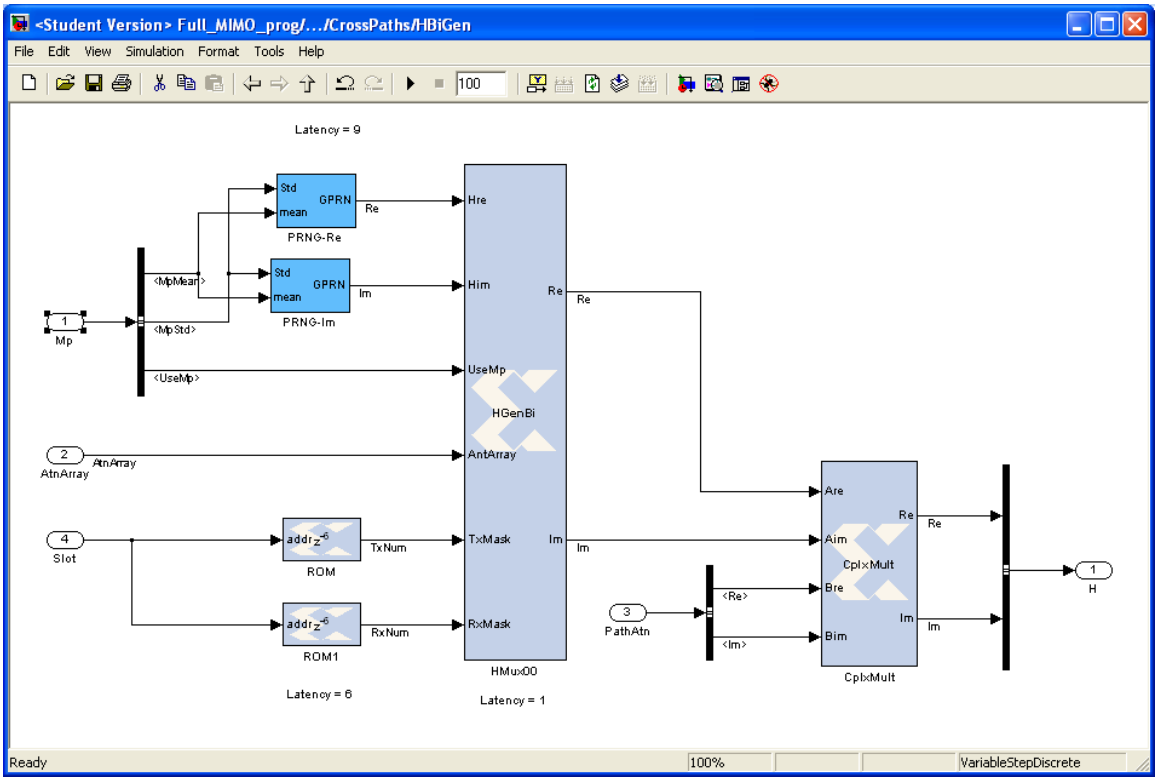

Figure 6-16 Channel Path Characteristic Generator

The cross path characteristic generator is shown in Figure 6-16. Path characteristic generation utilizes two Gaussian pseudorandom values that are adjusted to provide the multipath mean μ_m and variance σ_m^2 specified by the simulation conditions. Given the antenna configuration and an indication of whether multipath is assumed, a value for ψ_{ik} is created based on the following rules.

I. If the path is not selected, i.e. either the associated transmit or receive antenna is not utilized, then $\psi_{ik} = 0$.

II. If the path is selected but multipath is not assumed, then $\psi_{ik} = 1$.

III. Otherwise, the two Gaussian pseudorandom values are utilized for the real and imaginary parts of the complex value ψ_{ik}.

The value for h_{ik} is computed by multiplying ψ_{ik} by the complex path attenuation factor z_{ik} according to (19).

Decoder

The decoder utilized by the PGA hardware based MIMO model follows the method described in Chapter 5 for the software based model. The main concern for the hardware based model is to minimize the resources required by the implementation. This section compares the incremental implementation utilized in the PGA hardware model of this dissertation research to a direct implementation of (9), (10), (11) and (12).

In evaluating the implementations, a storage element is defined as the hardware memory resources required to hold one real signed valued. Complex values are represented in rectangular form and require two storage elements each. Only persistent storage is counted, that is storage for values that must be held longer than the time required to evaluate a single function. Storage for intermediate computational values is not counted.

Processing resources are counted in terms of additions and multiplications. For the purposes of comparison, it is understood that the PGA hardware possesses both multiplication and addition elements. The existence of multiply-accumulate elements of the PGA is ignored.

A simple multiplication is defined as the multiplication of two real signed values. Likewise, simple addition is defined as the addition of two real signed values. Complex multiplication and addition are implemented using combinations of simple operations.

Consider the direct implementation of (9), (10), (11) and (12). First, recognize that an STBC is, by definition, block oriented and an entire block of received symbols must be collected before decoding can be completed. In a direct implementation, all of the $r_k^{(t)}$ for one block are stored before processing can begin. The block period is 8 time slots with 3 symbols received during each time slot. Further, each received symbol is complex and requires 2 storage elements. Therefore, 48 storage elements are required to hold the received symbols before decoding can begin.

Next consider the processing resource demands of the decoding. We note that (9), (10), (11) and (12) are each a summation of 18 complex multiplications and 17 complex additions, or 72 complex multiplications and 68 complex additions for all of the processing.

Utilizing the tenants of complex mathematics [43], it can be seen from (58), (59) and (60) that complex addition requires 2 simple additions, while from (61) it can be seen that complex multiplication requires 4 simple multiplication and 2 simple additions. Thus, the decoding process by the direct implementation requires a total of 288 simple multiplications and 280 simple additions.

$$z_1 = a_1 + ib_1 \qquad \text{for } a_1 \text{ and } b_1 \text{ real} \tag{58}$$

$$z_2 = a_2 + ib_2 \qquad \text{for } a_2 \text{ and } b_2 \text{ real} \tag{59}$$

$$z_1 \pm z_2 = (a_1 \pm a_2) + i(b_1 \pm b_2) \tag{60}$$

$$z_1 z_2 = (a_1 a_2 - b_1 b_2) + i(a_1 b_2 + a_2 b_1) \tag{61}$$

Now consider the incremental implementation as described in Chapter 5. Table 6-2 shows the usage of the $p_i^{(t)}$ and $q_i^{(t)}$ terms in (31), (32), (33) and (34). Note that in any one time slot only three $p_i^{(t)}$ or $q_i^{(t)}$ terms need be computed and only three of the four symbol estimates are affected. Further note, as mentioned in Chapter 5, that the symbol estimates $\{\tilde{x}_0, \tilde{x}_1, \tilde{x}_2, \tilde{x}_3\}$ are produced by accumulating $p_i^{(t)}$ and $q_i^{(t)}$ terms from the appropriate time slots.

Table 6-2 Use of p and q Terms in Symbol Estimation

t	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
$\tilde{x}_0$	$+p_0$	$+p_1$	$+p_2$	–	$+q_0$	$+q_1$	$+q_2$	–
$\tilde{x}_1$	$+p_1$	$-p_0$	–	$+p_2$	$+q_1$	$-q_0$	–	$+q_2$
$\tilde{x}_2$	$+p_2$	–	$-p_0$	$-p_1$	$+q_2$	–	$-q_0$	$-q_1$
$\tilde{x}_3$	–	$-p_2$	$+p_1$	$-p_0$	–	$-q_2$	$+q_1$	$-q_0$

The incremental implementation is time slot oriented, such that one stage is computed during each time slot and the estimation results are accumulated over the block period. The process begins by initializing the accumulators to zero at time slot, $t = 0$. Using one

complex accumulator for each symbol estimate, $\tilde{x}_0$, $\tilde{x}_1$, $\tilde{x}_2$, and $\tilde{x}_3$ the process proceeds as follows during each time slot:

1. Get the received symbols for time slot
2. Compute $p_i^{(t)}$ during $t=0,1,2,3$ or compute $q_i^{(t)}$ during $t = 4, 5, 6, 7$
3. Accumulate the $p_i^{(t)}$ or $q_i^{(t)}$ terms as indicated in Table 6-2.

After time slot $t = 7$, the symbol estimates for the incremental architecture are equivalent to those produced by a direct implementation. Thus, storage for the incremental implementation consists of only the four complex accumulators or a total of 8 storage elements.

During any one time slot either the three $p_i^{(t)}$ or $q_i^{(t)}$ must be computed. Each requires 3 complex multiplications and 2 complex additions with another complex addition required to accumulate the estimate. Therefore, during any one time slot a total of 9 complex multiplications and 9 complex additions or 36 simple multiplications and 36 simple additions are required.

Table 6-3 presents a summary of the storage and resource requirements for both the direct implementation and the incremental architecture. Of the two methods described here, the incremental method has a lower demand for storage requiring only 8 storage elements compared to 48 storage elements for the direct method. More dramatic though is the difference in the demands for processing resources, where the direct method requires 288

multipliers and 280 adders, the incremental method requires only 36 multiplier and 36 adders.

Table 6-3 Comparison of Required Resources and Storage

	Direct Method	Incremental Method
Storage Elements	48	8
Simple Multiplications	288	36
Simple Additions	280	36

Furthermore, routing and interconnection in the PGA hardware synthesis, overhead and other resource demands that may grow in proportion to the number of multiply and addition operations has not been explicitly considered here. For instance, consider that every multiply and addition operation may require the operands to be conveyed to the multiplier or adder via a routing or other hardware resource. The number of routing channels per operand depends on the bit-width required. The number of such routes are proportional to number of multiplies and additions.

Detector

The PGA hardware based MIMO model utilizes a maximum likelihood (ML) detector to convert the symbol estimates produced by the decoder into codewords. Maximum likelihood detection is equivalent to choosing the symbol from the base QPSK modulation symbol $\{s_0, s_1, s_2, s_3\}$ that has the minimum Euclidian distance in signal space to the estimated symbol $\tilde{x}$ [32]. This decision processing requires that the

Euclidian distance be computed between $\tilde{x}$ and each of the four symbols s_k. Since these computations require considerable PGA hardware resources, an alternative computation is developed. The alternative computation is based on Haykin [32] and is equivalent to (62), where $\tilde{x}_r$ and $\tilde{x}_i$ are the real and imaginary components of $\tilde{x}$ and s_{kr} and s_{ki} are the real and imaginary components of the reference symbol s_k.

$$\left|\tilde{x} - s_k\right| = \sqrt{\left(\tilde{x}_r - s_{kr}\right)^2 + \left(\tilde{x}_i - s_{ki}\right)^2} \tag{62}$$

A direct implementation of (62) requires 3 additions, 2 multiplications and a square root operation. Furthermore, (62) must be computed four times for each estimated symbol. There are four estimated symbols for each MIMO code block transmission so that, in total 48 additions, 32 multiplications and 16 square root operations are required.

Consider that minimizing (62) is equivalent to minimizing (63) and so the square root operation can be eliminated.

$$\left|\tilde{x} - s_k\right|^2 = \left(\tilde{x}_r - s_{kr}\right)^2 + \left(\tilde{x}_i - s_{ki}\right)^2 \tag{63}$$

Now, expanding (63) and collecting terms leads to (64).

$$\left|\tilde{x} - s_k\right|^2 = \left(\tilde{x}_r^2 + \tilde{x}_i^2\right) - 2\left(x_r s_{kr} + x_i s_{ki}\right) + \left(s_{kr}^2 + s_{ki}^2\right) \tag{64}$$

The first term of (64) is the energy of the estimated symbol. This term is independent of the reference symbol s_k and therefore may be ignored. The last term of (64) is the reference symbol energy E_s which is constant for all the s_k utilized in this MIMO system

model and may also be ignored. Finally, disregarding the constant scalar on the center term leads to the result that minimizing (64) is equivalent to maximizing (65) where D_k is the distance metric between $\tilde{x}$ and s_k.

$$D_k = \tilde{x}_r s_{kr} + \tilde{x}_i s_{ki} \tag{65}$$

Now consider that the reference symbols $\{s_0, s_1, s_2, s_3\}$ defined as in Table 3-2 all have the same magnitude. Furthermore, the symmetry of the symbol constellation leads to the fact that magnitude of the inphase (real) s_{kr} and quadrature (imaginary) s_{ki} components of each symbol are also the same as expressed by (66).

$$|s_{kr}| = |s_{ki}| = \varphi \qquad \text{for } k = 1, 2, 3, \text{ and } 4 \tag{66}$$

Using (66) the symbols may be expressed as in (67), (68), (69) and (70).

$$s_0 = \varphi + j\varphi \tag{67}$$

$$s_1 = -\varphi + j\varphi \tag{68}$$

$$s_2 = \varphi - j\varphi \tag{69}$$

$$s_3 = -\varphi - j\varphi \tag{70}$$

Now m_r and m_i are defined to be the real and imaginary distance metrics terms given by (71) and (72).

$$m_r \equiv \varphi\, \tilde{x}_r \tag{71}$$

$$m_i \equiv \varphi \, \widetilde{x}_i \tag{72}$$

Using (71) and (72), (65) can be expressed as the distance metrics D_0, D_1, D_2 and D_3 in terms of m_r and m_i as in (73), (74), (75) and (76).

$$D_0 = m_r + m_i \tag{73}$$

$$D_1 = -m_r + m_i \tag{74}$$

$$D_2 = m_r - m_i \tag{75}$$

$$D_3 = -m_r - m_i \tag{76}$$

Utilizing this method developed as part of this dissertation research, it is only necessary to compute one value of m_r and m_i for each $\tilde{x}$, evaluate the distance metrics according to (73), (74), (75) and (76) and then choose s_k based on the largest value of D_k. The method requires only 2 multiplications and 4 additions for each of the four symbol estimates in a MIMO code block. Therefore, the hardware resource requirement for the detector is reduced to a total of 8 multiplications and 16 additions, while the complicated square root operation is completely eliminated.

Evolution of the Hardware Architecture

The initial approach of the PGA hardware based MIMO system model described in this chapter was a fully parallel implementation. This approach facilitates the greatest data throughput rate by processing 8-bits of data for every hardware clock cycle. However,

the PGA hardware resources are exhausted in the attempt to instantiate the processing for the requisite 8 timeslots of the MIMO code block.

The architecture is then modified to implement the processing of the 8 timeslots sequentially. The effect of this change is to reduce the data throughput rate to 1-bit per clock cycle. Changing to sequential timeslot processing is not sufficient to fit the hardware model to the available PGA resources. Therefore, the channel characteristic generators are also reduced from a fully parallel implementation requiring 18 Gaussian PNGs to the design that requires only 4 Gaussian PNGs.

With the initial completion of the full PGA hardware base MIMO model, it was found that the validation tests results where not satisfactory. Where software based MIMO model could match the expected validation data to within 1% for most BER versus SNR data points, the PGA hardware based model produced errors of 50% or greater in some cases. It was observed that the hardware based model's error increased with increasing SNR. It became apparent that the observations are consistent with increasing error as BER decreases.

Failure of the hardware model to pass the validation tests prompted a thorough investigation of the PGA hardware model. The hardware model was checked for errors in implementation, inaccuracy due to numeric representation limits and improperly synchronized signal timing. Although some such errors were observed and corrected, the results of the validation tests remained essentially unchanged.

Next, a series of progressive PGA hardware models were created and validated against predicted results. Each successive model is built using the components and techniques proven in the previous models. Each new PGA hardware model increased in overall complexity. The purpose of this progressive technique is to identify any stage where errors are introduced into the PGA hardware model.

These models progressed from an amplitude shift keying (ASK) model having two real signal points, to a binary phase shift keying (BPSK) model having two complex signal points, to a QPSK model with same four complex signal points as the base modulation of the MIMO model. All of these PGA hardware models produced the predicted results for BER versus SNR.

The next step was to construct what can be considered as the MIMO shell model. This PGA hardware model consists of all of the parts of the MIMO system model described earlier except for the transmitter, channel and decoder. The symbol mapper is connected directly to the detector to provide a complete signal path through the model. The purpose of the MIMO shell model is to confirm that no errors were introduced during the interchange of simulation conditions or results between the hardware and software layers. Another purpose of this model is to confirm the proper operation of the system without MIMO processing or communications channel. A simple additive white Gaussian noise (AWGN) channel is next inserted into the MIMO shell model. Then the MIMO transmitter and decoder is introduced but without a multipath channel. These PGA hardware models all produced satisfactory validation test results.

However, when the multipath channel characteristic generator is introduced, the results of the validation tests again became unsatisfactory. This led to concern about the random number generators and the fidelity of the Gaussian distribution they produced. Detailed analysis of the PNG sequences revealed unexpected correlations. A literature search confirmed correlation in LFSR sequences to be a known issue and lead to the skip-ahead technique as one possible solution [53][54][56]. The skip-ahead technique is then adapted to the MIMO system PGA hardware model and is a significant contribution of this dissertation research.

With the introduction of the skip-ahead technique into the path characteristic and channel noise generators, the result of the validation tests become comparable to the those produced by the software based MIMO system model. Although the validation results could be considered satisfactory at this point, a question remained as to whether the results could be further improved with increased fidelity of the PNG Gaussian distribution especially into the tail regions, that is, the region of the Gaussian CDF approaching a cumulative probability of 1.0. However, the hardware memory resources required to expand the look up tables here are nearly exhausted. This condition led to the development of the composite look up table design described earlier [50]. The introduction of composite look up table in the 10 Gaussian PNG reallocated 66% of the hardware memory resources. This resulted in the PGA hardware MIMO system model described in this chapter and is another significant contribution of this dissertation research.

CHAPTER 7
VALIDATION

The MIMO simulation models describe in the previous chapters are validated by comparing their results to published work. Three test cases are used. In each case, the simulation models are used to produce an observed BER verses SNR curve and the results compared to theoretical curves or data published in the reference work. The first test case reproduces the observed BER curve for the uncoded QPSK base modulation as given by Haykin [32]. The second and third test cases reproduce the observed BER curves for the STBC system using three transmit antennas with one and two receive antennas as given by Tarokh et al. [10]. The test conditions were created using the available controls of the models without modification by appropriately configuring the signal path attenuations, antenna configurations and multipath conditions.

Case 1: Uncoded QPSK

The first validation case reproduces the BER curve for an uncoded QPSK transmission. The BER versus SNR curve for this case as described by Haykin [32] is shown in Figure 7-1. This case is based on an uncoded QPSK transmission between one transmit antenna and one receive antenna over an additive white Gaussian noise (AWGN) channel without multipath interference.

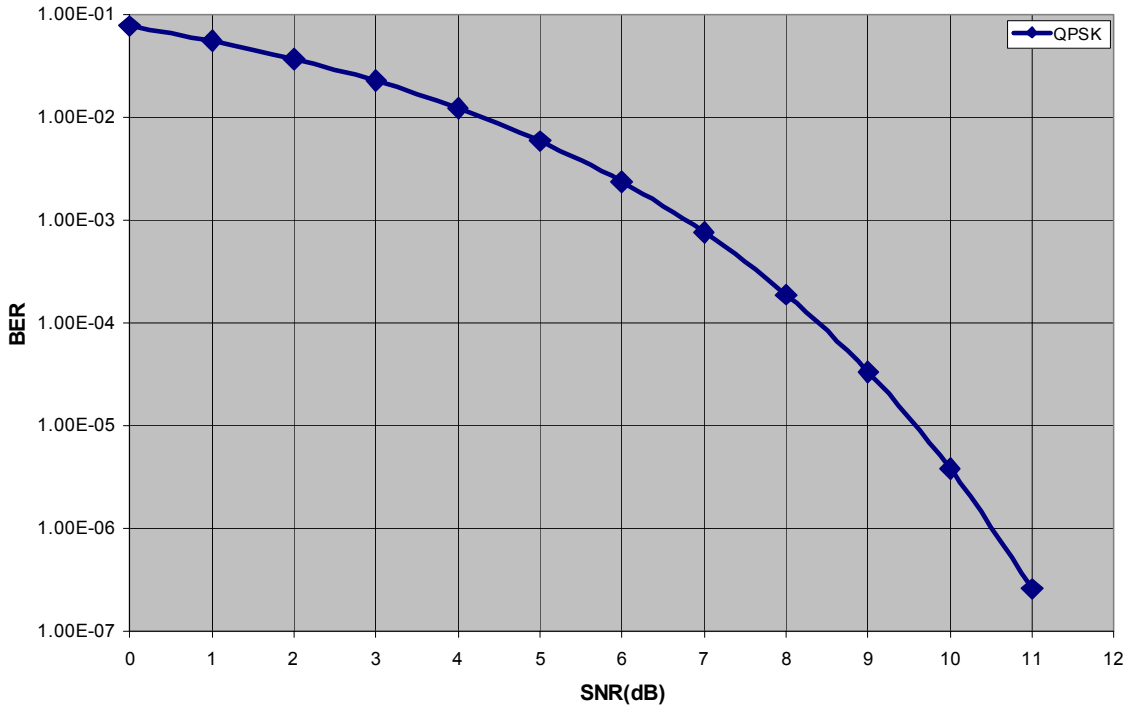

Figure 7-1 BER Curve for Uncoded QPSK

Reference Data

Probability of bit error P_b or BER for Case 1, as derived by Haykin, is given by (77) where E_b is signal energy per bit and N_0 is channel noise.

$$P_b = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{77}$$

Note here that in (77) the term E_b/N_0 is the SNR of the system. Since QPSK modulation carries two bits per symbol the bit energy is half the symbol energy as in (78) and the channel noise is determined by (79) where σ^2_{Noise} is the variance of the channel noise.

$$E_b = \frac{E_s}{2} \tag{78}$$

$$N_0 = 2\sigma_{Noise}^2 \tag{79}$$

Recalling that $E_s = 1$ in the models of this dissertation research; it can be seen that SNR as expressed in (80) leads directly to (36) where ρ is SNR.

$$\rho = \frac{E_b}{N_0} = \frac{E_s}{4\sigma_{Noise}^2} \tag{80}$$

Model Configuration

The conditions for this validation case are created in the models by disabling multipath and all but a single transmission path. In addition, the attenuation of the single transmission path is set to 3.01 dB. To understand why this attenuation is required, let the signal path characteristics be given as in (81).

$$h_{ik} = \begin{cases} h & for\ i = k = 0 \\ 0 & otherwise \end{cases} \tag{81}$$

Now assuming that only the signal path between the first transmit antenna and the first receive antenna exists. The signal observed by the Decoder over the block transmission is equal to *h* times the first column of **G** plus the AWGN random noise *n* as given in (82).

$$r = \begin{bmatrix} hx_0 + n \\ -hx_1 + n \\ -hx_2 + n \\ -hx_3 + n \\ hx_0^* + n \\ -hx_1^* + n \\ -hx_2^* + n \\ -hx_3^* + n \end{bmatrix} \tag{82}$$

Applying this to (9) gives (83), where *h* is assumed complex and all noise terms are collected into the final *n*.

$$\tilde{x}_0 = r^{(0)}h^* + \left(r^{(4)}\right)^* h \;=\; (hx_0 + n)h^* + (hx_0^* + n)^* h \;=\; 2|h|^2 x_0 + n \tag{83}$$

If the magnitude of *h* is chosen as $1/\sqrt{2}$, (83) becomes $\tilde{x}_0 = x_0 + n$ which is precisely the case for a QPSK symbol transmitted over an AWGN channel. The same reasoning can be applied to (10) through (12) so that $\tilde{x}_i = x_i + n$ for *i* = 0, 1, 2, 3. Thus the path attenuation must be set to approximately 3 dB to achieve these conditions as expressed in (84).

$$Attenuation = -20\log_{10}\left(\frac{1}{\sqrt{2}}\right) = 10\log_{10}(2) = 3.01\ dB \tag{84}$$

Case 2 and Case 3: MIMO

The second and third validation cases reproduce BER verses SNR curves published by Tarokh et al. [10]. These curves are produced for the same STBC implemented by the simulation models of this dissertation research. The second test case assumes a MIMO

coded signal transmitted from three transit antennas to a single receive antenna over a multipath AWGN channel. The third test case assumes the same signal and conditions but with two receive antennas. Both cases assume a multipath mean $\mu_m = 0$ and variance $\sigma_m^2 = 0.5$. The path attenuations are set to 0 dB for all transmission paths used. A plot of the BER versus SNR curves for these two cases is given in Figure 7-2.

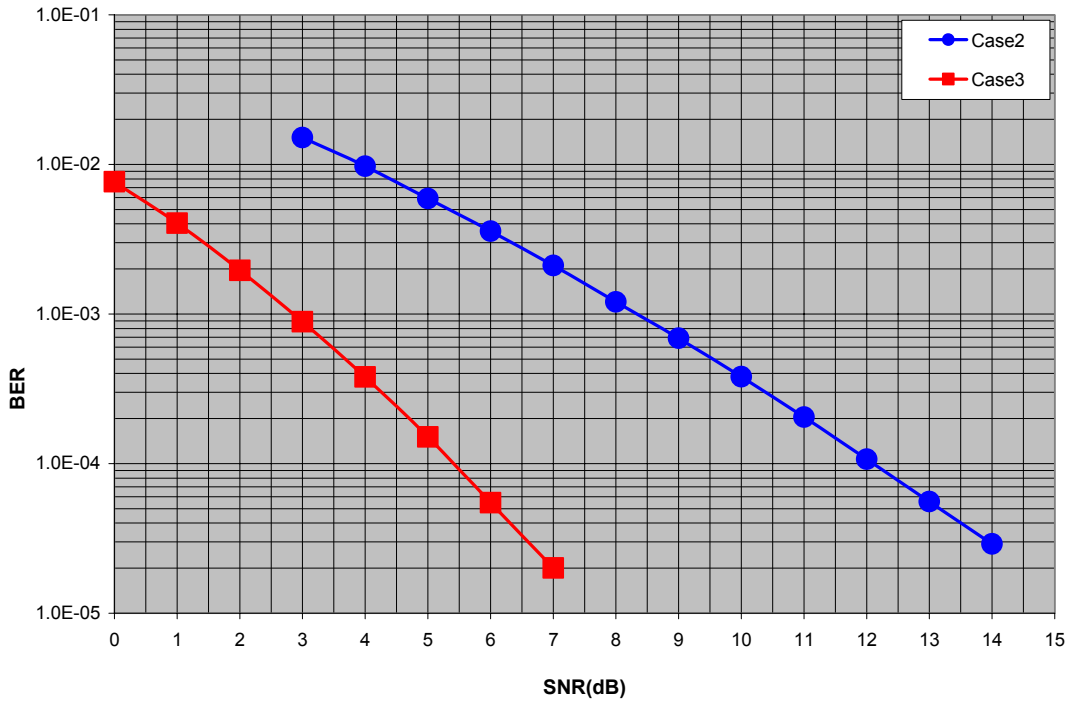

Figure 7-2 BER Curve of Validation Case 2 and Case 3

Reference Data

The reference data points for validation Case 2 and Case 3 are taken from the curves published by Tarokh et al. [10]. Unfortunately, the data published are presented only as plots. The data points are extracted from the published plots using the Engauge Digitizer

software [71]. This software allows data to be read from any plot, chart, drawing or map that is drawn to scale. After calibrating the software to the axes of the plots, the software is able to obtain the data points from the published curves with reasonable precision.

The data for validation Case 2 is taken from the 3 antenna curve of Figure 6 published in [10], while the data for validation Case 3 is taken from the 3 antenna curve of Figure 8 in [10]. The actual data utilized is given in Table 7-1. Note that the available data points do not cover the same range of SNR for all validation cases.

It should be noted that Tarokh defines SNR differently from the previous section of this dissertation research. Tarokh defines SNR or ρ' according to (85) and relative to symbol energy E_s whereas the simulation models define SNR in terms of bit energy E_b according to (80).

$$\rho' = \frac{E_s}{N_0} = 2\frac{E_b}{N_0} = 2\rho \tag{85}$$

A correction for this difference in SNR is easily accomplished by adding 3 dB to the SNR value for each reference data point. The correction has already been applied in Table 7-1.

Table 7-1 Reference Data for Validation Cases

SNR	BER Case 1	BER Case 2	BER Case 3
0	7.86×10^{-2}	---	7.65×10^{-3}
1	5.63×10^{-2}	---	4.04×10^{-3}
2	3.75×10^{-2}	---	1.96×10^{-3}
3	2.29×10^{-2}	1.51×10^{-2}	8.89×10^{-4}
4	1.25×10^{-2}	9.71×10^{-3}	3.80×10^{-4}
5	5.95×10^{-3}	5.91×10^{-3}	1.51×10^{-4}
6	2.39×10^{-3}	3.58×10^{-3}	5.48×10^{-5}
7	7.73×10^{-4}	2.11×10^{-3}	2.01×10^{-5}
8	1.91×10^{-4}	1.20×10^{-3}	---
9	3.36×10^{-5}	6.88×10^{-4}	---
10		3.81×10^{-4}	---
11		2.05×10^{-4}	---
12		1.07×10^{-4}	---
13		5.57×10^{-5}	---
14		2.91×10^{-5}	---

Model Configuration

The models are configured for validation Case 2 by enabling multipath, all three transmit antennas and one receive antenna. This results in the complete attenuation of all of the columns of **H** except for the first as in (86).

$$\mathbf{H} = \begin{bmatrix} h_{00} & 0 & 0 \\ h_{10} & 0 & 0 \\ h_{20} & 0 & 0 \end{bmatrix} \tag{86}$$

Similarly, validation Case 3 is configured to utilize multipath, all three transmit antennas and two receive antennas. This is accomplished by disabling one column of **H** as in (87).

$$\mathbf{H} = \begin{bmatrix} h_{00} & h_{01} & 0 \\ h_{10} & h_{11} & 0 \\ h_{20} & h_{21} & 0 \end{bmatrix} \tag{87}$$

Validation Results

This section presents the data obtained from both the software and hardware MIMO communications system models for each validation case as part of this dissertation research. Data obtained using the software model is presented in Table 7-2 through Table 7-6, while data obtained using the hardware model is presented in Table 7-7 through Table 7-11. For each validation case, the same conditions, SNR points and minimum number of bits are utilized for the software and hardware based models.

For each model there are 15 validation scenarios. Validation Case 1 utilizes one transmit antenna and one receive antenna and therefore result in 9 scenarios. Table 7-2 and Table 7-7 present the data obtained using the signal path between the first transmit antenna, Tx_0, and each of the three receive antennas, Rx_0, Rx_1 and Rx_2. Similarly, Table 7-3 and Table 7-8 present the data obtained when using the second transmit antenna, Tx_1, while Table 7-4 and Table 7-9 present the data obtained with the third transmit antenna, Tx_2.

Validation Case 2 and Case 3 each utilize three transmit antennas and result in three scenarios each. Validation Case 2 utilizes a single receive antenna with the results presented in Table 7-5 and Table 7-10 for Rx_0, Rx_1 and Rx_2. Validation Case 3 utilizes

two of the three receive antennas and the results are presented in Table 7-6 and Table 7-11 for each pair of receive antennas.

The data presented in the Tables includes the SNR, BER and error for each point and the mean error for each scenario. For each value of SNR, expressed in dB, the BER value obtained from the simulation is given. Error is defined as the difference between the BER value obtained in the test, BER_{sim}, and the reference data, BER_{ref}, given in Table 7-1 and expressed as a percentage of the reference value according to (88). The mean error then is the average of the error values over all SNR points.

$$Error = \frac{BER_{ref} - BER_{sim}}{BER_{ref}} \times 100 \quad \% \tag{88}$$

For each validation case, the SNR values are chosen from the available published or computed data. The lowest SNR value is selected as 0 dB or the lowest available SNR. As practical mater, the number of bits processed for each SNR point is chosen so as to limit the execution time required for any one software model simulation to less than 25 minutes. The highest SNR value was therefore selected so that an expected BER precision of 0.1% could be obtained while processing no more than 5 x 10^7 bits per SNR point. Note that the minimum bits processed are the same for each SNR point of a validation case. For validation Case 1 each SNR point is evaluated using 3 x 10^7 bits, validation Case 2 utilizes 3.5 x 10^7 bits and Case 3 uses 5 x 10^7 bits.

In general, both models match the reference data sets demonstrably well. The worst case software model validation test shows a mean error magnitude of only 3.62 % while 11 of

15 tests show 3.02 % or less. The worst case hardware model validation test shows a mean error magnitude of only 3.09 % with 12 of 15 tests showing 2.11 % or less.

Figure 7-3, Figure 7-4 and Figure 7-5 show plots of data generated by the software model for validation Case 1, Case 2 and Case 3 respectively. Figure 7-6, Figure 7-7 and Figure 7-8 show the plots for the data generated by the hardware model. In these plots, the observed BER versus SNR curves generated by the respective model for each test of a validation case is plotted along with the reference data. Although the individual curves cannot be distinguished in these Figures, it should be noted that there is little divergence of the curves across the entire SNR range.

It is noted that the curves generally show the greatest divergence at the highest SNR and that this observation is supported by the data of Table 7-2 through Table 7-11. This result may be attributed to a number of factors including those listed below. Suggested future work could involve improving the convergence of this data.

Some factors that may contribute to the observed divergence of the validation data from the reference include:

1. Insufficient number of sample bits, especially at high SNR where BER is low and error bits are rare such that the weight of a single error bit can be significant to the accuracy of the data.

2. Limitations of numeric precession within the models.

3. Inaccuracy of the estimated inverse Gaussian cumulative distribution functions utilized by the models especially in the tail region near a cumulative distribution of 1.0, which becomes increasingly significant as SNR increase.

4. The reference data for validation Case 1 is theoretical assuming mathematically perfect continuous Gaussian noise distribution and an infinite number of test bits.

5. The reference data for validation Case 2 and Case 3 was estimated from a graphic taken from a reproduction of the paper published by Tarokh et al. [10].

Overall, both the software and hardware based MIMO communications system simulation models of this dissertation research are considered to have passed the validation tests for all three cases.

Table 7-2 Software Model Validation Case 1 Results for Tx_0

	Rx_0		Rx_1		Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
0	7.84×10^{-2}	-0.38%	7.83×10^{-2}	-0.46%	7.83×10^{-2}	-0.48%
1	5.60×10^{-2}	-0.50%	5.60×10^{-2}	-0.48%	5.61×10^{-2}	-0.39%
2	3.73×10^{-2}	-0.60%	3.73×10^{-2}	-0.58%	3.73×10^{-2}	-0.63%
3	2.26×10^{-2}	-1.04%	2.27×10^{-2}	-0.78%	2.28×10^{-2}	-0.56%
4	1.24×10^{-2}	-0.97%	1.23×10^{-2}	-1.29%	1.24×10^{-2}	-1.05%
5	5.83×10^{-3}	-2.05%	5.86×10^{-3}	-1.63%	5.86×10^{-3}	-1.59%
6	2.33×10^{-3}	-2.40%	2.33×10^{-3}	-2.40%	2.32×10^{-3}	-2.73%
7	7.48×10^{-4}	-3.22%	7.38×10^{-4}	-4.49%	7.41×10^{-4}	-4.10%
8	1.81×10^{-4}	-5.03%	1.77×10^{-4}	-7.49%	1.76×10^{-4}	-8.02%
9	2.93×10^{-5}	-12.78%	2.80×10^{-5}	-16.64%	2.88×10^{-5}	-14.27%
Mean Error		-2.90%		-3.62%		-3.38%

Table 7-3 Software Model Validation Case 1 Results for Tx_2

	Rx_0		Rx_1		Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
0	7.82×10^{-2}	-0.52%	7.84×10^{-2}	-0.32%	7.83×10^{-2}	-0.46%
1	5.60×10^{-2}	-0.50%	5.60×10^{-2}	-0.50%	5.60×10^{-2}	-0.43%
2	3.73×10^{-2}	-0.58%	3.72×10^{-2}	-0.74%	3.74×10^{-2}	-0.42%
3	2.27×10^{-2}	-0.87%	2.27×10^{-2}	-0.65%	2.27×10^{-2}	-0.69%
4	1.24×10^{-2}	-0.97%	1.24×10^{-2}	-0.97%	1.23×10^{-2}	-1.29%
5	5.88×10^{-3}	-1.31%	5.83×10^{-3}	-2.06%	5.86×10^{-2}	-1.63%
6	2.35×10^{-3}	-1.60%	2.31×10^{-3}	-3.19%	2.33×10^{-3}	-2.44%
7	7.39×10^{-4}	-4.36%	7.46×10^{-4}	-3.52%	7.38×10^{-4}	-4.46%
8	1.79×10^{-4}	-6.45%	1.75×10^{-4}	-8.33%	1.82×10^{-4}	-4.72%
9	3.00×10^{-5}	-10.79%	3.06×10^{-5}	-9.00%	2.86×10^{-5}	-14.86%
Mean Error		-2.79%		-2.93%		-3.14%

Table 7-4 Software Model Validation Case 1 Results for Tx_3

	Rx_0		Rx_1		Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
0	7.83×10^{-2}	-0.43%	7.83×10^{-2}	-0.44%	7.84×10^{-2}	-0.37%
1	5.61×10^{-2}	-0.36%	5.60×10^{-2}	-0.47%	5.60×10^{-2}	-0.45%
2	3.73×10^{-2}	-0.44%	3.73×10^{-2}	-0.63%	3.72×10^{-2}	-0.74%
3	2.27×10^{-2}	-0.74%	2.27×10^{-2}	-0.82%	2.27×10^{-2}	-0.60%
4	1.24×10^{-2}	-0.81%	1.24×10^{-2}	-1.21%	1.24×10^{-2}	-1.21%
5	5.85×10^{-3}	-1.69%	5.86×10^{-3}	-1.66%	5.84×10^{-3}	-1.90%
6	2.34×10^{-3}	-1.94%	2.33×10^{-3}	-2.36%	2.34×10^{-3}	-2.15%
7	7.52×10^{-4}	-2.70%	7.36×10^{-4}	-4.81%	7.38×10^{-4}	-4.53%
8	1.80×10^{-4}	-5.82%	1.83×10^{-4}	-4.14%	1.78×10^{-4}	-6.92%
9	2.95×10^{-5}	-12.18%	3.28×10^{-5}	-2.46%	2.98×10^{-5}	-11.38%
Mean Error		-2.71%		-1.90%		-3.02%

Table 7-5 Software Model Validation Case 2 Results

	Rx_0		Rx_1		Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
3	1.51×10^{-2}	-0.45%	1.51×10^{-2}	-0.31%	1.51×10^{-2}	-0.45%
4	9.61×10^{-3}	-1.02%	9.63×10^{-3}	-0.82%	9.61×10^{-3}	-1.00%
5	5.98×10^{-3}	1.18%	5.97×10^{-3}	0.98%	5.96×10^{-3}	0.87%
6	3.60×10^{-3}	0.64%	3.61×10^{-3}	0.81%	3.60×10^{-3}	0.64%
7	2.11×10^{-3}	-0.11%	2.10×10^{-3}	-0.44%	2.10×10^{-3}	-0.49%
8	1.20×10^{-3}	-0.71%	1.20×10^{-3}	-0.21%	1.20×10^{-3}	-0.04%
9	6.63×10^{-4}	-3.60%	6.63×10^{-4}	-3.60%	6.71×10^{-4}	-2.36%
10	3.61×10^{-4}	-5.25%	3.62×10^{-4}	-5.15%	3.63×10^{-4}	-4.83%
11	1.94×10^{-4}	-5.28%	1.98×10^{-4}	-3.23%	1.98×10^{-4}	-3.23%
12	1.07×10^{-4}	0.04%	1.04×10^{-4}	-3.23%	1.04×10^{-4}	-2.39%
13	5.49×10^{-5}	-1.44%	5.21×10^{-5}	-6.41%	5.34×10^{-5}	-4.10%
14	2.93×10^{-5}	0.63%	2.81×10^{-5}	-3.32%	2.88×10^{-5}	-0.95%
Mean Error		-1.39%		-2.38%		-1.69%

Table 7-6 Software Model Validation Case 3 Results

	Rx_0 & Rx_1		Rx_0 & Rx_2		Rx_1 & Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
0	7.59×10^{-3}	-0.79%	7.58×10^{-3}	-0.92%	7.59×10^{-3}	-0.79%
1	3.99×10^{-3}	-1.24%	3.98×10^{-3}	-1.61%	4.00×10^{-3}	-1.22%
2	1.95×10^{-3}	-0.47%	1.94×10^{-3}	-0.72%	1.94×10^{-3}	-0.67%
3	8.79×10^{-4}	-1.06%	8.80×10^{-4}	-1.01%	8.80×10^{-4}	-0.92%
4	3.69×10^{-4}	-2.92%	3.72×10^{-4}	-2.16%	3.74×10^{-4}	-1.60%
5	1.44×10^{-4}	-4.51%	1.44×10^{-4}	-4.58%	1.45×10^{-4}	-3.91%
6	5.51×10^{-5}	0.45%	5.28×10^{-5}	-3.67%	5.31×10^{-5}	-3.16%
7	1.90×10^{-5}	-5.26%	1.79×10^{-5}	-10.64%	2.01×10^{-5}	0.33%
Mean Error		-1.97%		-3.16%		-1.49%

Table 7-7 Hardware Model Validation Case 1 Results for Tx_0

	Rx_0		Rx_1		Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
0	7.84×10^{-2}	-0.38%	7.83×10^{-2}	-0.43%	7.84×10^{-2}	-0.38%
1	5.60×10^{-2}	-0.43%	5.60×10^{-2}	-0.43%	5.60×10^{-2}	-0.48%
2	3.73×10^{-2}	-0.47%	3.73×10^{-2}	-0.63%	3.73×10^{-2}	-0.47%
3	2.28×10^{-2}	-0.52%	2.27×10^{-2}	-0.78%	2.27×10^{-2}	-0.78%
4	1.24×10^{-2}	-0.97%	1.24×10^{-2}	-0.89%	1.24×10^{-2}	-0.65%
5	5.88×10^{-3}	-1.21%	5.90×10^{-3}	-0.99%	5.91×10^{-3}	-0.79%
6	2.36×10^{-3}	-1.14%	2.35×10^{-3}	-1.69%	2.36×10^{-3}	-1.14%
7	7.52×10^{-4}	-2.68%	7.54×10^{-4}	-2.39%	7.61×10^{-4}	-1.47%
8	1.88×10^{-4}	-1.68%	1.80×10^{-4}	-5.56%	1.86×10^{-4}	-2.36%
9	3.37×10^{-5}	0.19%	3.12×10^{-5}	-7.25%	3.17×10^{-5}	-5.85%
Mean Error		-0.93%		-2.10%		-1.44%

Table 7-8 Hardware Model Validation Case 1 Results for Tx_2

	Rx_0		Rx_1		Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
0	7.82×10^{-2}	-0.55%	7.84×10^{-2}	-0.32%	7.84×10^{-2}	-0.33%
1	5.61×10^{-2}	-0.41%	5.61×10^{-2}	-0.39%	5.61×10^{-2}	-0.38%
2	3.73×10^{-2}	-0.55%	3.73×10^{-2}	-0.60%	3.73×10^{-2}	-0.55%
3	2.27×10^{-2}	-0.60%	2.27×10^{-2}	-0.87%	2.28×10^{-2}	-0.56%
4	1.24×10^{-2}	-0.89%	1.24×10^{-2}	-0.65%	1.24×10^{-2}	-0.65%
5	5.89×10^{-3}	-1.04%	5.90×10^{-3}	-0.87%	5.91×10^{-3}	-0.77%
6	2.35×10^{-3}	-1.73%	2.36×10^{-3}	-1.23%	2.35×10^{-3}	-1.65%
7	7.56×10^{-4}	-2.16%	7.50×10^{-4}	-2.88%	7.58×10^{-4}	-1.96%
8	1.82×10^{-4}	-4.82%	1.87×10^{-4}	-2.31%	1.85×10^{-4}	-3.36%
9	2.90×10^{-5}	-13.79%	3.18×10^{-5}	-5.34%	3.25×10^{-5}	-3.47%
Mean Error		-2.65%		-1.55%		-1.37%

Table 7-9 Hardware Model Validation Case 1 Results for Tx_3

	Rx_0		Rx_1		Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
0	7.83×10^{-2}	-0.50%	7.83×10^{-2}	-0.42%	7.83×10^{-2}	-0.41%
1	5.60×10^{-2}	-0.48%	5.60×10^{-2}	-0.45%	5.60×10^{-2}	-0.54%
2	3.73×10^{-2}	-0.55%	3.73×10^{-2}	-0.63%	3.73×10^{-2}	-0.47%
3	2.28×10^{-2}	-0.56%	2.27×10^{-2}	-0.65%	2.27×10^{-2}	-0.78%
4	1.24×10^{-2}	-0.89%	1.24×10^{-2}	-0.73%	1.24×10^{-2}	-0.73%
5	5.91×10^{-3}	-0.82%	5.89×10^{-3}	-1.01%	5.89×10^{-3}	-1.11%
6	2.36×10^{-3}	-1.39%	2.37×10^{-3}	-0.60%	2.36×10^{-3}	-1.02%
7	7.51×10^{-4}	-2.86%	7.64×10^{-4}	-1.19%	7.56×10^{-4}	-2.13%
8	1.82×10^{-4}	-4.72%	1.86×10^{-4}	-2.68%	1.83×10^{-4}	-4.30%
9	3.08×10^{-5}	-8.32%	3.11×10^{-5}	-7.43%	3.14×10^{-5}	-6.74%
Mean Error		-2.11%		-1.58%		-1.82%

Table 7-10 Hardware Model Validation Case 2 Results

	Rx_0		Rx_1		Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
3	1.51×10^{-2}	-0.11%	1.51×10^{-2}	-0.38%	1.51×10^{-2}	-0.18%
4	9.68×10^{-3}	-0.32%	9.63×10^{-3}	-0.80%	9.65×10^{-3}	-0.67%
5	5.98×10^{-3}	1.23%	5.97×10^{-3}	1.13%	6.00×10^{-3}	1.55%
6	3.60×10^{-3}	0.56%	3.62×10^{-3}	1.26%	3.58×10^{-3}	0.08%
7	2.07×10^{-3}	-1.68%	2.11×10^{-3}	0.08%	2.11×10^{-3}	0.17%
8	1.18×10^{-3}	-1.95%	1.20×10^{-3}	-0.62%	1.21×10^{-3}	0.37%
9	6.68×10^{-4}	-2.86%	6.74×10^{-4}	-1.99%	6.66×10^{-4}	-3.11%
10	3.62×10^{-4}	-5.01%	3.65×10^{-4}	-4.31%	3.65×10^{-4}	-4.20%
11	1.93×10^{-4}	-5.77%	1.92×10^{-4}	-6.41%	1.95×10^{-4}	-4.75%
12	1.01×10^{-4}	-6.04%	1.02×10^{-4}	-4.73%	1.03×10^{-4}	-3.89%
13	5.26×10^{-5}	-5.55%	5.38×10^{-5}	-3.43%	5.24×10^{-5}	-5.91%
14	2.80×10^{-5}	-3.80%	3.01×10^{-5}	3.45%	2.92×10^{-5}	0.22%
Mean Error		-3.09%		-1.56%		-1.95%

Table 7-11 Hardware Model Validation Case 3 Results

	Rx_0 & Rx_1		Rx_0 & Rx_2		Rx_1 & Rx_2	
SNR(dB)	BER	Error	BER	Error	BER	Error
0	7.63×10^{-3}	-0.34%	7.61×10^{-3}	-0.53%	7.61×10^{-3}	-0.56%
1	4.02×10^{-3}	-0.62%	4.00×10^{-3}	-1.07%	4.01×10^{-3}	-0.82%
2	1.96×10^{-3}	0.25%	1.96×10^{-3}	-0.01%	1.95×10^{-3}	-0.31%
3	8.90×10^{-4}	0.13%	8.91×10^{-4}	0.26%	8.84×10^{-4}	-0.48%
4	3.80×10^{-4}	-0.08%	3.82×10^{-4}	0.50%	3.73×10^{-4}	-1.92%
5	1.49×10^{-4}	-1.26%	1.48×10^{-4}	-2.19%	1.47×10^{-4}	-2.92%
6	5.55×10^{-5}	1.25%	5.29×10^{-5}	-3.46%	5.21×10^{-5}	-4.99%
7	1.92×10^{-5}	-4.06%	1.82×10^{-5}	-9.15%	1.85×10^{-5}	-7.85%
Mean Error		-0.59%		-1.95%		-2.48%

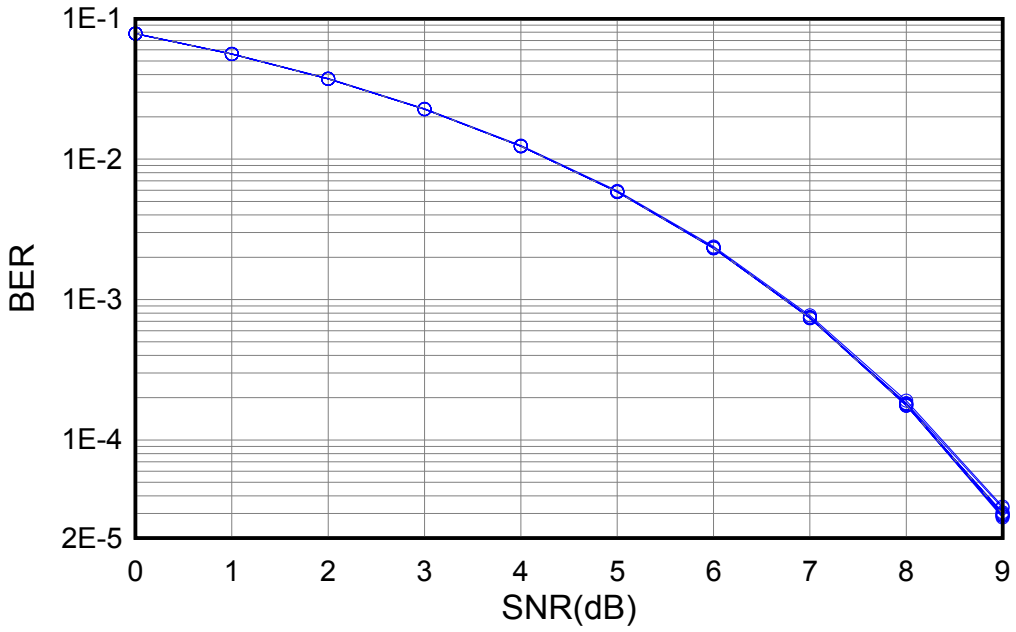

Figure 7-3 Overlay of Case 1 Data for Software Model

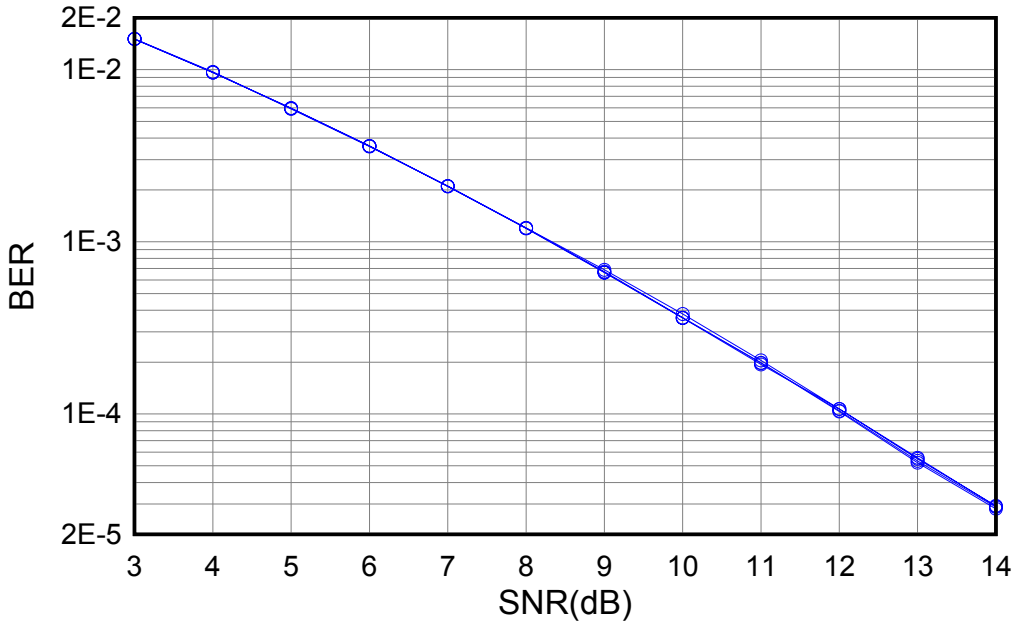

Figure 7-4 Overlay of Case 2 Data for Software Model

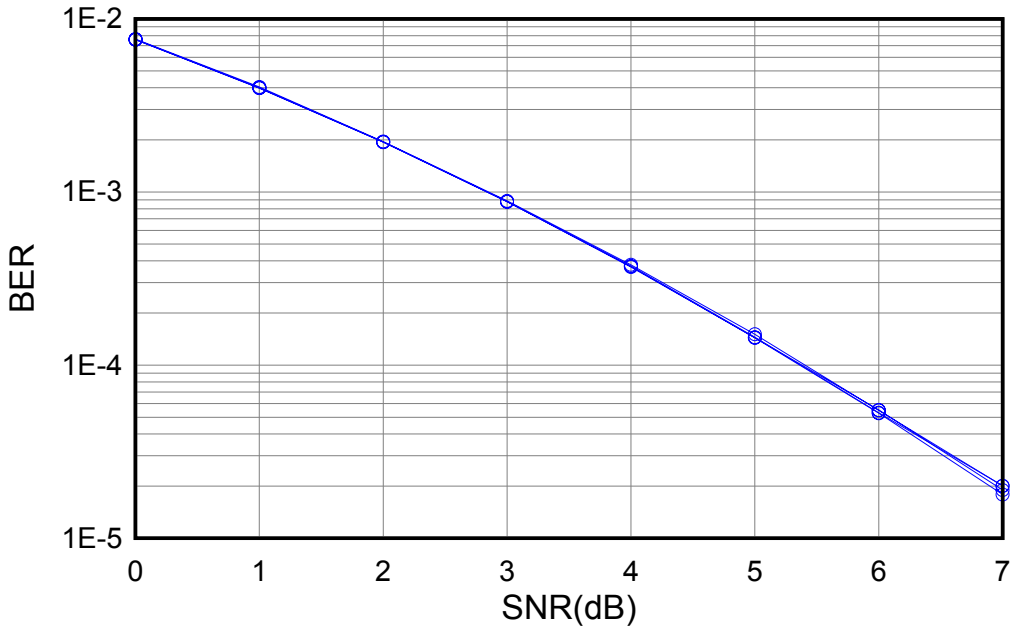

Figure 7-5 Overlay of Case 3 Data for Software Model

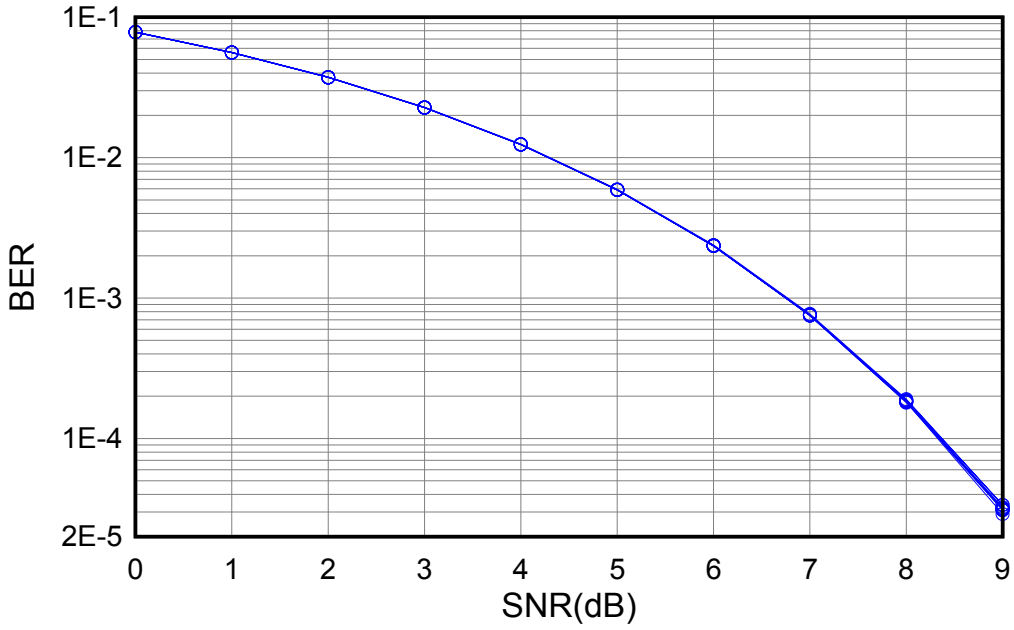

Figure 7-6 Overlay of Case 1 Data for Hardware Model

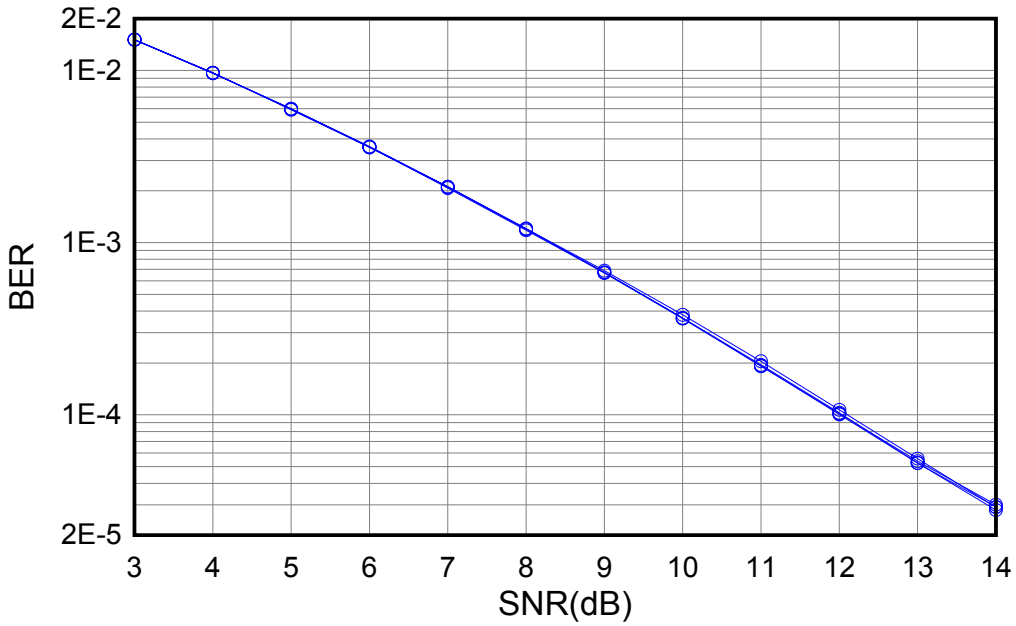

Figure 7-7 Overlay of Case 2 Data for Hardware Model

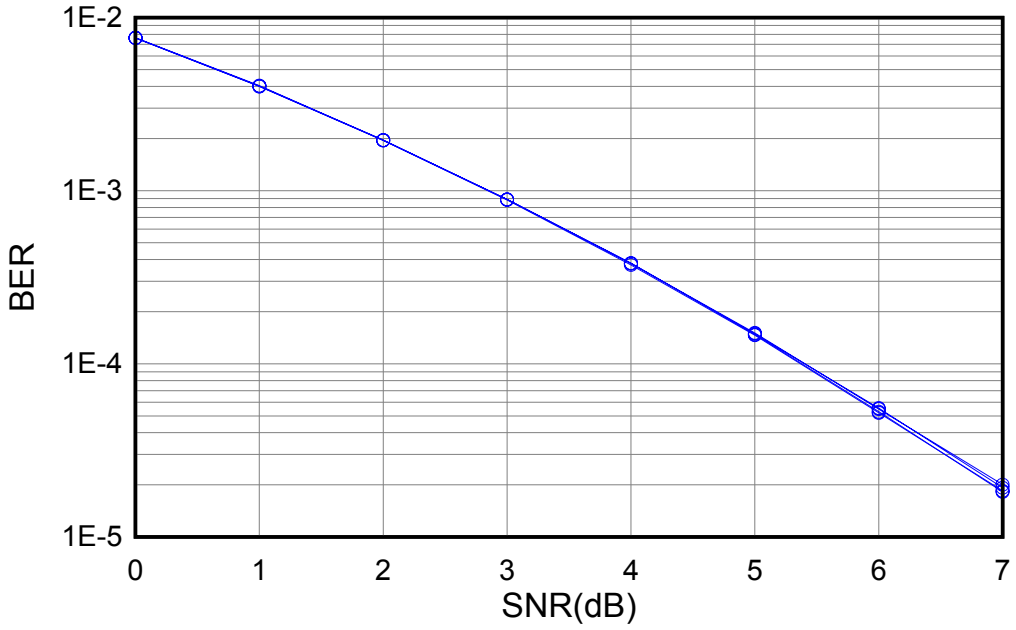

Figure 7-8 Overlay of Case 3 Data for Hardware Model

CHAPTER 8
RESULTS AND FUTURE DIRECTION

This chapter examines the resulting software and hardware models created for this dissertation research and illustrates their use in MIMO communication systems analysis. The models are examined in terms of their processing rate and the critical resources required for the hardware model. Example simulations are presented to show, first how the processing rate of the hardware based model allows simulation of performance curves for advanced MIMO codes. Second, examples are given to show the use of the models to examine the transition of MIMO system performance as individual channel paths fade out. The chapter concludes with a discussion of what was shown by the creation of these simulation models and an outline of directions for future research.

Processing Rate

One reason for creating the hardware MIMO system model is to achieve a greater processing rate than could be achieved with the software based model. Here the processing rate of each model is compared using actual time required to perform simulations. Both models are used to perform the complete set of simulations and the performance is analyzed in terms of processing time per bit and overhead per simulation.

The simulation test set is based on the three validation cases described in Chapter 7. Each simulation case is configured for 10 SNR points and is repeated 5 times; with every iteration the number of processed bits is increased. The time required to perform each simulation as record in the simulation results file are presented in Table 8-1 and Table 8-2.

Table 8-1 Software Model Performance Data

Bits per SNR (10^6 bits)	Case 1 - 10 SNR (seconds)	Case 2 - 10 SNR (seconds)	Case 3 - 10 SNR (seconds)
1	35.2	34.8	36.9
5	155.8	162.7	171.8
10	305.8	322.7	337.5
20	613.6	646.9	666.9
30	884.9	934.2	982.0

Table 8-2 Hardware Model Performance Data

Bits per SNR (10^6 bits)	Case 1 - 10 SNR (seconds)	Case 2 - 10 SNR (seconds)	Case 3 - 10 SNR (seconds)
1	113.8	114.2	113.0
5	116.9	117.6	117.7
10	123.1	125.1	123.2
50	178.1	174.4	176.0
100	236.3	237.7	233.3

In analyzing the performance data for the models, it is considered that the execution time of any simulation, T_{sim}, is composed of a mean processing time per bit, R, and a mean overhead per SNR point, O, so that execution time is given by (89) where B_T is the number bits processed per SNR point and N_{SNR} is the number of points. In the hardware model, the overhead is primarily due to the time required to download and initialize the hardware simulation model before starting the simulation. System Generator does this for each SNR point. In the software model, the overhead is primarily due to the sharing of the CPU with other unrelated processes executing on the computer. Therefore, the

overhead associated with the software model is more random in nature than is the overhead associated with hardware model.

$$T_{sim} = N_{SNR}\left(B_T R + O\right) \tag{89}$$

Fitting a first degree polynomial to the data of Table 8-1 and Table 8-2 and dividing the zero order coefficient by the number of SNR points yields the data of Table 8-3. It is clear from this data that the mean processing rate for the hardware model approaches the expected 120 nanoseconds per bit and that this rate is more than 23 times faster than the mean processing rate for the software model. The hardware model does however have an overhead of more than 11 seconds per SNR point compared to less than 1 second for the software model. This hardware model overhead is primarily due to the model being downloaded to the PGA by System Generator prior to each SNR point evaluation as mentioned above. This overhead could be eliminated in the future by redesign of the Simulink layer so that the hardware model remains resident in the PGA.

Even considering this additional overhead, the total execution time for a simulation using the hardware model is still less than that required for the same simulation using the software model when the bits per SNR point is above 5×10^6 as is observed in the data of Table 8-1 and Table 8-2.

Table 8-3 Simulation Processing Rate Data

		Hardware Model	Software Model
Case 1	Bit Rate (nsec)	126	2948
	Overhead (sec)	11.2	1.0
Case 2	Bit Rate (nsec)	125	3119
	Overhead (sec)	11.2	0.9
Case 3	Bit Rate (nsec)	123	3216
	Overhead (sec)	11.2	0.9

Resource Utilization

The Xilinx Vertex-4 XC4VSX35 FGPA provides limited internal resources for the construction of the hardware MIMO system model. Table 8-4 shows the resources required for the hardware model in relation to the resources available.

Table 8-4 Hardware Resource Utilization

Resource	Available	Initial		Skip-Ahead LFSR		Final	
DSP48	192	153	79%	149	77%	149	77%
RAM16B	192	170	89%	166	86%	58	30%
Total Slices	15360	12900	84%	12966	84%	12756	83%
SliceM	7680	2590	33%	2701	35%	2599	34%

There are four classes of internal resources; DSP48, RAM16B, Slices and SliceM. The DSP48 resources are computational blocks capable of performing 18-bit multiplications and additions. The DSP48 resources are primarily utilized for multiplication operations

within the hardware MIMO system model. The RAM16B resources are configurable memory blocks that are primarily utilized as look up tables. Slices are the basic logic resources and are used to create control logic, shift registers, storage registers, adders, comparators and any necessary interconnecting logic. Slices are organized in pairs consisting of a SliceL and a SliceM. Therefore, one half of all Slices are of the SliceM type. The SliceM resources have a special property in that they can implement a delay register. Delay registers are utilized throughout the model to synchronize signal transfer between processing blocks and ease timing constraints.

Table 8-4 shows the number of units in each resource class available on the XC4VSX35. Note that the Slices class includes both SliceL and SliceM types so that the number of units in the SliceM class is also counted in the Slices class. The next two columns in Table 8-4 show the typical resources utilized by the initial complete working MIMO system model. The initial hardware models did not include the skip-ahead LFSR or the composite lookup table described in Chapter 6. The first column indicates the number of units utilized from each resource class while the second column expresses the same information as a percentage of the total available resources in the class. The remainder of the data in Table 8-4 provides the resource utilization in number of units and percent of available units for the model after the introduction of the skip-ahead LFSR without composite lookup table and for the final model that includes both.

The data of Table 8-4 shows that the utilization of DSP48, Slices and SliceM resources remained relatively constant as the hardware model design is advanced. The utilization

of RAM16B resources dropped substantially from 166 to 58 units with the introduction of the composite lookup table. It should also be noted that no loss of performance was observed with the introduction of the composite lookup table design. The final implementation of the MIMO system model left a reasonable margin of 17% of available Slices, 23% of available DSP48 and 70% of available RAM16B resources remaining for the Xilinx Vertex-4 XC4VSX35 FPGA.

Using the MIMO Model

This section discuss how the of the MIMO system models created in this dissertation research can be utilized in the study of MIMO system performance. The discussion focuses on two areas. First, the use of the processing rate of the hardware model to evaluate the very low BER available from advanced MIMO codes is demonstrated. Second, the use of the models to study the transition of system performance under the condition of individual channel paths fade is examined. Examples are used to illustrate both areas.

Simulating with More Bits

One advantage of MIMO communications systems is the low BER that can be obtained for a given value of SNR relative to other communications systems. Consequently, it is necessary to simulate a larger numbers of bits to ascertain the BER versus SNR performance curves for MIMO systems compared to other communications systems. Simulating the number of bits required to get precise performance curves can be a deterrent to investigating performance at high values of SNR or to investigating many different scenarios due to the time required to process the simulation. The processing

rate of the hardware MIMO system model developed as part of this dissertation research addresses this issue.

Consider the performance curves presented in [10] that where used as the validation cases of this dissertation research. Note that the published performance curves utilized where limited to BER values of 1 x 10^{-6}. Note also that performance curves for a case of 3 receive antennas are not presented in [10]. Obtaining such data requires considerable computation time.

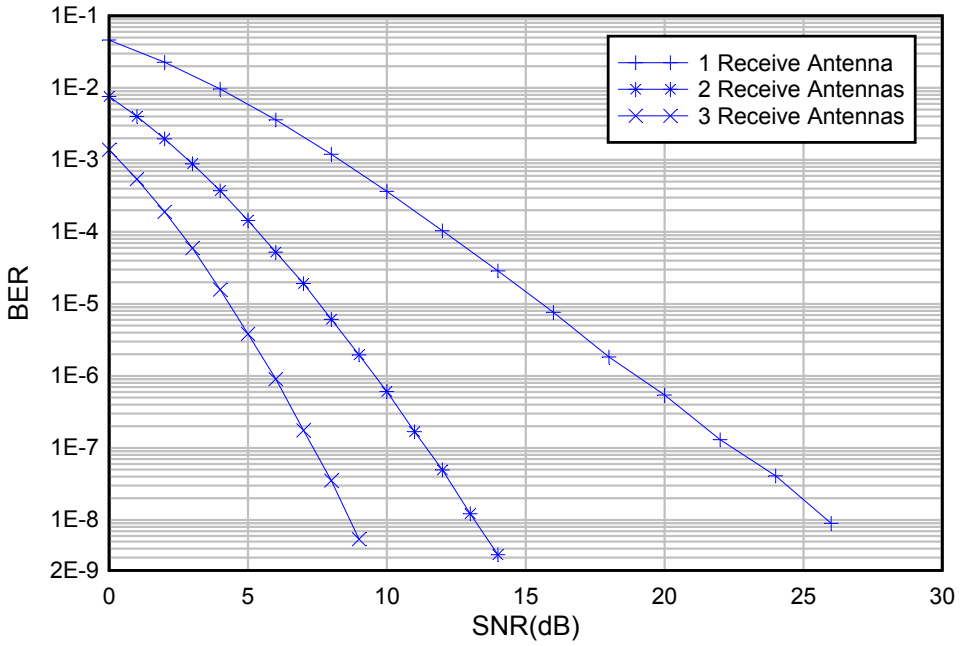

Figure 8-1 Extended Performance Curves

Using the hardware MIMO system model of this dissertation research the performance curves of Figure 8-1 are produced. The curves of Figure 8-1 show the BER versus SNR performance of the MIMO system for the case of 1, 2 and even for 3 receive antennas

under the same conditions as the curves presented in [10]. The 1 and 2 receive antenna cases are the validation test cases used in Chapter 7 but the SNR range here is extended to show BER values better than 1×10^{-8}. The case of 3 receive antennas shows the full performance capability of this STB code and the utility of the hardware simulation model.

In producing Figure 8-1 the SNR range for each curve was divided into two or three ranges. Each SNR range utilized the number bits per SNR point that provided reasonable precision while minimizing excess processing time. The ranges and bits per SNR point used for each curve are given in Table 8-5. The target precision for these curves was 1% so that approximately 100 errors per SNR are expected. The maximum number of bits that can be processed for any SNR point is limited by the model design to approximately 4.3×10^{9}; however this limitation can be easily increased in a future revision of the hardware model.

Table 8-5 Processing Ranges for Extended Performance Curves

	1 Receive Antenna		2 Receive Antennas		3 Receive Antennas	
	SNR (dB)	Bits	SNR (dB)	Bits	SNR (dB)	Bits
Range A	0 to 20	2.5×10^{8}	0 to 8	1.0×10^{8}	0 to 5	8.0×10^{7}
Range B	22 to 26	3.0×10^{8}	9 to 11	6.0×10^{8}	6 to 7	8.0×10^{8}
Range C			12 to 14	4.3×10^{8}	8 to 9	4.3×10^{9}

The total number of bits processed for each curve and the total processing time required are presented in Table 8-6. Note that approximately the same number of bits was required in each case and that it was possible to compute each curve in under a remarkable 30 minutes. In comparison, the last row of Table 8-6 presents the estimated time required for the simulation of the same curves using the software model. The estimates are based on the data of Table 8-3. The single receive antenna estimate assumes the processing rate for validation Case 2, while the estimates for the two and three receive antenna cases assume the processing rate for validation Case 3.

Table 8-6 Total Bits and Time for Extended Performance Curves

	1 Receive Antenna	2 Receive Antennas	3 Receive Antenna
Total Bits	1.2×10^{10}	1.5×10^{10}	1.1×10^{10}
Total Hardware Simulation Time	27 min	36 min	26 min
Estimate Time for Software Simulation	10.2 hours	13.8 hours	9.5 hours

Path Fading

Another use for the models created in this dissertation research is in the evaluation of the MIMO system performance when one or more signal channel paths fade. As described in Chapter 2 and Chapter 3, the development of MIMO systems assumes signals are transmitted over multipath channels. However, the published analyses typically assume probabilistic aggregate channel path characteristics. The models of this dissertation

research allow for additional attenuation and phase shift to be applied to each channel path. Actual receivers may enter zones of shading where the signal from one or more of the transmit antennas is blocked by a local obstruction. An actual receiver may also have a "keyhole" view of a transmit antenna that precludes one or more signal paths between transmit and receive antennas. In addition, mobile receivers may move into and out of such conditions.

Using the path attenuation feature of the simulation models, the MIMO system performance curves can be evaluated at a series of attenuations showing how the performance will change as a receiver passes through a zone of shading. Figure 8-2 through Figure 8-5 uniquely provide examples of changing performance under some path fading conditions as part of this dissertation research. In the examples the system is configured as for validation Case 2 with three transmit antennas and two receive antennas. In each case, the curves show performance of the system over a series of attenuations applied to one or more channel signal paths.

Figure 8-2, shows the performance change as one of the receive antennas is shaded. That is, all three channel signal paths terminating at the same receive antenna are attenuated together. Figure 8-3, shows the effect of obstructing a single channel signal path. Figure 8-4, shows the effect of obstructing two channel signal paths terminating at the same receive antenna. Finally, Figure 8-5 presents the effect of blocking all of the signal paths from one of the transmit antennas.

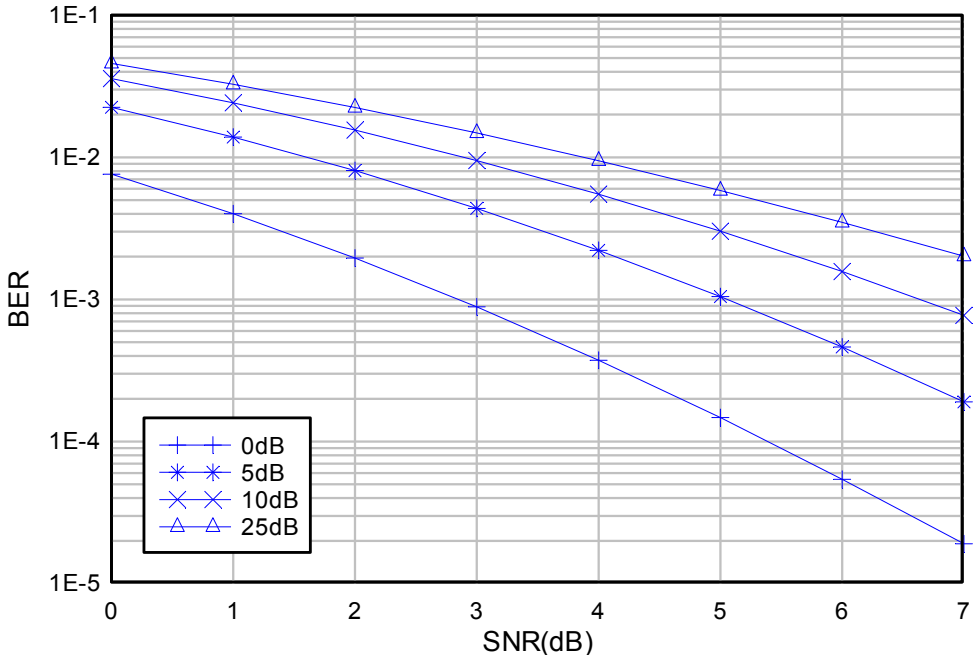

Figure 8-2 Shaded Receive Antenna

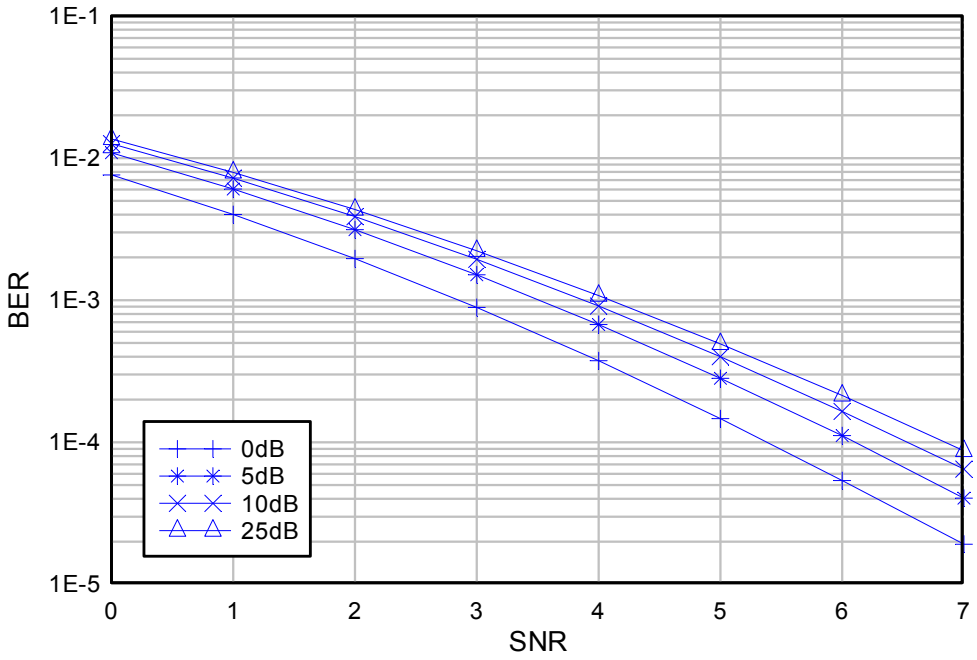

Figure 8-3 Single Blocked Channel Path

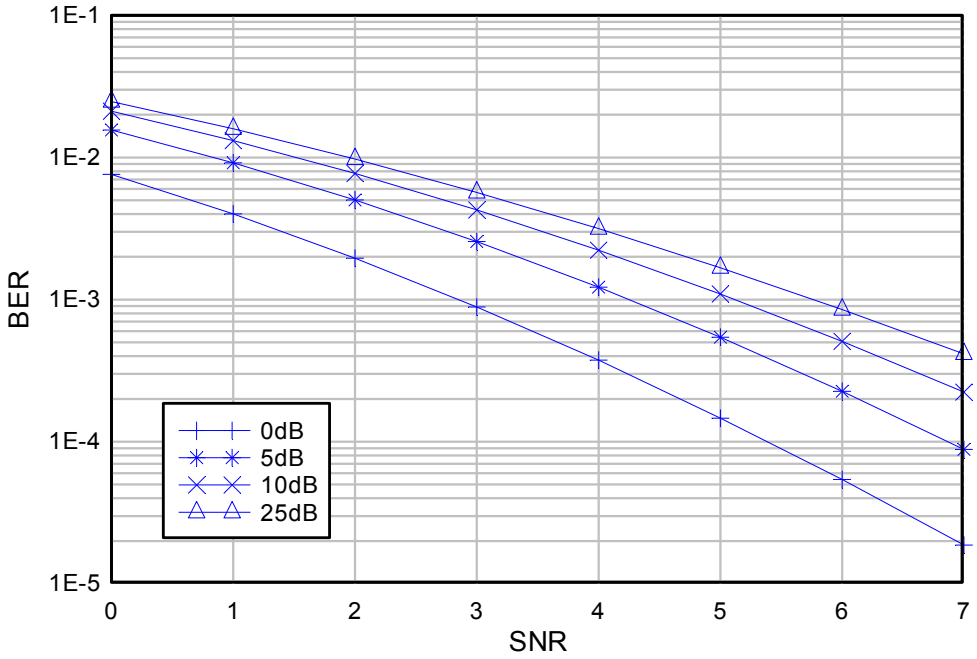

Figure 8-4 Two Block Channel Paths

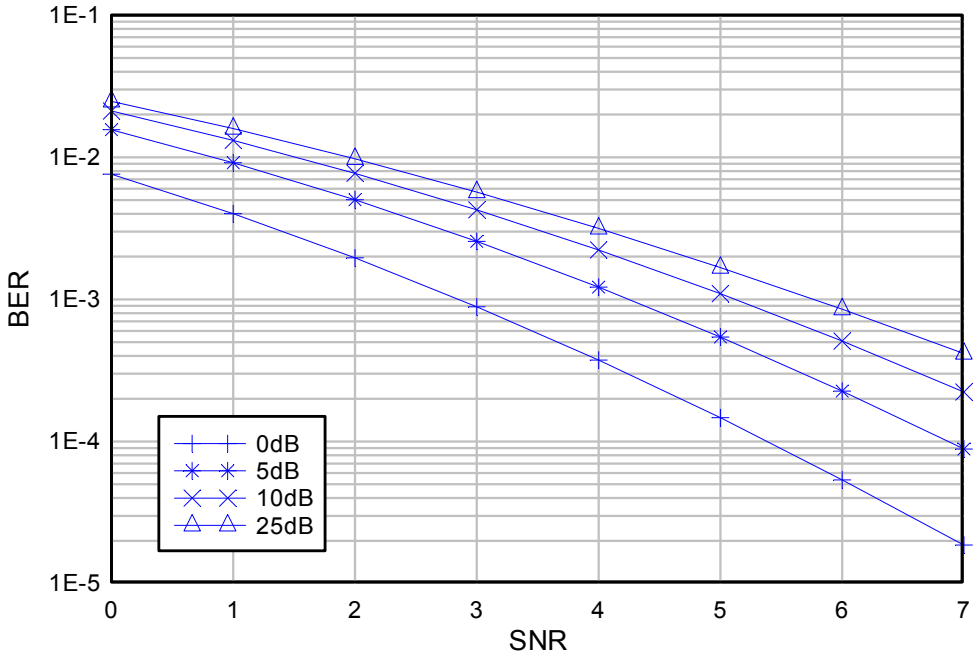

Figure 8-5 Obstructed Transmit Antenna

In Figure 8-2 through Figure 8-5 the 0 dB curve is the same and, in fact, is the Case 2 validation test curve. As may be expected, in all cases, as path attenuation increases the curves shift up and to the right indicating loss of performance; that is a greater BER at each SNR value. Also, as may be expected, the worst loss of performance occurs in Figure 8-2 where three signal paths are attenuated and least loss of performance occurs in Figure 8-3 where only a single path is attenuated. Figure 8-4 and Figure 8-5 show similar performance loss as each of these cases involve the attenuation of two signal paths.

Note that in all cases, the greatest loss of performance is observed with the first 5 dB step in path attenuation and that the change in performance decreases as the attenuation increase. This observation can be most readily seen in Figure 8-2. Consider first a BER level of 3×10^{-2} and note that there is a 2.2 dB loss of performance between the 0 dB and 5 dB attenuation curves or 0.44 dB loss of performance for each 1 dB of path attenuation. Now consider a BER level of 1×10^{-2} and note that there is 1 dB loss of performance between the 10 dB and the 25 dB attenuation curves or 0.07 dB loss of performance for each 1 dB of path attenuation. These observations suggest that MIMO system performance may be vulnerable to obstructed signal path conditions. Investigation of this phenomenon is a salient topic for suggested future study.

Path Phase Delay

The models allow for phase delay as well as attenuation to be applied to the channel signal paths. However, path phase delay is in fact canceled by the decoder and so shows no effect on the performance curves produced by the software and hardware simulation models. This can be seen by examining (83) and noting that, ignoring the noise term, the

transmitted symbol, x_0, is affected only by the square magnitude of the channel characteristic h. In the models presented here, the channel characteristic cancels out completely in the decoder because perfect knowledge of channel state information is assumed. Suggested future work could advance the models to include a method for estimating the channel characteristic from a training sequence or other means. At that time the error in phase estimation could affect the performance and thus the ability to apply phase delay becomes significant.

Summary of Accomplishments

This dissertation research has accomplished the following:

1. Develop both software and hardware based models for the study of MIMO communications system performance under conditions of signal path fading.

2. Demonstrate how hardware accelerated models can greatly shorten simulation time allowing for both the evaluation of lower BER levels and evaluation of more simulation scenarios than may otherwise be practical.

3. Develop a method to whiten the spectral content of an LFSR random number generator that can be utilized effectively in PGA hardware without incurring significant additional resource requirements.

4. Develop a method for generating Gaussian distributed random numbers in PGA hardware using reduced size lookup tables while allowing the trade off between table size and accuracy.

5. Develop an incremental linear MIMO decoder method that minimizes the requirements for storage and processing resources in PGA hardware, but is also useful in software based implementations.

6. Demonstrate how MIMO communications systems can be further analyzed for performance under the conditions of signal path fading and antenna shading.

7. Demonstrate the viability of a three antenna linearly decoded STBC MIMO communications system through its implementation and verification on a readily available PGA device.

Future Directions

There are seemingly two approaches that can be pursued in furthering the work presented here, the advancement of the MIMO system model and study of MIMO system performance under conditions of fading signal paths. The first approach involves both improving the efficiency and accuracy of the model implementation and extending the model design to incorporate the additional features described and in support of alternative STB codes. The second approach includes use of these or similar models to investigate MIMO system performance under the numerous combinations of conditions. It is reasonable to expect that the development along either approach will influence development along the other.

Advancing the Model

The models presented here can be advanced in two ways. First, the existing design can be made more efficient and accurate, especially in the hardware implementation. Some improvements of this type include:

1. Increasing the accuracy of the Gaussian number sources. The software model utilizes the Winitzki [41] approximation for inverse *erf()* function, which was selected in order to maintain comparability with the hardware model. The transformation from uniform to Gaussian distribution can be replaced with other more accurate methods. In the hardware model, the Gaussian distribution accuracy can be improved by using the composite LUT design process to create tables with a greater number of sample points.

2. The hardware model can be extended by redesign to support simulations requiring more than 4×10^9 bits allowing evaluation of lower BER conditions and allowing improved precision of currently obtainable BER values.

3. Elimination of overhead and development of a more convenient user interface for the hardware model. The overhead of loading the hardware model for each SNR point can be eliminated by modifying the implementation to allow the PGA configuration to remain resident when changing simulation conditions. The addition of graphical or other convenient user interface could also improve the usability of the model.

4. Port the hardware model to a more advanced PGA. The hardware model could be ported to more advance PGA such as the Xilinx Vertex-5 or Vertex-6. Porting the model can allow for even faster processing rates and open additional resources that can allow for expansion of the model.

The model design can be expanded to include additional capabilities. Some improvements of this kind include:

1. Addition of a channel characteristic estimator. An equalizer or another technique may be added to the model design to determine the channel path characteristics without the assumption of complete channel state information. The inclusion of such a mechanism in the model would allow the simulations to more closely approximate actual receiver operation.

2. Alternate decode and detector designs. The decoder and detector as well as other components of the model can be readily replaced with alternate methods or implementations with the resulting models used to evaluate the change in system performance.

3. Design of the models for additional STB codes. Space time block codes involving 2, 3 and 4 transmit antennas have been proposed [10] and others can be developed. Updating the models will allow other STB codes to be investigated using the principles presented here.

Study of MIMO System Performance

Employing the principles of attenuated signal paths presented here and using models such as those developed opens the possibility for the investigation of MIMO system performance under conditions of degraded signal paths. Here four degradation scenarios where presented to illustrate the types of analyses that can be performed using these models. However, many more such scenarios are possible including independent fading of two or more signal paths, as well as more detailed (greater number of attenuation values) simulations. As demonstrated here, such studies would require a large number of simulations with a large number of bits especially when advanced codes yielding low BER values are involved. Accelerated models such as the hardware based model developed here as part of this dissertation research can greatly assist such studies.

REFERENCES

[1] V. Tarokh, H. Jafarkhani & A. Calderbank, "Space-time block codes from orthoganl designs," *IEEE Transactions on Information Theory,* vol. *45*, no. 5, pp. 1456-1467, Jul. 1999.

[2] A. Paulraj, R. Nabar & D. Gore, *Introduction to Space-Time Wireless Communications*, United Kingdom: Cambridge University Press, 2003.

[3] C. Shannon, "A mathematical theory of communications," *The Bell Systems Technical Journal,* vol. *27*, pp. 379-423, 623-656, 1948.

[4] I. Telatar, "Capacity of multi-antenna Gaussian channels," *Technical Report #BL0112170-950615-07TM*, AT&T Bell Laboratories, 1995.

[5] A. Goldsmith, S. Jafar, N. Jindal & S. Vishwanath, "Capacity limits of MIMO channels," *IEEE Journal on Selected Areas in Communications,* vol. 21, no. 5, pp. 684-702, Jun. 2003.

[6] V. Rivera-Alvarez, D. Torres-Román & V. Kontorovitch, "On MIMO space-time coded systems: unleashing the spatial domain," *2nd International Conference on Electrical and Electronics Engineering (ICEEE) and XI Conference on Electrical Engineering (CIE 2005)*, pp. 467-470.

[7] G. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, Lucent Technologies, Inc, pp. 41-59, (autumn) 1996.

[8] G. Foschini, & M. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications,* Netherlands: Kluwer Academic Publishers, vol. 6, pp. 311-335, 1998.

[9] S. Alamouti. "A simple transmit diversity technique for wireless communications," *IEEE Journal on Select Areas in Communications*, vol. 16, no. 8 pp. 1451-1458*,* Oct. 1998.

[10] V. Tarokh, H. Jafarkhani, & A. Calderbank. "Space-time block coding for wireless communications: performance results," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 3, pp.451-460*,* Mar 1999.

[11] A. Calderbank, N. Seshadri, & V. Tarokh. "Space-time codes for wireless communications," *IEEE International Symposium on Information Theory*, p. 146, Jun. 1997.

[12] N. Seshadri, V. Tarokh, & A. Calderbank, "Space-time codes for wireless communications: code construction," *47th IEEE Vehicular Technology Conference,* vol. 2, pp. 637-641, May 1997.

[13] V. Tarokh, N. Seshadri, & A. Calderbank, (1998, Mar). "Space-time codes for high data rate wireless communication: performance criterion and code construction," *IEEE Transactions on Information Theory,* vol. 44, no. 2, pp.744-765, Mar. 1998.

[14] S. Caban, C. Mehlführer, A. Scholtz, & M. Rupp, "Indoor MIMO transmissions with Alamouti space-time block codes," *8th International Symposium on DSP and Communication Systems, DSPCS'2005 & 4th Workshop on the Internet, Telecommunications and Signal Processing, WITSP'2005*.

[15] P. Goud, C. Schlegel, R. Hang, W. A. Krzymien, Z. Bagley, S. Messerly, et al., "MIMO Channel measurements for an indoor office environment," *Wireless 2003 proceedings*, Calgary, Alberta, pp. 423-427, Jul. 2003.

[16] L. G. Barbero, & J. S. Thompson, "Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems," *ICC '06 IEEE International Conference on Communications*, vol. 7, pp. 3082-3087, Jun. 2006.

[17] K. S. Bialkowski, P. Uthansakul, M. E. Bialkowski, & A. Postula. (2007, Aug. 6). Design of MIMO test bed with an FPGA board for fast signal processing. [Online]. Available: http://epress.lib.uts.edu.au/dspace/bitstream/2100/86/1/130_Bialkowski.pdf.

[18] J. Dowle, S. H. Kuo, K. Mehrota, & I. V. McLoughlin, "An FPGA-based MIMO and space-time processing platform," *EURASIP Journal on Applied Signal Processing*, pp. 1-14, Jun. 2006.

[19] T. Koike, Y. Seki, H. Murata, Y. Susumu, & K. Araki, "FPGA prototyping of MIMO detector for over-1gps," *Proceedings of the XXVIIth URSI General Assembly*, 2005.

[20] C. Mehlfuhrer, M. Rupp, F. Kaltenberger, & G. Humer, "A scalable rapid prototyping system for real-time MIMO OFDM transmissions," *Second IEE/EURASIP Conference on DSP enabled Radio*, Sep. 2005.

[21] P. Murphy, F. Lou, & J. P. Frantz. (2007, Aug 8). A hardware test bed for the implementation and evaluation of MIMO algorithms. [Online]. Available: http://koala.ece.rice.edu/pubs/Mur2003Oct5AHardware1.pdf.

[22] Xilinx, Inc. (2006, May 24). *ML401/ML402/ML403 Evaluation Platform User Guide*, Publication Number UG080(v2.5). [Online]. Available: http://www.xilinx.com.

[23] Xilinx, Inc. (2007). *Development System Reference Guide 9.1i*. [Online]. Available: http://www.xilinx.com.

[24] The Mathworks, Inc., *MATLAB*®. [Online]. Available: http://www.mathworks.com/products/MATLAB.

[25] The Mathworks, Inc., *Simulink*®. [Online]. Available: http://www.mathworks.com/products/simulink.

[26] Xilinx, Inc. (2007, Mar.). *Xilinx System Generator for DSP User's Guide*, v9.1.01. [Online]. Available: http://www.xilinx.com/support/sw_manuals/sysgen_ug.pdf.

[27] P. Balaban, & J. Salz, "Dual diversity combining and equalization in digital cellular mobile radio," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 2, May 1991.

[28] A. Wittneben, "Basestation modulation diversity for digital simulcast," *41st IEEE Vehicular Technology Conference: Gateway to the Future Technology in Motion*, pp. 848-853, May 1991.

[29] J. Winters, J. Salz, & R. Gitlin, "The capacity of wireless communication systems can be substantially increased by the use of antenna diversity," *1st IEEE International Conference on Universal Personal Communications,* pp. 02.01.1-02.01.5, Sep.-Oct. 1992.

[30] N. Seshadri, & J. Winters, "Two signaling schemes for improving the error performance of frequency-division-duplex (FDD) transmission systems using transmitter antenna diversity," *43rd IEEE Vehicular Technology Conference*, pp. 508-511, May 1993.

[31] P. Wolniansky, G. Foschine, G. Golden, & R. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," *Proceedings of the ISSSE-98*, Pisa, Italy, Sep 29, 1998.

[32] S. Haykin, *Communications Systems, 4th Ed*., New York: John Wiley & Sons, Inc., 2001, pp. 309-468.

[33] W. C. Jakes, *Microwave Mobile Communications*, New York: John Wiley & Sons, Inc., 1974.

[34] J. Proakis, *Digital Communications, Third Edition*, New York: McGraw-Hill, 1995.

[35] D. Gesbert, M. Shafi, D. Shiu, P. Smith, A. Naguib, "From theory to practice: an overview of MIMO space-time coded wireless systems," *IEEE Journal on Selected Areas in Communications,* vol. 21, no. 3, pp. 281-302, Apr. 2003.

[36] E. G. Larsson, & P. Stocia, *Space-time Block Codes for Wireless Communications*, United Kingdom: Cambridge University Press, 2003.

[37] T. Marzetta, & B. Hochwald, "Capacity of a mobile multiple-antenna communications link in Rayleigh flat fading," *IEEE Transactions on Information Theory,* vol. 45, no. 1, pp.139-157, Jan. 1999.

[38] V. Tarokh, A. Naguib, N. Seshadri, & A. Calderbank, "Space-time codes for high data rate wireless communication: mismatch analysis," *IEEE International Conference on Communication: Towards the Knowledge Millennium,* vol. 1, pp. 309-313, Jun. 1997.

[39] Xilinx, Inc. (2007, Apr 10). *Vertex-4 User Guide*, Publication Number UG070(v2.2). [Online]. Available: http://www.xilinx.com.

[40] Xilinx, Inc. (2007, Jan 23). *Vertex-4 Family Overview*, Publication number DS112(v2.0). [Online]. Available: http://www.xilinx.com.

[41] S. Winitzki, (2008, Feb 6). A Handy Approximation for the Error Function and its Inverse. Unpublished. [Online]. Available: http://homepages.physik.uni-muenchen.de/~Winitzki/.

[42] S. Winitzki, "Uniform Approximations for Transcendental Functions," *Computational Science and Its Applications – ICCSA 2003*. Berlin, Springer, pp. 962-972, 2003.

[43] *CRC Standard Mathematical Tables and Formulae, 30th ed.*, CRC Press, New York, 1996, pp. 593-598.

[44] "IEEE Standard for Floating-Point Arithmetic," IEEE Standard 754, 2008

[45] Microsoft Corp. (2009, Jul. 9). *.NET Framework Class Library,* Microsoft Developer Network. [Online] Available: http://msdn.microsoft.com/en-us/library/system.random.aspx.

[46] The MathWorks Inc. (2009, Jul. 9). *MATLAB Documentation*. [Online] Available: http://www.mathworks.com/support/.

[47] Microsoft Corp. (2009, Jul. 9). *.NET Framework Class Library,* Microsoft Developer Network. [Online]. Available http://msdn.microsoft.com/en-us/library/system.math.aspx.

[48] L. Colavito & D. Silage, "An incremental PGA architecture for STBC MIMO decoding", *Sarnoff Symposium 2009*, IEEE, Mar. 2009.

[49] L. Colavito & D. Silage, "Efficient PGA LFSR implementation whitens pseudorandom numbers", accepted to *International Conference on Reconfigurable Computing and FPGAs*, IEEE, Dec. 2009.

[50] L. Colavito & D. Silage, "Composite look-up table Gaussian pseudo-random number generator", accepted to *International Conference on Reconfigurable Computing and FPGAs*, IEEE, Dec. 2009.

[51] M. Schollmeyer & W. Tranter, "Noise generators for simulation of digital communications systems", *Proc. 24th Annual Symp. Simulation*, New Orleans, LA., 1991, pp. 264-275.

[52] Xilinx, Inc, (2003, Mar.) Linear feedback shift register v3.0, DS257(v1.0), Included with Xilinx ISE Software available at http://www.xilinx.com .

[53] G. Marsaglia, "Random numbers fall mainly in the planes", *Proc. NAS*, vol. 61, pp. 25-28, Sep. 1968.

[54] F. C. Ionescu, "Theory and practice of a fully controllable white noise generator", *Int. Semiconductor Conf.*, Sinaia, Romania, vol. 2, pp. 319-322, 1996.

[55] P. P. Chu and R. E. Jones, "Design techniques of FPGA based random number generator", *2nd Annu. Military Aerospace Applicat. Programmable Devices and Tech. Conf.*, John Hopkins Univ., pp 28-30, Sep 1999.

[56] P. Hellekalek, "Good random number generators are (not so) easy to find", *Math. And Comput. In Simulation*, vol. 46, pp. 485-505, 1998.

[57] P. L'Ecuyer and S. Côté, "Implementing a random number package with splitting facilities", *ACM Trans. Math. Software*, vol. 17, no. 1, pp. 98-111, Mar. 1991.

[58] P. L'Ecuyer, "Software for uniform random number generation: Distinguishing the good and the bad", *Proc. 2001 Winter Simulation Conf.*, pp. 95-105.

[59] P. L'Ecuyer, "Uniform random number generators", *Proc. 1998 Winter Simulation Conf.*, pp. 97-104.

[60] R.R. Coveyou & R. D. Macpherson, "Fourier analysis of uniform random number generators", *Journal of the ACM*, vol. 14, no. 1, pp. 100-119, Jan. 1967.

[61] J.-L. Danger, A. Ghazel, E. Boutillon, & H. Laamari, "Efficient FPGA implementation of Gaussian noise generator for communications channel

emulation", *7th IEEE Int. Conf. Electronics, Circuits and Systems*, 2000, vol. 1, pp 366-369.

[62] Alimohammad, S. Fard, B. Cockburn, & C Schlegel, "A compact and accurate Gaussian variate generator", *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 5, pp. 517-527, May 2008.

[63] D. Lee, R. Cheumg, J. Villasenor, & W. Luk, "Inversion-based hardware gaussian random number generator: A case study of function evaluation via hierarchical segmentation", *IEEE Int. Conf. Field-Programmable Tech.*, 2006, pp. 33-40.

[64] D. Thomas, W. Luk, P. Leong, & J. Villasenor, "Gaussian random number generators", *ACM Computing Surveys*, vol. 39, no. 4, pp 11:1-11:38, Oct 2007.

[65] Ghazel, E. Boutillon, J.-L. Danger, G. Gulak, & H. Laamari, "Design and performance analysis of high speed AWGN communication channel emulator", *IEEE Pacific Rim Conf. Commun., Comput. And Signal Process.*, 2001, vol. 2, pp. 347-377.

[66] D. Lee, A. Gaffar, O. Mencer, & W. Luk, "MiniBit: bit-width optimization via Affine arithmetic", *Proc. 42nd Design Automation Conf.*, 2005, pp. 837-840.

[67] D. Lee, J. Villasenor, & P. Cheung, "Hierarchical segmentation schemes for function evaluations", Proc. *IEEE Int. Conf. Field-Programmable Tech.*, 2003, pp. 92-99.

[68] D. Lee, J. Villasenor, W. Luk, & P. Leong, "A hardware Gaussian noise generator using the Box-Muller method and its error analysis", *IEEE Trans. Comput.*, vol. 55, no. 6, pp. 659-671, Jun. 2006.

[69] D. Lee, W. Luk, J. Villasenor, & P. Cheung, "A Gaussian noise generator for hardware-based simulations", *IEEE Trans. Comput.*, vol. 53, no. 12, pp. 1523-1533., Dec. 2004.

[70] J. McCollum, J. Lancaster, D. Bouldin, & G. Peterson, "Hardware acceleration of pseudo-random number generation for simulation applications", *Proc. 35th Southeastern Symp. Syst. Theory*, 2003, pp. 299-303.

[71] Mitchell, M. (2009, Sep.). *Engauge Digitizer*, Ver. 4.1, [Online] Available: http://digitizer.sourceforge.net.

[72] The Mathworks, Inc., *Fixed-Point Numbers: Data Types and Scaling (Simulink® Fixed Point™).* [Online]. Available: http://www.mathworks.com/access/helpdesk/help/toolbox/fixpoint/index.html?/access/helpdesk/help/toolbox/fixpoint/ug/.