2021-09

A FIELD-PROGRAMMABLE GATE ARRAY IMPLEMENTATION OF A COGNITIVE RADAR TARGET RECOGNITION SYSTEM

Sessions, Calvin A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A FIELD-PROGRAMMABLE GATE ARRAY IMPLEMENTATION OF A COGNITIVE RADAR TARGET RECOGNITION SYSTEM

by

Calvin A. Sessions

September 2021

Thesis Advisor: Ric Romero
Co-Advisor: Douglas J. Fouts

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
A FIELD-PROGRAMMABLE GATE ARRAY IMPLEMENTATION OF A COGNITIVE RADAR TARGET RECOGNITION SYSTEM

5. FUNDING NUMBERS

6. AUTHOR(S) Calvin A. Sessions

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

The objective of this study is to design a field-programmable gate array (FPGA) implementation of a cognitive radar (CRr) target recognition system for electronic warfare applications. This thesis expands on the closed-loop adaptive matched waveform transmission technique called probability of weighted energy (PWE). This work also investigates the feasibility of applying the PWE technique in a functional digital hardware realization. Initially, a PWE Monte Carlo simulation model is developed in the Verilog hardware description language that is simulated in the Xilinx Vivado environment. The Verilog module components developed in the Monte Carlo model are then incorporated into a CRr target recognition system experiment utilizing the Xilinx VCU118 Evaluation Board. The VCU118 features the Virtex UltraScale+ high-performance FPGA to accomplish CRr adaptive waveform generation and transmission, digital signal processing requirements, and target classification. The Rohde & Schwarz SMW200A Vector Signal Generator and FSW Signal & Spectrum Analyzer function as the radar system transmitter and receiver, respectively, while the FPGA implementation enables the closed feedback loop used by the CRr.

14. SUBJECT TERMS
cognitive radar, probability of weighted energy, field-programmable gate array, target classification, eigenwaveform

15. NUMBER OF PAGES
99

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A FIELD-PROGRAMMABLE GATE ARRAY IMPLEMENTATION OF A COGNITIVE RADAR TARGET RECOGNITION SYSTEM

Calvin A. Sessions
Lieutenant Commander, United States Navy
BS, Western Washington University, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2021

Approved by: Ric Romero
Advisor

Douglas J. Fouts
Co-Advisor

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this study is to design a field-programmable gate array (FPGA) implementation of a cognitive radar (CRr) target recognition system for electronic warfare applications. This thesis expands on the closed-loop adaptive matched waveform transmission technique called probability of weighted energy (PWE). This work also investigates the feasibility of applying the PWE technique in a functional digital hardware realization. Initially, a PWE Monte Carlo simulation model is developed in the Verilog hardware description language that is simulated in the Xilinx Vivado environment. The Verilog module components developed in the Monte Carlo model are then incorporated into a CRr target recognition system experiment utilizing the Xilinx VCU118 Evaluation Board. The VCU118 features the Virtex UltraScale+ high-performance FPGA to accomplish CRr adaptive waveform generation and transmission, digital signal processing requirements, and target classification. The Rohde & Schwarz SMW200A Vector Signal Generator and FSW Signal & Spectrum Analyzer function as the radar system transmitter and receiver, respectively, while the FPGA implementation enables the closed feedback loop used by the CRr.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ADC	analog-to-digital converter
CLB	configurable logic block
CRr	cognitive radar
DAC	digital-to-analog converter
DSP	digital signal processing
EM	electromagnetic
EMS	electromagnetic spectrum
ES	electronic support
EW	electronic warfare
FIFO	first-in, first-out
FIR	finite impulse response
FPGA	field-programmable gate array
FSM	finite state machine
GPIO	General Purpose Input/Output
JEMSO	Joint Electromagnetic Spectrum Operations
IO	input/output
IP	intellectual property
IQ	in-phase/quadrature
LED	light emitting diode

LFSR linear-feedback shift register

LUT look-up table

LVDS low-voltage differential signaling

MAP maximum a posteriori

NATO North Atlantic Treaty Organization

PDF probability density function

PWE probability of weighted energy

QSFP+ Quad Small Form-Factor Pluggable+

RCS radar cross section

RF radio frequency

SNR signal-to-noise ratio

SoC System-on-Chip

SPI serial peripheral interface

TCL tool command language

UART Universal Asynchronous Receiver/Transmitter

VHDL Very High-Speed Integrated Circuit Hardware Description Language

Acknowledgments

I would like to thank my thesis advisors, Professor Douglas Fouts and Professor Ric Romero, for their guidance, mentorship, and expertise throughout my time at the Naval Postgraduate School. The combination of their industry and academic experience provided me with the immense support required for completing a hardware intensive, applications-oriented project.

I would also like to extend my gratitude to the Electrical and Computer Engineering Laboratory Director, Mr. Robert Broadston, and Radar Lab Manager, Mr. Manuel Badiola, for their laboratory test equipment assistance and access to project resources. For all hardware applications engineering support and technical contributions, I would like to thank Mr. PB Balasubramanium of Rohde and Schwarz.

Finally to my beautiful wife and high school sweetheart, Janet, and our two bright and handsome boys, Cayden and Camren, I would like to express my deepest appreciation for their continued love, energy, and strength during this process. I could not have achieved this accomplishment without their enduring support.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Joint military target identification and tracking systems require a high degree of confidence and reliability when maneuvering across the volatile electromagnetic spectrum (EMS). In Joint Publication 3-85 [1], the framework for Joint Electromagnetic Spectrum Operations (JEMSO) is established as an integrated necessity to achieve and maintain battlespace awareness within the operational environment in support of force protection, intelligence, and military objectives, as well as deconfliction and coordination efforts during peacetime operations. Specifically, electronic support (ES), a division of electronic warfare (EW), is delivered through radar system capabilities enabling target detection within the EMS "for the purpose of immediate threat recognition, targeting, planning" and execution of joint and combined military operations [1].

Challenges resulting from the increasing demand of conventional radar employment in the congested electromagnetic (EM) maneuver spaces, however, have led to the emergence of cognitive radar (CRr) research [2]. Interest in CRr systems has gained attention in modern radar development over the past 15 years [2] due to its fully adaptive, dynamic closed-loop capabilities that enable real-time decision-making within its environment through the employment of the cognitive remote sensing perception-action cycle [3]. In the recent North Atlantic Treaty Organization (NATO) technical study on CRr systems [4], the NATO Sensor Electronics Technology Task Group further validates the integration of cognitive radar sensing applications in military and peacekeeping missions as a solution to improve existing radar technology, as well as fostering new development of ES-related capabilities.

In this thesis, we expand on the CRr research developments conducted in [5]–[14] and focus on a hardware realization of a CRr target classification system and its proposed implementation into ES operations. Specifically, we incorporate the real-time adaptive matched waveform transmission technique, which is termed as probability of weighted energy (PWE), through the implementation of a high-performance field-programmable gate array (FPGA) system design.

1.1 Thesis Objective

The objective of this study is to design an FPGA implementation of a CRr target recognition system utilizing the PWE adaptive waveform transmission technique developed in [5], [8], [14].

This proof-of-concept study investigates the feasibility of incorporating PWE adaptive matched waveform theory into a real-time hardware application that is accomplished in two parts. In the first part of this study, a PWE Monte Carlo model consisting of 15,000 trials is developed, synthesized, and simulated in the Xilinx Vivado [15] environment using the Verilog hardware description language. This model incorporates a previously developed PWE MATLAB simulation [16] as a benchmark that is realized into a digital logic design.

In the second part of this study, a functional digital hardware implementation of the CRr target recognition system is demonstrated with the Xilinx VCU118 Evaluation Board and incorporates the Verilog module designs developed in the PWE Monte Carlo simulation model. The VCU118, featuring the Virtex UltraScale+ high-performance FPGA, is used to accomplish CRr adaptive waveform generation, digital signal processing requirements, and target classification. The Rohde & Schwarz (R&S) SMW200A Vector Signal Generator and FSW Signal & Spectrum Analyzer function as the radar system transmitter and receiver, respectively. Together, with the custom FPGA hardware design and R&S radio frequency (RF) equipment, a hardware-in-the-loop emulation of a CRr system with automatic target recognition is presented.

1.2 Thesis Organization

We first provide a conceptual background on cognitive radar systems and discuss the PWE adaptive matched waveform theory in Chapter 2. In Chapter 3, we describe the PWE Verilog-based Monte Carlo design process and summarize the Vivado simulation results. The CRr hardware implementation is then covered in Chapter 4, where we begin with a hardware and equipment overview. We later present the FPGA-based Verilog design process and functional proof-of-concept demonstration. At the end of Chapter 4, we report the target classification functionally of the CRr hardware design and experimental observations. Finally, in Chapter 5, we complete this body of work with a thesis summary, conclusion, and recommendations for future work.

CHAPTER 2:
Cognitive Radar for Target Recognition

2.1 Cognitive Radar Overview

In [3], the concept of a CRr is formally defined as a radar system comprised of the following attributes: (1) intelligent signal processing that "builds on learning through integration of the radar with the surrounding environment"; (2) feedback from the receiver to facilitate intelligence; and (3) preservation of received radar return information. While conventional radar systems may feature the limited ability to adapt to its changing environment based on processed received radar returns, the CRr dynamically incorporates the transmitter into a fully adaptive, closed feedback loop perception-action cycle, allowing the radar to: a) learn from return signal interaction with the EM environment and b) update the transmitted radar signal with respect to Bayesian decision inference and prior knowledge [3], [17], [18]. Shown in Figure 2.1 is a block diagram representation of the CRr perception-action cycle from [2].

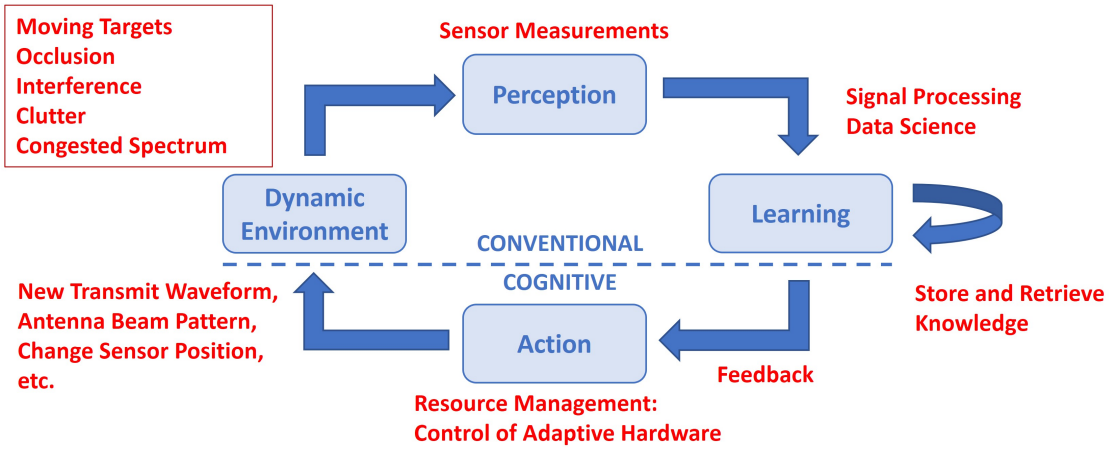


Figure 2.1. Block Diagram of the Cognitive Radar Perception-Action Cycle. Source: [2].

Fundamental components required to realize the CRr model include: (1) digital signal processing (DSP) capabilities with the real-time computational ability to measure and process environmental data; (2) digital memory to store and retrieve new observations and prior knowledge; and (3) radar transmitter/receiver hardware to support the adaptively reconfigured waveform synthesis and interaction process within the EMS [18], as depicted in Figure 2.2. Supporting hardware requirements include, but are not limited to, digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) for data/RF translation, as well as other radar-based communication system components.

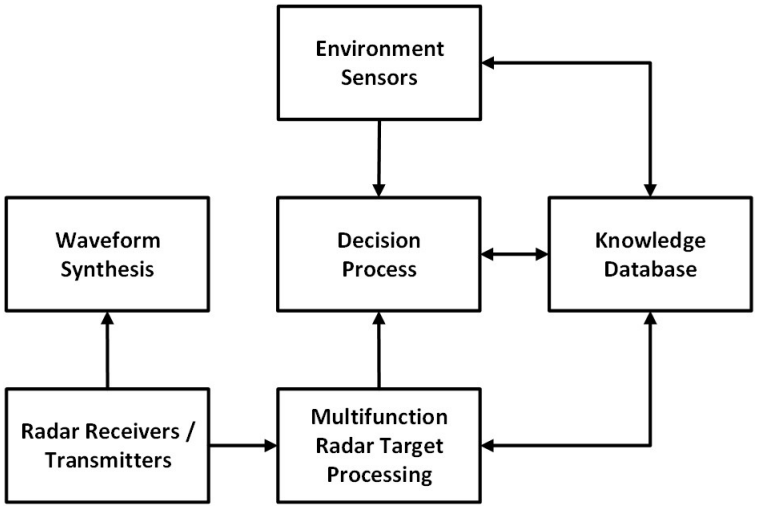


Figure 2.2. Basic Hardware Block Diagram of Perception-Action Cycle Model. Source: [18].

2.2 Probability of Weighted Energy Theory

The target recognition framework introduced by Goodman et al in [17] utilizes eigen-based adaptive waveform transmissions with sequential multiple hypothesis testing to facilitate confidence-based decisions. These concepts were further developed in [5]–[14], where one of the techniques was an improved signal-to-noise ratio (SNR)-based matched waveform technique called "probability of weighted energy" (PWE). Prior work demonstrates the PWE technique to be an effective target classification method, reducing latency, minimizing computational complexity, and eliminating search algorithms used in previously developed methods [5].

Also referred to as "probability of weighted eigenwaveforms" [11], the notion of PWE is derived in [5], [8], [14] and is summarized here for convenience. PWE facilitates target recognition through a sequence of multiple radar transmissions generated from the summation of known matched optimal waveforms under SNR (eigenwaveforms), $|X_i^{opt}(f)|^2$, that correspond to stored target responses [5]. Each eigenwaveform is individually weighted proportionally by their respective hypothesis probability, P_i. In the frequency domain, this transmit waveform is represented as $|X_T(f)|^2$, which is depicted in Figure 2.3 and is

$$|X_T(f)|^2 = \sum_{i=1}^{M} P_i |X_i^{opt}(f)|^2. \tag{2.1}$$

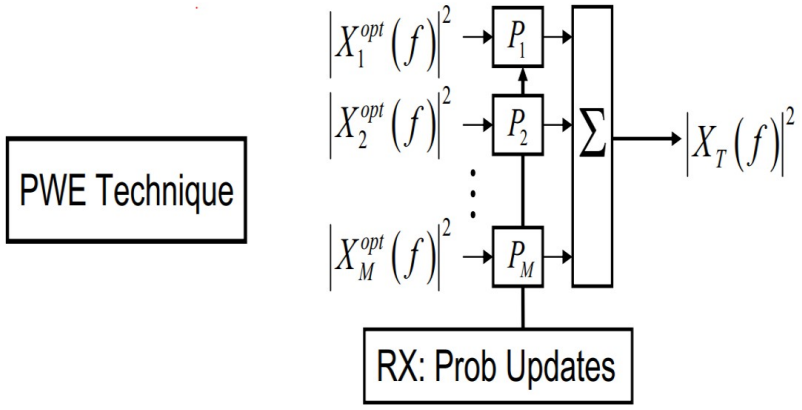


Figure 2.3. PWE Adaptive Matched Waveform Transmission Block Diagram. Source: [5].

The initial transmission waveform, assuming equal prior probabilities, consists of eigenwaveforms equally scaled by the probabilities $P_1 = P_2 = P_2 ... = P_M = 1/M$, where M represents the total number of stored target responses. Probability updates are calculated using Bayesian principles based on the observed interaction with the interrogated target via the radar return signal. The subsequent waveform is then recalculated, generated, and then re-transmitted. This process is repeated for multiple iterations that is pre-defined in the system. Upon processing the last radar return received signal, the CRr target recognition system classifies the interrogated target based on the most likely hypothesis dictated by the last probability update through maximum a posteriori (MAP) decision [5].

The PWE CRr closed feedback loop block diagram is depicted in Figure 2.4. Assuming normalized sampling time, let **x** represent the complex-valued adaptive transmit waveform, **y** be the received target return signal, **h** be the target response, and * be the convolution operation. The target return signal with additive Gaussian noise, **w**, is

$$\mathbf{y} = \mathbf{h} * \mathbf{x} + \mathbf{w}. \tag{2.2}$$

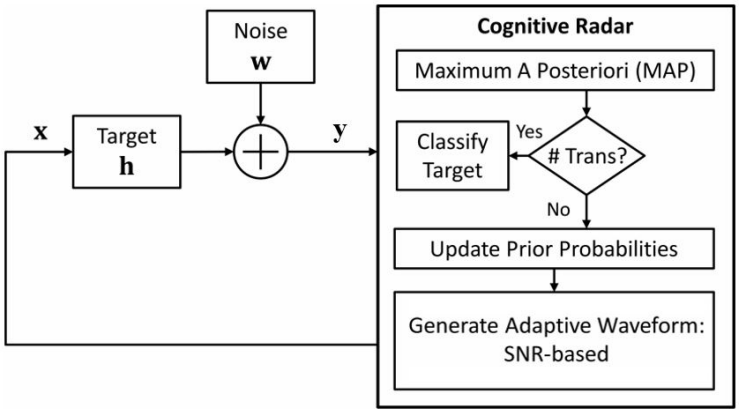


Figure 2.4. Cognitive Radar Closed Feedback Loop Block Diagram Using PWE. Source: [9].

From the CRr adaptive waveform generator of Figure 2.4, each optimal target eigenwaveform is derived from the autocorrelation, $\mathbf{R_H}$, of the target response convolution matrix, **H**, with respect to the known target response, **h**. The autocorrelation is

$$\mathbf{R_H} = \mathbf{H}^H\mathbf{H}, \tag{2.3}$$

where H denotes the Hermitian operation and **H** is

$$\mathbf{H} = \begin{bmatrix} \mathbf{h} & 0 & 0 & 0 \\ \cdot & \mathbf{h} & 0 & 0 \\ \cdot & \cdot & \cdot\cdot & \cdot\cdot \\ \cdot & & & \mathbf{h} \end{bmatrix}. \tag{2.4}$$

The target eigenwaveform produced from Equations 2.3 and 2.4 is represented as $\bar{\mathbf{q}}_{max}$ [8].

Using $\bar{\mathbf{q}}_{i,max}$, where $i = 1, 2, 3...M$, corresponding to the M target hypotheses, the initial transmit waveform of Equation 2.1 can be expressed in the discrete time domain as

$$\mathbf{x}_{\text{pwe}} = \sum_{i=1}^{M} \sqrt{P_i}\bar{\mathbf{q}}_{i,\max}. \tag{2.5}$$

This summation of individual eigenwaveforms is then normalized in relation to its energy, $E_{\mathrm{x}_{\text{pwe}}}$. To incorporate any transmit energy, E_x, the transmit waveform becomes

$$\mathbf{x} = \sqrt{E_x}\frac{\mathbf{x}_{\text{pwe}}}{\sqrt{E_{x_{\text{pwe}}}}}. \tag{2.6}$$

Referring to the received target return signal, $\mathbf{y}$, of Figure 2.4, each individual target hypothesis, $\mathcal{H}$, is expressed as

$$\begin{aligned}
&\mathcal{H}_1 : \mathbf{y} = \mathbf{x} * \mathbf{h}_1 + \mathbf{w} = \mathbf{H}_1\mathbf{x} + \mathbf{w} \\
&\mathcal{H}_2 : \mathbf{y} = \mathbf{x} * \mathbf{h}_2 + \mathbf{w} = \mathbf{H}_2\mathbf{x} + \mathbf{w} \\
&\mathcal{H}_3 : \mathbf{y} = \mathbf{x} * \mathbf{h}_3 + \mathbf{w} = \mathbf{H}_3\mathbf{x} + \mathbf{w} \\
&\ldots \\
&\mathcal{H}_M : \mathbf{y} = \mathbf{x} * \mathbf{h}_M + \mathbf{w} = \mathbf{H}_M\mathbf{x} + \mathbf{w}.
\end{aligned} \tag{2.7}$$

Probability updates are determined from the probability density function (PDF) corresponding with previous probability calculations, $P_{i,k}$, which are then used to generate the subsequent transmit waveform. The probability update is given by

$$P_{i,k+1} = \beta p\left(\mathbf{y}_{k+1}|\mathcal{H}_i\right) P_{i,k}, \tag{2.8}$$

where k represents the waveform iteration count, y_{k+1} is the most recent target return signal, and β ensures unit total probability [14]. Per Figure 2.4, when the last target return signal is received, target classification is determined through a MAP algorithm. In previous PWE works, 4 or more iterations have been used to perform target classification.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Verilog Modeling and Design Simulation

3.1 Verilog Model Overview

In this chapter, we build upon the PWE concepts introduced in Chapter 2. Prior to realizing the CRr target recognition system design in hardware, we develop, synthesize, and simulate a PWE Monte Carlo model in the Xilinx Vivado Design Suite using the Verilog hardware description language. This Monte Carlo simulation model, consisting of 15,000 trials, utilizes a previously developed PWE simulation implemented in MATLAB [16] as a benchmark and translates its algorithms into a digital system logic design.

In the first part of this chapter, we focus on the Verilog design parameters, high-level architecture, and finite state machine (FSM) algorithm. Individual Verilog module components are further detailed later in Section 3.2, as well as a description and derivation of the target hypothesis parameters. At the end of this chapter, we present the simulated Vivado results in Section 3.3. In this model development, we not only examine the PWE digital hardware logic design feasibility, but we also provide the Verilog building blocks for the FPGA implementation and hardware system integration, which is presented in Chapter 4.

3.2 Verilog Model Design Description

3.2.1 Performance Factors and Data Handling

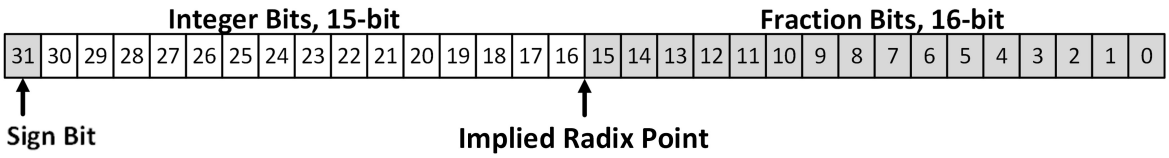
Several device performance factors are considered early in this Monte Carlo Verilog model development to facilitate the ultimate hardware design implementation objective. Device performance factors include: (1) data size and memory constraints; (2) timing-related limitations and latency; (3) hardware real estate and internal routing considerations; and (4) device capabilities. Specifically, this Verilog model incorporates the Xilinx VCU118 Evaluation Board parameters in Vivado based on its hardware performance, features, and test equipment compatibility that will be further detailed in Chapter 4.

Of the performance factors listed, data size is a major design consideration, serving as a critical standard that is incorporated into all Verilog modules. One of the challenges with adapting a hardware design from a computer-generated simulation modeled in a high-level programming language, such as MATLAB, is achieving the required precision and range with respect to data size. With MATLAB, the default data type is 64-bit, double-precision floating-point values ranging from -2.22507×10^{308} to 1.79769×10^{308} with 2.2204×10^{-16} accuracy [19]. In this hardware design, a 32-bit, signed fixed-point representation is selected to meet data precision requirements with the VCU118 and to interface with Rohde & Schwarz intellectual property (IP) core and proprietary data bus formats.

Using the signed 32-bit Q15.16 format [20], which is shown in the example of Figure 3.1, the sign of a generic fixed-point value, $x[31:0]$, is represented at the most significant bit (bit 31). Bits 0 through 15 represent the fractional portion, with the implied radix point set between bits 15 and 16. The integer portion is represented in bits 16 through 30. This data format provides a range from -32,768 to approximately 32,767.999985, with 0.000015 (2^{-16}) precision. Overflow and underflow handling is considered and incorporated throughout the design process. The appropriate scaling factors and Verilog code to generate a printed output in the Vivado Tool Command Language (TCL) Console is demonstrated at the bottom of Figure 3.1.

$Scaling\ Factor = 2^{-16}$

$Fixed\ Point\ Represntation: \quad x[31:0]\times2^{-16}$

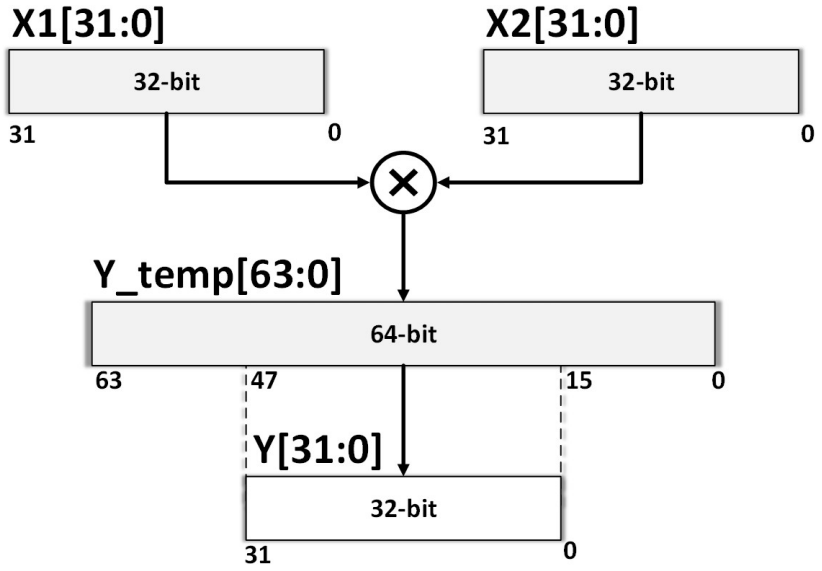



```
$itor($signed(x)*2.0**-16.0) // Verilog Test Bench Output
```

Figure 3.1. Q15.16, 32-bit Signed Fixed-Point Representation in Verilog

Another data processing consideration related to hardware logic designs adapted from high-level programming language models is the action of performing fundamental mathematical functions, such as multiplication, division, square root, and exponential expressions. Although Vivado IP cores can be used to perform such tasks, in this project, we design a set of mathematical Verilog modules from the bottom-up to control proper 32-bit Q15.16 data handling between Verilog instances to allow flexibility for design-specific functions. These Verilog modules will be discussed later in Section 3.2.4.

The process depicted in Figure 3.2 is performed for all 32-bit fixed-point multiplication expressions in this design. As illustrated, the product of two 32-bit values, which is generalized in this example as $X1$[31:0] and $X2$[31:0], yields a 64-bit result that is stored in a temporary register. To properly represent the product, this temporary value is shifted to the right by 16 bits and then stored into a 32-bit final product register, which is generalized here as Y[31:0]. This method provides an adequate approximation suitable for the accuracy requirements of this design. For designs requiring higher accuracy, an 8-bit shift can be performed on both the multiplicand and multiplier with rounding-error logic incorporated into the algorithm.




```
Y = (X1*X2)>>16 // Fixed Point Multiplication in Verilog
```

Figure 3.2. Q15.16, 32-bit Signed Fixed-Point Multiplication in Verilog

Note for all division functions where the divisor is constant, such as $x/4$ where x is a generic variable, fractional multiplication is incorporated for latency and hardware routing considerations, as well as simplicity. In such situations, fractional parameters are defined as constants in Verilog and implemented as multiplication functions, such as $0.25x$. The division module, to be discussed later, uses Euclidean iterative algorithms [21], which can be costly and generally less accurate in comparison to multiplying by known fractional constants.

3.2.2 Top Level Design Description

The system diagram shown in Figure 3.3 features the top level Monte Carlo simulation Verilog module, $Target\_Recognition.v$, which is instantiated under the testbench, $Target\_Recognition.tb$. The testbench delivers the clock, reset signal, and a 32-bit test input parameter to the target recognition module. $Target\_Recognition.v$ processes and generates eight 32-bit output values to the testbench that are defined as test points in Figure 3.3. All output values are translated in the testbench module and printed to the tool command language (TCL) console in Vivado for observation.

$Target\_Recognition.v$ manages all behavioral logic, data processing, and input/output (IO) handling through the FSM algorithm, which is further described in Section 3.2.3. As depicted in Figure 3.3, a total of 10 Verilog modules ($inst2 - inst11$) are instantiated under $Target\_Recognition.v$ to support all processing requirements. The main supporting modules featured are $Division.v$ ($inst2, inst12$), $Squareroot.v$ ($inst3, inst8 - inst11$), $Random\_Number\_Generator.v$ ($inst4$), $Exponential.v$ ($inst5$), $XPWE.v$ ($inst6$), and $FIR\_Filter.v$ ($inst7$). All modules are further detailed in Sections 3.2.4 through 3.2.8.

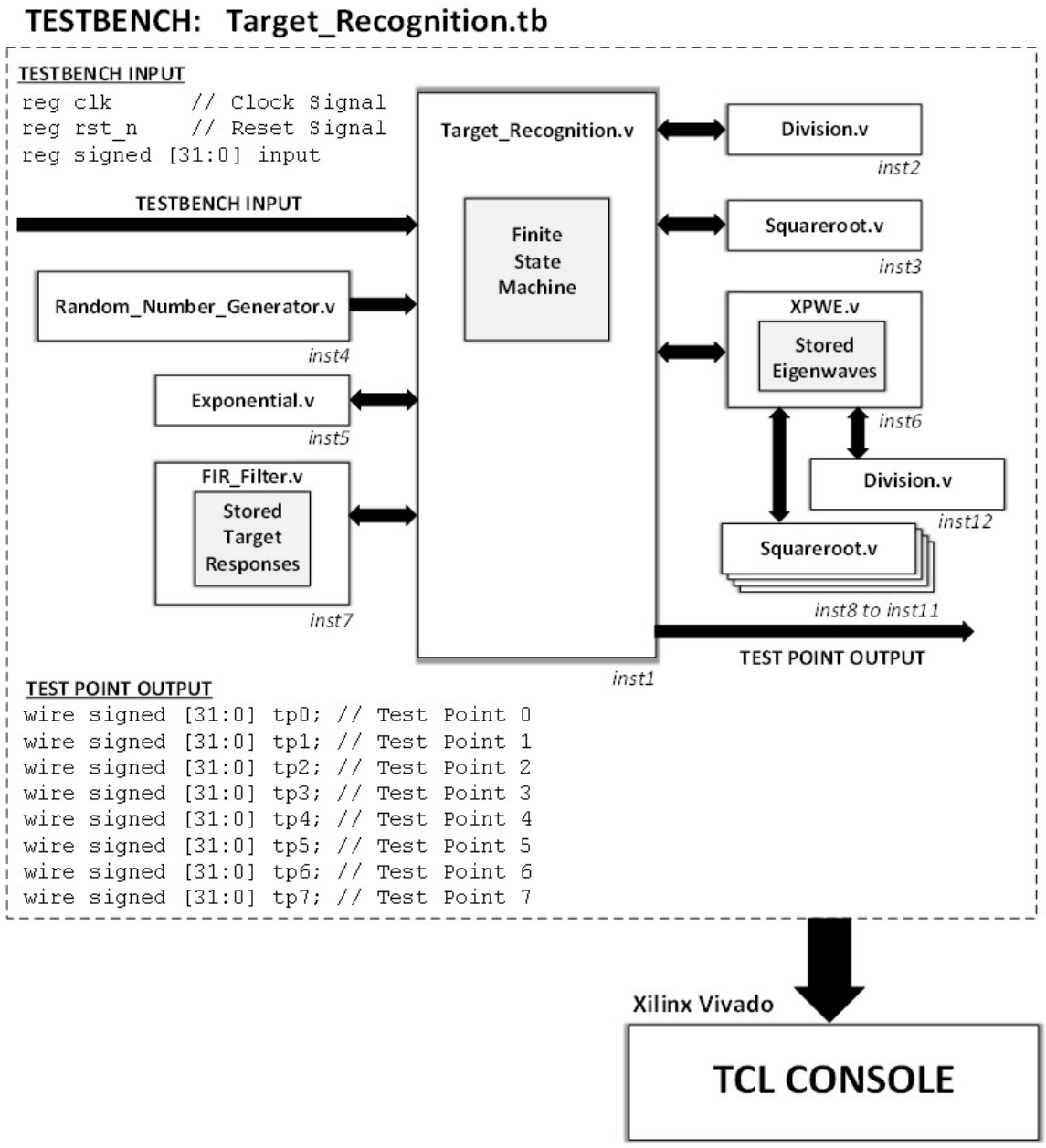


Figure 3.3. Top Level Block Diagram of PWE Target Recognition Monte Carlo Design in Verilog

As an overview, $Division.v$ provides 32-bit fixed-point division and fractional results using Euclidean iterative methods [21]. $Squareroot.v$ provides a 32-bit fixed-point square root approximation using an iterative algorithm involving calculating the nearest square [21]. $Random\_Number\_Generator.v$ is comprised of a 3-bit linear-feedback shift register (LFSR) to generate a random number to select a target hypothesis (in a Monte Carlo trial) and twenty 19-bit LFSRs to approximate and simulate in-phase and quadrature Gaussian noise using the Central Limit Theorem. $Exponential.v$ makes use of an elaborate look-up table (LUT) to quickly determine exponential functions. The PWE adaptive transmit waveform, $\mathbf{x}$, is generated in the $XPWE.v$ module, which holds the stored target hypothesis eigenwaveforms. Finally, the finite impulse response (FIR) module, $FIR\_Filter.v$, utilizes multiply-and-accumulate techniques to preform convolution functions. The four target responses are stored in this module.

3.2.3 Finite State Machine Description

Shown in Figure 3.4, the FSM incorporates the PWE classification technique into a Monte Carlo procedure. The system initially generates a PWE adaptive matched waveform, $\mathbf{x}$, which is transmitted at a set transmit energy, E_x. In a separate process, a simulated target return waveform, $\mathbf{y}$, is generated based on a randomly selected target response at the beginning of each Monte Carlo trial (as mentioned previously). After each transmission, the probability corresponding to each target hypothesis is calculated. The subsequent adaptive waveform is generated using the updated probabilities. This process is repeated for 4 adaptive transmissions. After processing the last target return, a MAP decision determines the target classification, which is then compared with the target that was selected in the first trial (purely for the purposes of evaluating classification performance). The probability of correct classification, P_{cc}, is calculated after 1,000 Monte Carlo trials for each increasing transmit energy, E_x. The transmit energy ranges from -30 dB units up to 10 dB units and is incremented into 15 levels for a total of 15,000 trials.

State	Description
0	System Initialization: Read Transmit Energy Level input from testbench Store into register *ks*
1	**FOR LOOP: *ks*[0:14], 15 iterations at increasing Energy Levels, E_x** Determine Square root of Energy Level scaling associated with ks value
2	Generate Adaptive Transmit Waveform, **x** Probability update input *ptheta*[0:3] = 0.25 **XPWE.v**
3	Read Results from XPWE Module
4	**FOR LOOP: *kms*[0:999], 1000 trials at each Energy Level, E_x** Assign Randomly Selected Target **Random_Number_Generator.v**
5	**FOR LOOP: *iter*[0:3], Transmit Adaptive Waveform for 4 iterations** Generate Complex-Valued Target Return: $\mathbf{s}_g = \mathbf{x}*\mathbf{h}$ convolution **FIR_Filter.v**
6	Read Results from FIR_Filter.v, $\mathbf{s}_g$
7	Add Noise to Target Return Signal, $\mathbf{y} = \mathbf{s}_g + \mathbf{w}$ **Random_Number_Generator.v**
8	**FOR LOOP: il = 0:3, 4 iterations for each Target Response** For each Target Response, Update: $\mathbf{S} = \mathbf{x}*\mathbf{h}_h$ convolution **FIR_Filter.v**
9	Read Results from FIR_Filter. **S**
10	Likelihood Calculation (a): SS = re[**S'S**]
11	Likelihood Calculation (b): b = 2re[**S'y**]
12	Likelihood Calculation (c): re[b]-SS **If il < 4, Return to State 8**
13	Calculate PDF value **Exponential.v**
14	Calculate Probability Update for each Target Hypothesis **Division.v**
15	Update Transmission Waveform, **x** **XPWE.v**
16	Read Results from XPWE Module Increment *iter* counter: **If *iter* < 4, Return to State 5**
17	Target Classification: Determine Max ptheta against *target_sel* Update error counter for each mismatch
18	Increment *kms* counter: **If *kms* < 999, Return to State 4**
19	Update Probability of Correct Classification, P_{cc} **Division.v**
20	Increment *ks* counter: **If *ks* < 14, Return to State 1**
21	**End Monte Carlo Simulation**

Figure 3.4. Finite State Machine Behavioral Algorithm for Monte Carlo Model in Verilog

The FSM in Figure 3.4 is comprised of 22 states. In State 0, following system initialization, the target recognition module receives the transmit energy value from the testbench input and stores the value in a designated register defined in Verilog as ks. The value of ks is then associated with the corresponding energy scaling value, $\sqrt{E_x}$ from Equation 2.5, which is determined by a LUT in State 1. Note that $Squareroot.v$ ($inst3$) was originally designed to perform this function, however, utilizing a LUT to represent $\sqrt{E_x}$ constants is not only faster, but yields greater accuracy in the overall performance.

In State 2, $\sqrt{E_x}$ and the probability update values, which are designated by P_i and $ptheta$[0:3] in Verilog, are sent to the $XPWE.v$ module to generate the adaptive transmit waveform, $\mathbf{x}$. Note that during this first transmission, each probability value is initialized as $ptheta[0:3] = 1/4$. The FSM then transitions to State 3, where it remains until $XPWE.v$ returns the complex-valued transmit waveform result, $\mathbf{x}$, which is defined in Verilog as registers s[30:0] and s_j[30:0].

In a parallel process, $Random\_Number\_Generator.v$ generates both in-phase and quadrature Gaussian noise values and a 2-bit pseudo-random number with every clock cycle. In State 4, the 2-bit output provides $target\_recognition.v$ with a random target selection value, $target\_sel$, associated with a target hypothesis response, $\mathbf{h}$. This $target\_sel$ value is sent to $FIR\_Filter.v$ in State 5, where the convolved result, $\mathbf{s}_g = \mathbf{x} * \mathbf{h}$, is stored into an array represented by 61 registers. The FSM subsequently transitions to and remains in State 6 until $\mathbf{s}_g$ is ready. The simulated Gaussian noise from $Random\_Number\_Generator.v$ is then added to $\mathbf{s}_g$ in State 7 to generate a simulated complex-valued target return waveform, $\mathbf{y}$, represented in Verilog as yy[60:0] = sg[60:0] + $Noise$[60:0] and yy_j[60:0] = sg_j[60:0] + $Noise\_j$[60:0], in accordance with Equation 2.7.

For iteration count, il[0:3], the complex-valued adaptive transmit waveform, which is represented in Verilog as s[30:0] and s_j[30:0], is convolved with each stored target response, hh[0:3][30:0], in States 8 and 9, and thus resulting in S[0:3][60:0] and S_j[0:3][60:0]. Using these outputs and the simulated complex-valued target response, the likelihood value is calculated in States 10 through 13 by evaluating each target hypothesis PDF, as depicted in Figure 3.4 and dictated in Equation 2.8 in Chapter 2. $Exponential.v$ is called in State 13 to generate the likelihood value and update $ptheta$[0:3] in State 14.

In States 15 and 16, the adaptive waveform, **x**, is re-calculated and transmitted via $XPWE.v$ using the updated $ptheta$[0:3] values. This process is repeated in States 5 through 16 for 4 iterations, as monitored by the iteration counter, $iter$. The FSM then transitions into State 17 after the last target return waveform is processed.

In State 17, target classification is determined by the system from the final $ptheta$[0:3] values and then compared against $target\_sel$ that was randomly selected in State 4. The error counter, err, is incremented for every incorrect target classification result. States 4 through 18 are repeated for 1000 trials before transitioning to State 19 to calculate P_{cc}, where $P_{cc} = 1 - (err/1000)$. States 1 through 19 are repeated until 15 E_x levels are complete. State 20 increments and evaluates the E_x levels to be used, until all of the 15,000 trials are simulated. The transition into State 21 marks the conclusion of the Monte Carlo simulation. P_{cc} and target classification errors for each E_x level are printed out onto the Vivado TCL Console.

3.2.4 Verilog Module Description: Mathematical Functions

As mentioned previously in Section 3.2.1, division, square root, and exponential math functions are developed in this project to support signed 32-bit fixed-point operations while preserving Q15.16 data formatting between modules. Figures 3.5 and 3.6 provide detailed representations of the Verilog modules as schematic diagrams, which are depicted as $Division.v$, $Squareroot.v$, and $Exponential.v$.

Note that the inter-module communication standard applied throughout this project incorporates a device $enable$ signal originating from the module requesting service that is accompanied by input data. A device $complete$ signal and the processed output data are then returned to the originator from the service-providing module upon process completion. The requesting module sets the $enable$ signal for the duration of the process and then subsequently de-asserts the $enable$ signal upon acknowledging receipt of the $complete$ signal. Specific to the math modules shown in Figures 3.5 and 3.6, the $enable$ ports are represented as $division\_en$, $sqrt\_en$, and $exponential\_en$. The device $complete$ ports are depicted as $division\_complete$, $sqrt\_complete$, and $exponential\_en$. All modules operate with respect to the rising edge of the clock input.

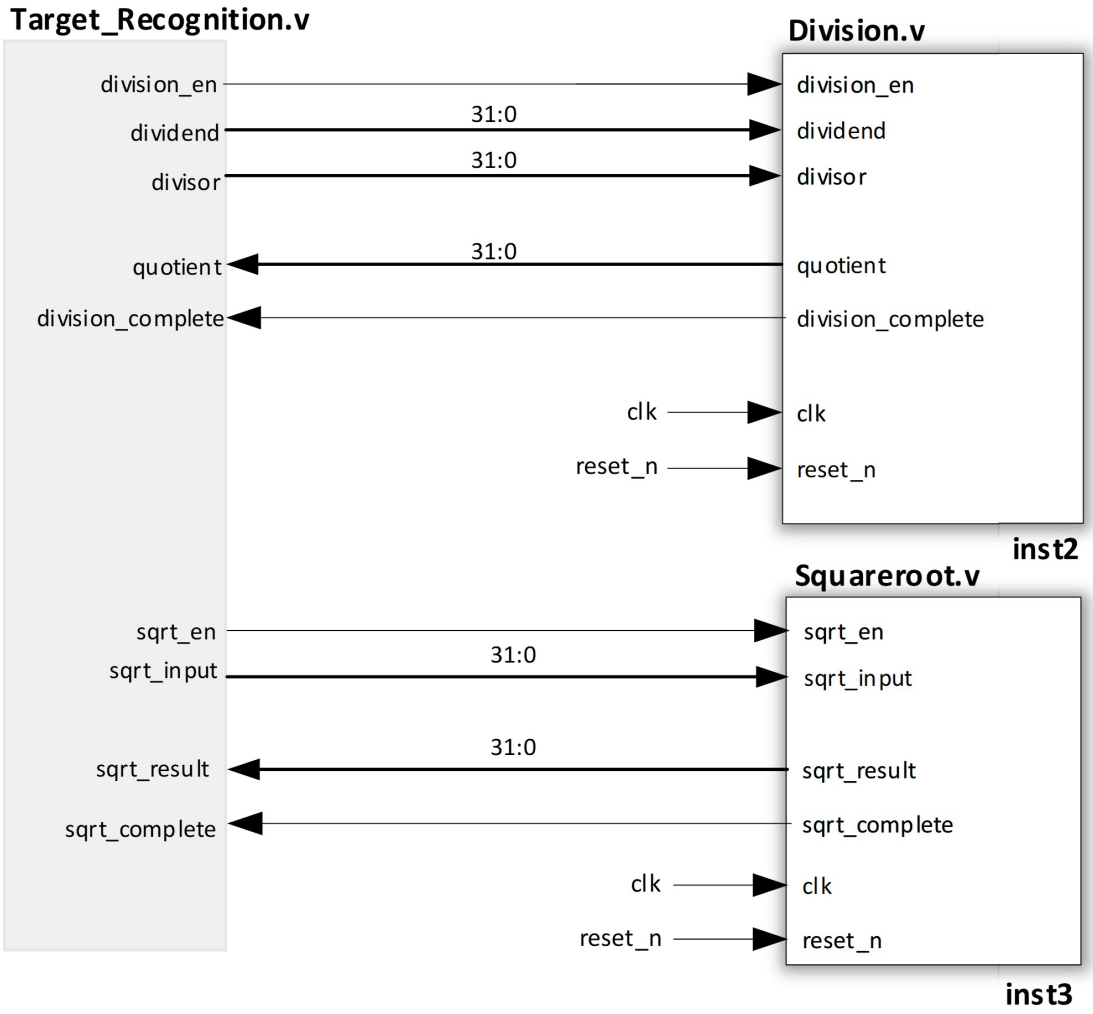


Figure 3.5. Schematic Diagram of Division and Square Root Verilog Modules

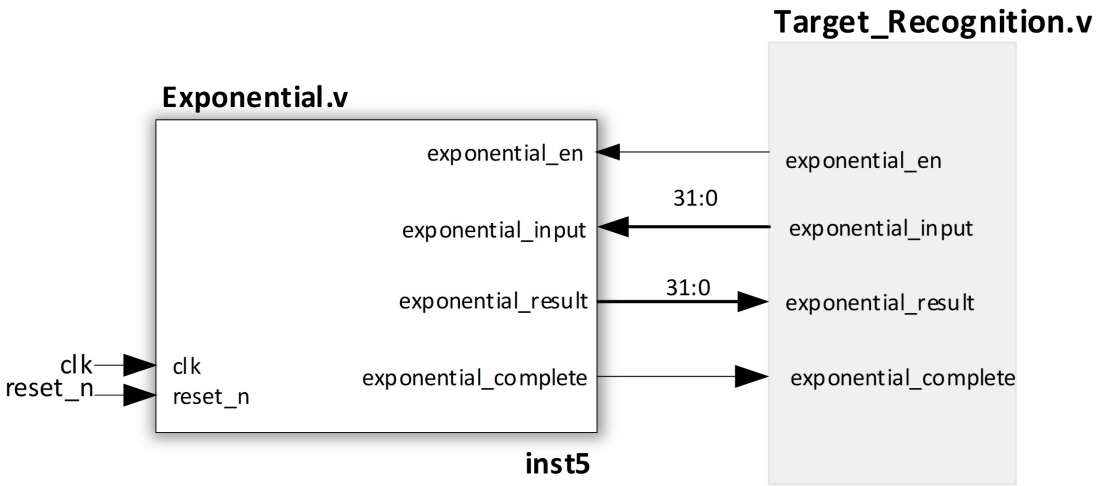


Figure 3.6. Exponential Verilog Module Schematic Diagram

Shown at the top right portion of Figure 3.5, the *division.v* module receives *dividend* and *divisor* input values as signed 32-bit data and outputs a signed 32-bit *quotient* up to three decimal places. The division procedure incorporates Euclidean iterative processes to approximate integer and fractional calculations [21]. The division module returns $quotient = 0$ for all cases where either the $dividend = 0$ or the fraction is less than 0.001. If the $divisor = 0$ or is less than or equal to 3.0518×10^{-5}, the division module returns the maximum Q15.16 system value, approximately 32,767.999985, to represent infinity. This module is primarily used to calculate probability updates and $XPWE.v$ scaling.

In the square root process, shown at the bottom right of Figure 3.5, a simple counter and iterative multiplication approach are used to estimate the square root of the data input received, $sqrt\_input$, by calculating the nearest square [21]. As the iteration counter, k, increments, the square of k, denoted as $k\_product$, is calculated and compared to the input value. The scaled resulting output, $sqrt\_result$, is generated if $k\_product$ is greater than or equal to the input. Module inputs less than or equal to zero are designed to return a zero value, as $Squareroot.v$ does not support calculating imaginary outputs. This function is incorporated in $XPWE.v$ for calculating the probability update scaling, per Equation 2.5.

The square root module outputs an approximation, accepting inputs up to 31-bits where each k count used to determine the nearest square rounds to increments of 2^{-16}. The average percent error is approximately 5%, which of course should be dictated by practical system requirements. We deem this error acceptable in this work. For higher value inputs, such as $\sqrt{10}$, errors are observed to be approximately 0.06%, while smaller input values, such as $\sqrt{0.001}$, yield approximately 11.25% error. Although the algorithm is simple to implement, the associated latency costs accumulated is a drawback. Future work requiring greater precision at increased speed can incorporate algorithms, such as the "Babylonian Method." This would require designing a more precise division algorithm.

As illustrated in the schematic of Figure 3.6, $exponential.v$ receives a signed 32-bit input, $exponential\_input$ and delivers the 32-bit output, $exponential\_result$. The output is generalized by the function e^x, where the variable x is again used to describe a generic input value. This method uses two sets of LUTs to handle positive and negative input values that are 665 and 709 elements wide, respectively. A MATLAB algorithm was developed offline to determine required values stored in the LUTs. Because of the risk of data overflow or underflow related to signed 32-bit ranges, input values $x \geq 10.39063$ yield output $e^x = 32,553.00621$. For values $x \leq$ -11.078, the module outputs 1.54465×10^{-5}. These ranges and approximations are deemed acceptable for the performance in this design application, which typically results in less than 1% error. The exponential module is used to calculate the likelihood value in probability update calculations, as shown in Equation 2.8.

Despite the speed, simplicity, and acceptable accuracy of incorporating a LUT method, cost considerations related to the circuitry and/or memory expended may be a factor for more complex designs. To free up register usage, simple case statements are utilized to develop the LUTs. For future work, it is recommended to implement the "Coordinate Rotation Digital Computer" (CORDIC) algorithm for increased performance, accuracy, and design efficiency [21].

3.2.5 Verilog Module Description: Random Number Generator

The pseudo-random number generator is depicted in the schematic diagram shown in Figure 3.7. Two different LFSR designs are implemented to generate the randomly selected target and complex-valued noise that were previously described in Section 3.2.3. *Random_Number_Generator.v* runs independently from the FSM and generates a 2-bit pseudo-random output, *random*, and two 32-bit pseudo-Gaussian outputs, *noise* and *noise_j*, at each rising clock edge input.

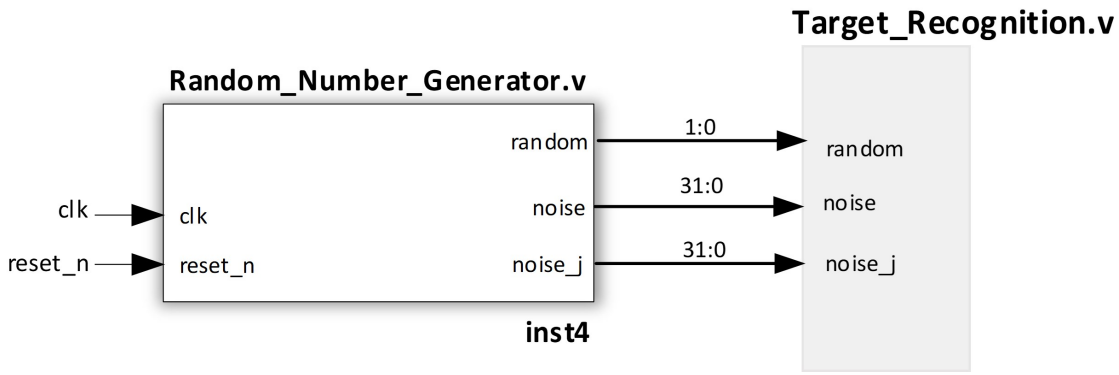


Figure 3.7. Pseudo-Random Number Generator Verilog Module Schematic

Figure 3.8 illustrates a simple 3-bit LFSR design inspired by [22], which generates a pseudo-random counting sequence by feeding bits 0 and 2 of the 3-bit input register into an Exclusive OR gate that is routed to bit 1 of the 3-bit output register. As shown in the diagram, bits 2 and 1 of the input register are assigned directly to bits 2 and 0 of the 3-bit output, respectively. A second Exclusive OR gate with inputs originating from bits 2 and 0 of the 3-bit output register is then routed to bit 0 of the 2-bit output register. The output of the first Exclusive OR gate from the 3-bit input register provides the assignment for bit 1 of the 2-bit output register.

Upon system initialization, the 3-bit input register is loaded with a seeded value and the counter updates with respect to the clock. The 2-bit pseudo-random register is routed to the *Random_Number_Generator.v* output, *random*, which is fed directly into the target recognition top-level model. This output is only read, however, during State 4 of the FSM shown in Figure 3.4 and detailed in Section 3.2.3.

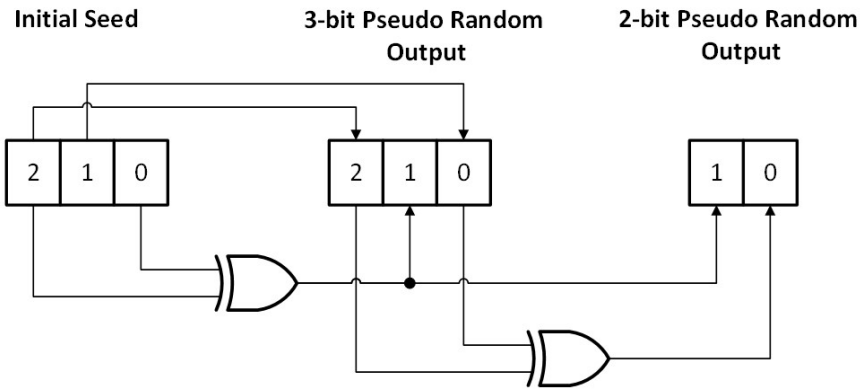


Figure 3.8. 2-bit and 3-bit Pseudo-Random Number Generator, Verilog Schematic Representation

Using the concepts from the 3-bit pseudo-random number generator design of Figure 3.8, the 19-bit LFSR of Figure 3.9 is developed to achieve the desired range. In this design, bits 0 through 17 of the initially seeded 19-bit input register are assigned to bits 1 through 18 of the 19-bit pseudo-random output register. Bits 18 and 16 of the input register are fed into an Exclusive OR gate, in which its output is routed to bit 0.

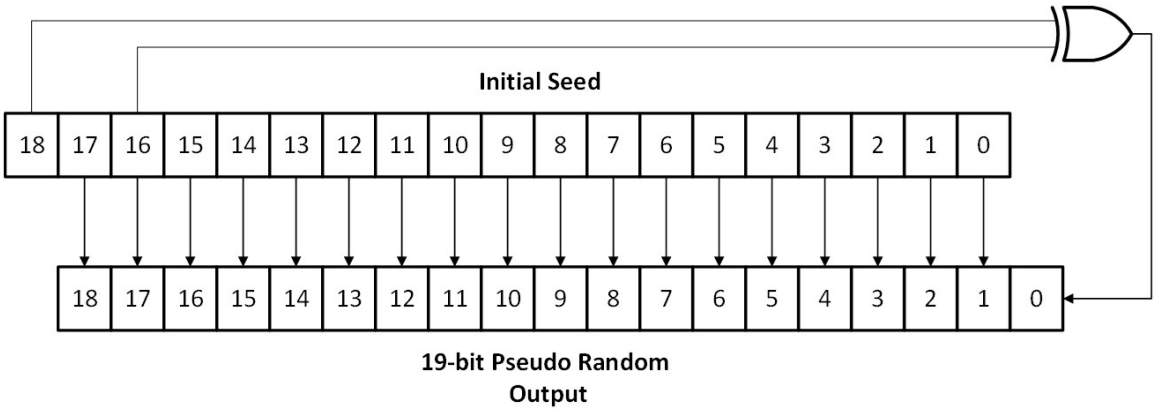


Figure 3.9. 19-bit Pseudo-Random Number Generator, Verilog Schematic Representation

To simulate complex-valued Gaussian noise, this 19-bit model is duplicated 20 times, each seeded with different input values. A set of 10 LFSRs is assigned to both the in-phase noise generator and quadrature system, modeled in Figure 3.10. Using the Central Limit Theorem to estimate a Gaussian distribution, the 10 LFSR inputs are accumulated in a summation block, then averaged and scaled, as represented in the diagram with $1/10\sqrt{2}$. The mean of this distribution is observed to be approximately 2.8 V. As shown in Figure 3.10, this mean value is subtracted from the summation block output to center the Gaussian distribution at 0 V.

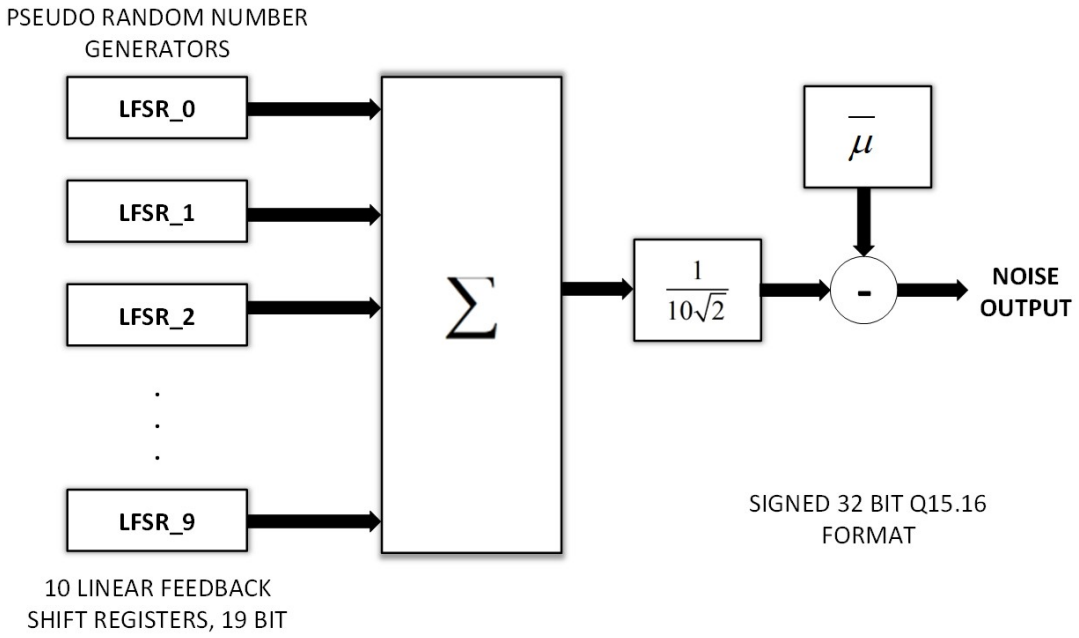


Figure 3.10. Pseudo-Gaussian Noise Generator Performed in Verilog Using 10 19-bit Pseudo-Random Number Generators

Figure 3.11 depicts a histogram of the pseudo-random noise sample generated. As described in the FSM algorithm detailed in Figure 3.4 and Section 3.2.3, the simulated noise outputs are only recorded by the system in State 7, when the simulated complex-valued target return waveform, **y**, is produced.

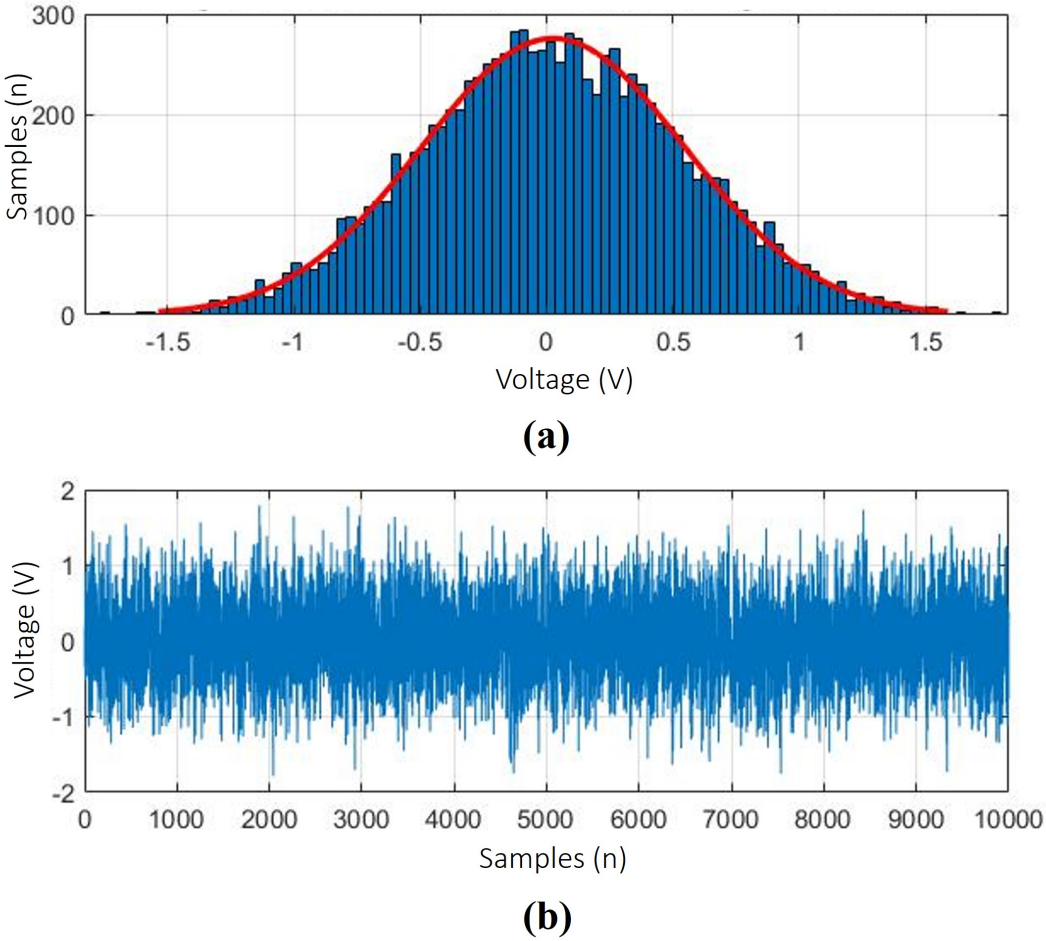


Figure 3.11. (a) Histogram of Gaussian Noise Model Using Verilog Noise Generator (b) Gaussian Noise Generation Over 10,000 Samples

3.2.6 Verilog Module Description: Transmit Waveform Generator

The adaptive matched transmit waveform generator, $XPWE.v$, is described in Figure 3.12. This module receives updated probability values from the main target recognition module, $ptheta0$, $ptheta1$, $ptheta2$, $ptheta3$, and $squareroot\_Es\_input$ as signed 32-bit Q15.16 formatted data.

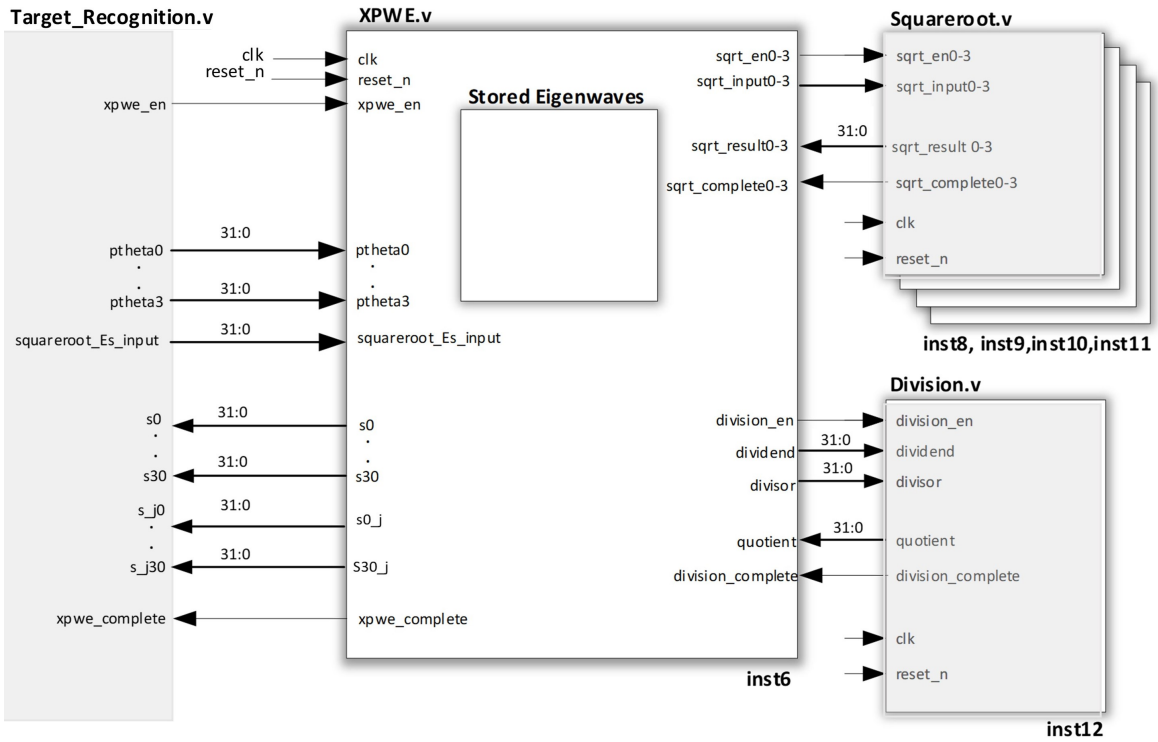


Figure 3.12. Transmit Waveform Generator Verilog Module Schematic

Using the stored eigenwaveforms, a division module, and 4 square root modules, $XPWE.v$ performs the calculations for **x** from Equations 2.5 and 2.6. $XPWE.v$ is used during the initial waveform transmission in State 2 and updated waveform transmission in State 15, as described previously in Figure 3.4. The in-phase and quadrature results are returned to $target\_recognition.v$ as signed 32-bit values represented by signal outputs $s0 - s30$ and $s\_j0 - s\_j30$, respectively. This module processes the calculation for **x** into separate steps through its own 10-state FSM, as detailed in Figure 3.13. Note that the stored target eigenwaveform parameters will be discussed later in Section 3.2.8.

State	Description
0	Idle State: Scan for enable signal from Target Recognition Module *xpwe_en* = SET
1	Initialization State: Read Inputs from Target Recognition Module
2	Calculate *sqrt(ptheta*[0:3]) **Squareroot.v - inst8, inst9, inst10, inst11**
3	Calculate Energy: *E_PWE = x_pwe*x_pwe*
4	Determine *sqrt(E_PWE)*: Normalization Scaling **Squareroot.v - inst8**
5	Read *sqrt(E_PWE)* result from squrareroot module
6	Determine 1/*sqrt(E_PWE)* using Division Module **Division.v – inst12**
7	Read 1/*sqrt(E_PWE)* result from Division Module
8	Generate Adaptive Transmit Waveform
9	Exit State: Send completion signal to Target Recognition module *xpwe_complete* = SET

Figure 3.13. Transmit Waveform Generator Finite State Machine Behavioral Process of $XPWE.v$

3.2.7 Verilog Module Description: Finite Impulse Response Filter

All convolution calculations, which are described in Equations 2.2 and 2.7 of Chapter 2 and are performed during States 5 and 8 of the FSM depicted in Figure 3.4, are processed by the finite impulse response filter module, $FIR\_Filter.v$, shown in Figure 3.14. The FIR filter conducts multiply-and-accumulate functions corresponding to the target response selection, t[0:3], as indicated by the sel signal and signed 32-bit complex-valued inputs received from the target recognition module, x[30:0] and x_j[30:0]. The convolved results are returned to the target recognition module as y[60:0] and y_j[60:0]. This module is used to calculate the values $\mathbf{s}_g$ of State 5 and $\mathbf{S}$ of State 8, as described previously in Section 3.2.3. Details regarding the four stored target responses, h[0:3], are further discussed in Section 3.2.8.

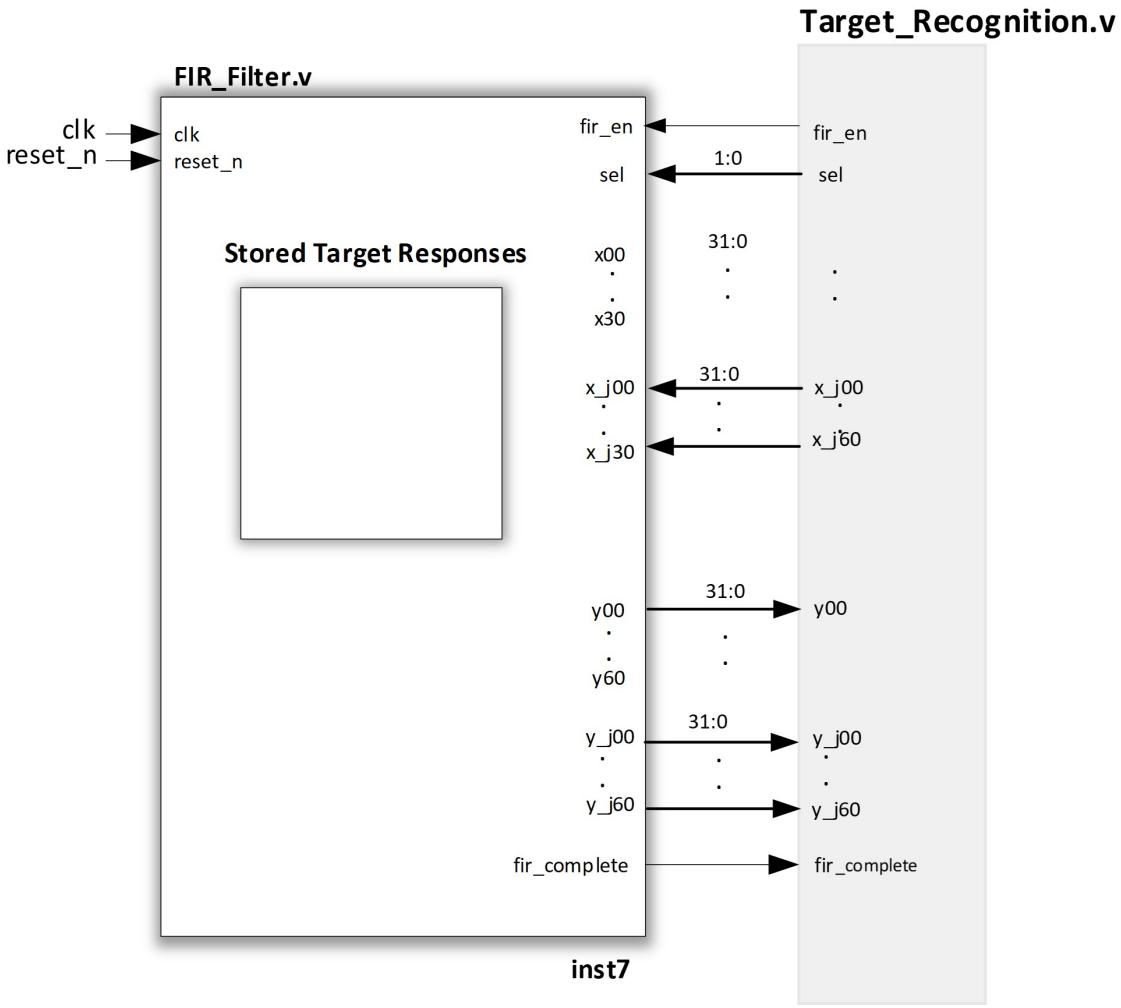


Figure 3.14. Finite Impulse Response Filter Verilog Module Schematic, FIR.v

Figure 3.15 depicts a high-level representation of the FIR Filter behavioral design in Verilog. The process initializes upon receiving the enable signal from the target recognition module. An internal LUT is implemented to access one of the four stored target responses indicated by the 2-bit *sel* input. The 32-bit filter input data are then read and stored in a 31-element register array that is used to update a 31-element shifting system. This shifting system is designed to operate similar to a shift register on a larger scale, as shown in the diagram as the "FIR Filter Input Buffer."

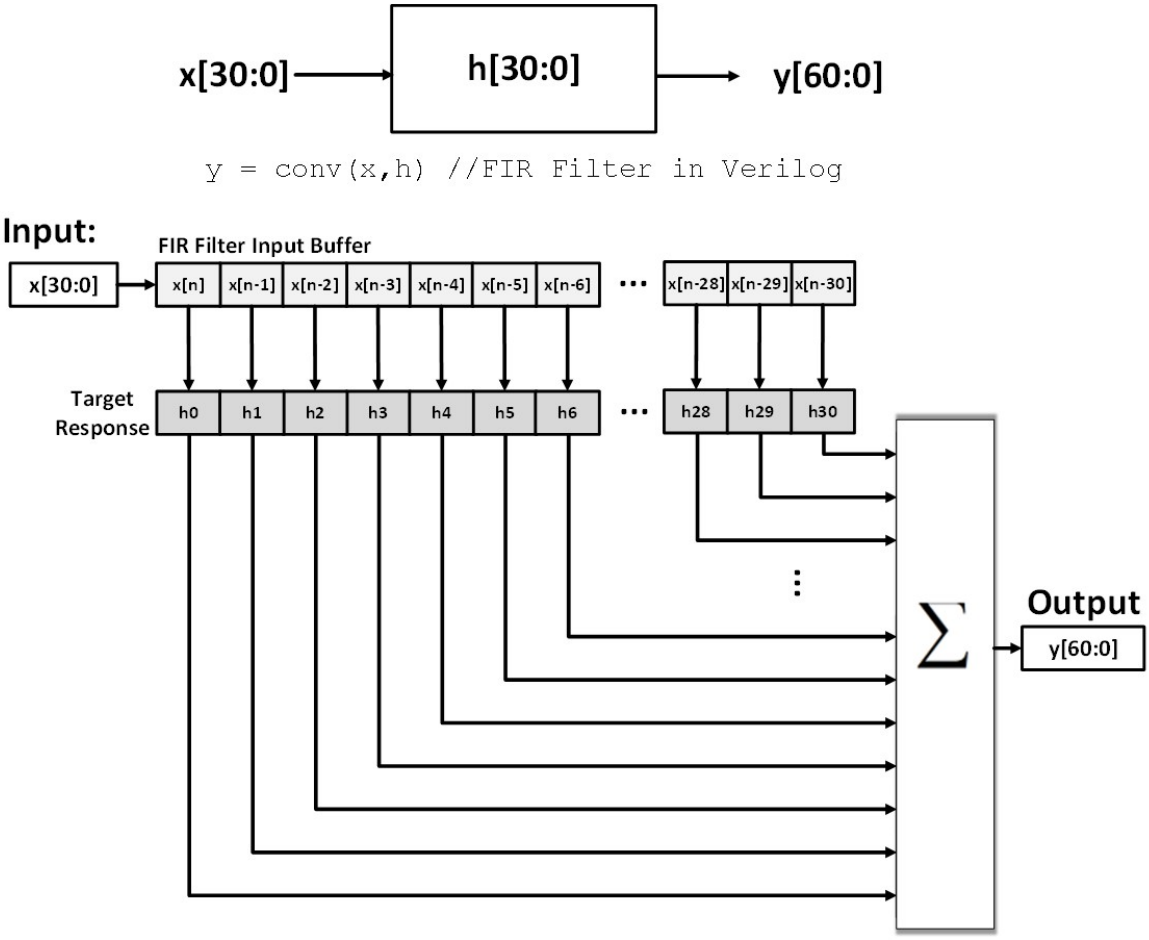
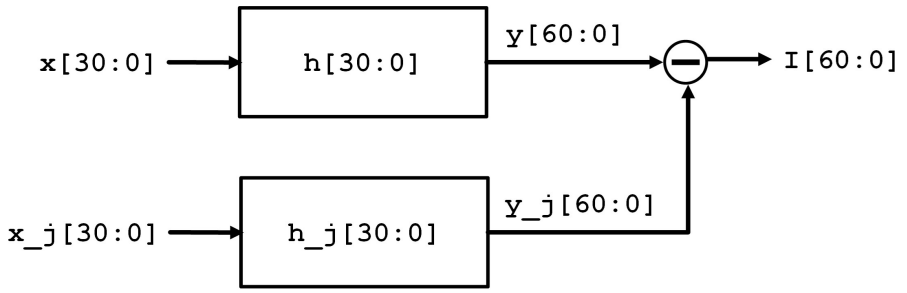


Figure 3.15. Finite Impulse Response Filter Block Diagram in Verilog

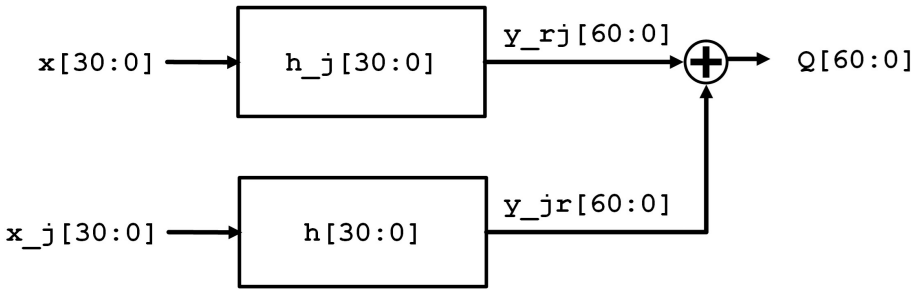
Each element of the shifting input buffer is cleared upon initialization. A counter, fir_count, is utilized to manage the FIR Filter Input Buffer load and right-shift function. A single input value is retrieved with every clock cycle for the length of the array, starting from $x[0]$ to $x[30]$. When the counter is greater than 30, zeros are introduced and shifted to the right with each cycle until the count reaches 60. The resulting convolution is stored and forwarded via y[60:0] and y_j[60:0].

$$I = \left(x_{real} * h_{real}\right) - \left(x_{imaginary} * h_{imaginary}\right)$$



(a)

$$Q = \left(x_{real} * h_{imaginary}\right) + \left(x_{imaginary} * h_{real}\right)$$



(b)

Figure 3.16. (a) In-Phase and (b) Quadrature FIR Filter Output Performed in Verilog

This design-specific module consists of four FIR filter blocks that are divided into two pairs dedicated to the in-phase filter and quadrature IO, as depicted in Figure 3.16. All four blocks execute the convolution process described in Figure 3.15 concurrently to optimize data parallelism, although additional hardware and routing are required. The overall performance of this module is observed to meet project design standards, yielding acceptable error levels when compared to MATLAB modeling, which is further discussed in Section 3.3.

3.2.8 Target Response and Eigenwaveform Parameters

In this design, four complex-valued target responses (**h0**,**h1**,**h2**,**h3**), each of which has 31 samples, are defined in Verilog and stored in the FIR filter module. Figure 3.17 depicts the in-phase/quadrature (IQ) and magnitude of each target response, derived from the data points listed in Table 1. Both the in-phase and quadrature components in each target response are accessed separately via a LUT configured in an 8 x 31 matrix.

Target responses, **h1**-**h3**, were arbitrary responses originating from a the MATLAB PWE Monte Carlo benchmark [16]. The target response for **h0**, however, was derived from measurements collected from an actual 1.090 GHz bandpass, narrowband filter with a 14 MHz bandwidth. This bandpass filter, shown in Figure 3.20, is used to emulate a target response in the hardware application design setup discussed later in Chapter 4.

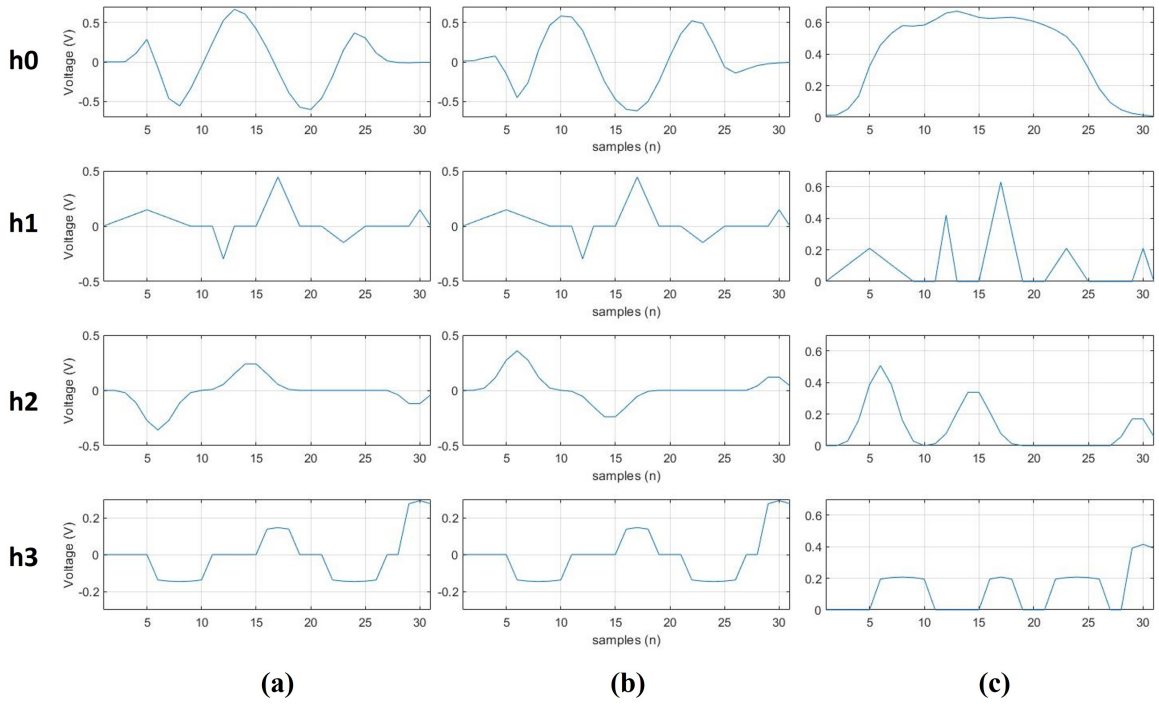


Figure 3.17. (a) In-Phase, (b) Quadrature, and (c) Magnitude Target Responses Stored in System

Table 3.1. Stored Target Responses

	h0	h1	h2	h3
0	0.0020 + j0.0115	0.0000 + j0.0000	0.0000 + j0.0000	0.0000 + j0.0000
1	0.0006 + j0.0142	0.0371 + j0.0371	0.0000 + j0.0000	0.0000 + j0.0000
2	0.0027 + j0.0502	0.0741 + j0.0741	-0.0203 + j0.0203	0.0000 + j0.0000
3	0.1098 + j0.0744	0.1112 + j0.1112	-0.1135 + j0.1135	0.0000 + j0.0000
4	0.2859 - j0.1496	0.1482 + j0.1482	-0.2725 + j0.2725	0.0000 + j0.0000
5	-0.0721 - j0.4514	0.1112 + j0.1112	-0.3585 + j0.3585	-0.1379 - j0.1379
6	-0.4617 - j0.2633	0.0741 + j0.0741	-0.2725 + j0.2725	-0.1444 - j0.1444
7	-0.5567 + j0.1626	0.0371 + j0.0371	-0.1135 + j0.1135	-0.1466 - j0.1466
8	-0.3389 + j0.4667	0.0000 + j0.0000	-0.0203 + j0.0203	-0.1444 - j0.1444
9	-0.0558 + j0.5810	0.0000 + j0.0000	0.0000 + j0.0000	-0.1379 - j0.1379
10	0.2417 + j0.5690	0.0000 + j0.0000	0.0087 - j0.0087	0.0000 + j0.0000
11	0.5255 + j0.3984	-0.2965 - j0.2965	0.0539 - j0.0539	0.0000 + j0.0000
12	0.6672 + j0.0769	0.0000 + j0.0000	0.1492 - j0.1492	0.0000 + j0.0000
13	0.6049 - j0.2468	0.0000 + j0.0000	0.2390 - j0.2390	0.0000 + j0.0000
14	0.4191 - j0.4728	0.0000 + j0.0000	0.2390 - j0.2390	0.0000 + j0.0000
15	0.1749 - j0.6004	0.2224 + j0.2224	0.1492 - j0.1492	0.1379 + j0.1379
16	-0.1114 - j0.6198	0.4447 + j0.4447	0.0539 - j0.0539	0.1466 + j0.1466
17	-0.3892 - j0.4989	0.2224 + j0.2224	0.0087 - j0.0087	0.1379 + j0.1379
18	-0.5729 - j0.2440	0.0000 + j0.0000	0.0000 + j0.0000	0.0000 + j0.0000
19	-0.6031 + j0.0715	0.0000 + j0.0000	0.0000 + j0.0000	0.0000 + j0.0000
20	-0.4607 + j0.3572	0.0000 + j0.0000	0.0000 + j0.0000	0.0000 + j0.0000
21	-0.1833 + j0.5212	-0.0741 - j0.0741	0.0000 + j0.0000	-0.1379 - j0.1379
22	0.1508 + j0.4876	-0.1482 - j0.1482	0.0000 + j0.0000	-0.1444 - j0.1444
23	0.3671 + j0.2286	-0.0741 - j0.0741	0.0000 + j0.0000	-0.1466 - j0.1466
24	0.3037 - j0.0641	0.0000 + j0.0000	0.0000 + j0.0000	-0.1444 - j0.1444
25	0.1125 - j0.1411	0.0000 + j0.0000	0.0000 + j0.0000	-0.1379 - j0.1379
26	0.0146 - j0.0922	0.0000 + j0.0000	0.0000 + j0.0000	0.0000 + j0.0000
27	-0.0085 - j0.0467	0.0000 + j0.0000	-0.0399 + j0.0399	0.0000 + j0.0000
28	-0.0099 - j0.0224	0.0000 + j0.0000	-0.1195 + j0.1195	0.2757 + j0.2757
29	-0.0069 - j0.0111	0.1482 + j0.1482	-0.1195 + j0.1195	0.2932 + j0.2932
30	-0.0043 - j0.0056	0.0000 + j0.0000	-0.0399 + j0.0399	0.2757 + j0.2757

To obtain the bandpass filter response for **h0**, a vector network generator (VNA) was utilized and swept from 1.07 GHz to 1.13 GHz, as shown in the equipment screen capture of Figure 3.18. The S21 magnitude and phase measurements captured in the frequency domain ($Tr4$ and $Tr7$) were used to obtain the baseband time domain representation, shown in Figure 3.19. This signal was then down-sampled to 31 samples, listed previously in Table 3.1.

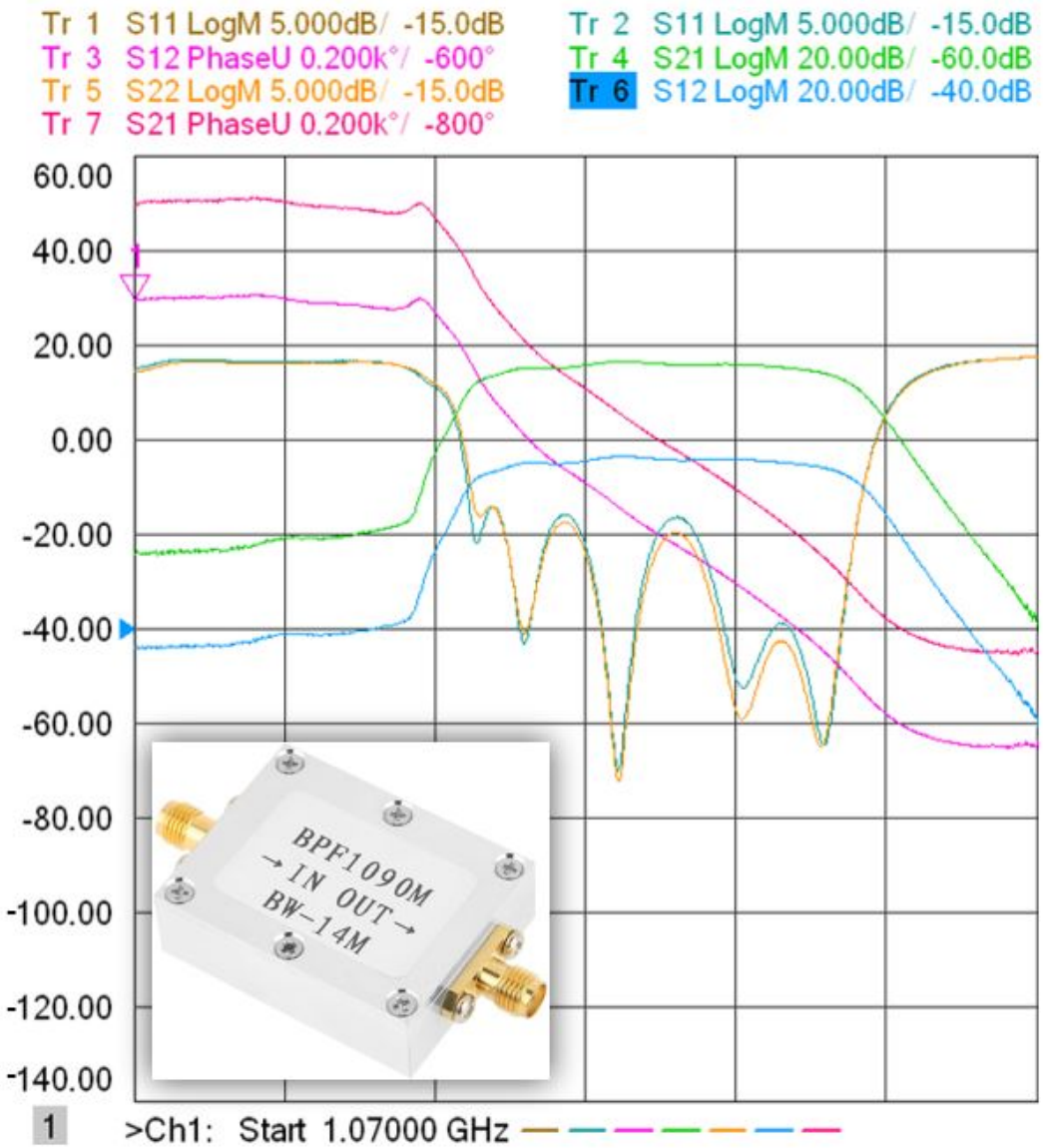
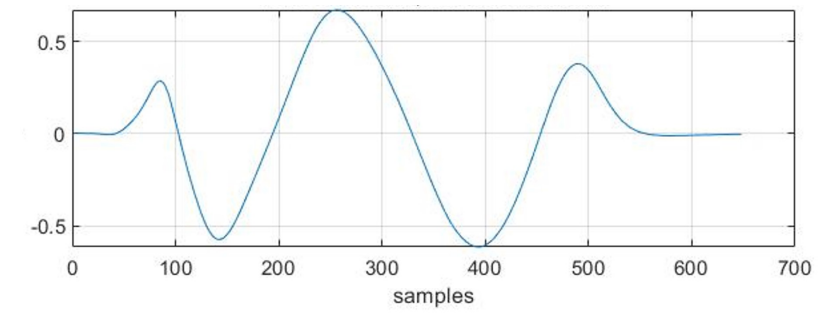
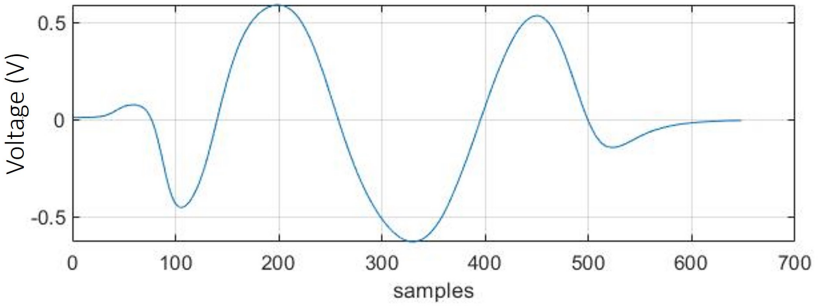


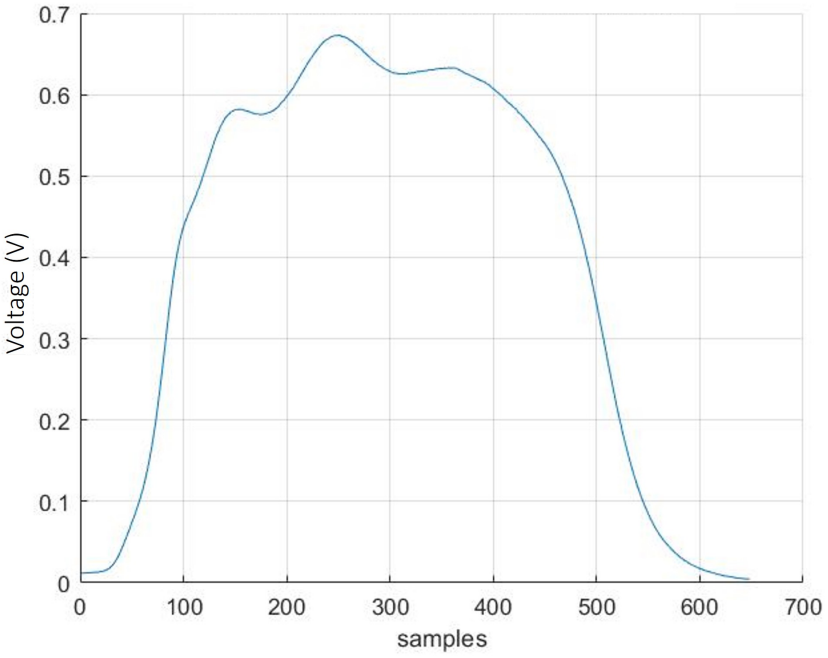
Figure 3.18. Vector Network Analyzer Measurements: Magnitude and Phase of a 1.090 GHz Narrowband Bandpass Filter in the Frequency Domain



(a)



(b)



(c)

Figure 3.19. Baseband: (a) In-Phase, (b) Quadrature, and (c) Magnitude Derived from 1.090 GHz Bandpass Filter Measurements from VNA

Figure 3.20 and Table 3.2 depict the complex-valued eigenwaveforms accessed via a LUT within the adaptive waveform generator module. This LUT is constructed similarly to the target response vectors defined previously. Using Equations 2.3 and 2.4 introduced in Chapter 2, eigenwaveforms, $\bar{\mathbf{q}}_{i,max}$, represented in Verilog as registers **se0**, **se1**, **se2**, **se3**, were calculated offline in MATLAB corresponding to a priori knowledge of the 4 target responses, (**h0**, **h1**, **h2**, **h3**). Since the eigenwaveforms may be pre-calculated (assuming known targets), generating a matrix convolution algorithm in the digital design is not required, thus saving memory and hardware logic resources. Any future additions of target possibilities, of course, require calculating the eigenwaveforms offline.

Note in Figure 3.20 that the quadrature component of eigenwaveforms **se1**-**se3** are null. Unlike the actual response from the filter, **h1**-**h3** have similar in-phase and quadrature responses thereby yielding zero-quadrature responses. Such coincidence does not affect implementation of CRr PWE technique in our work herein.

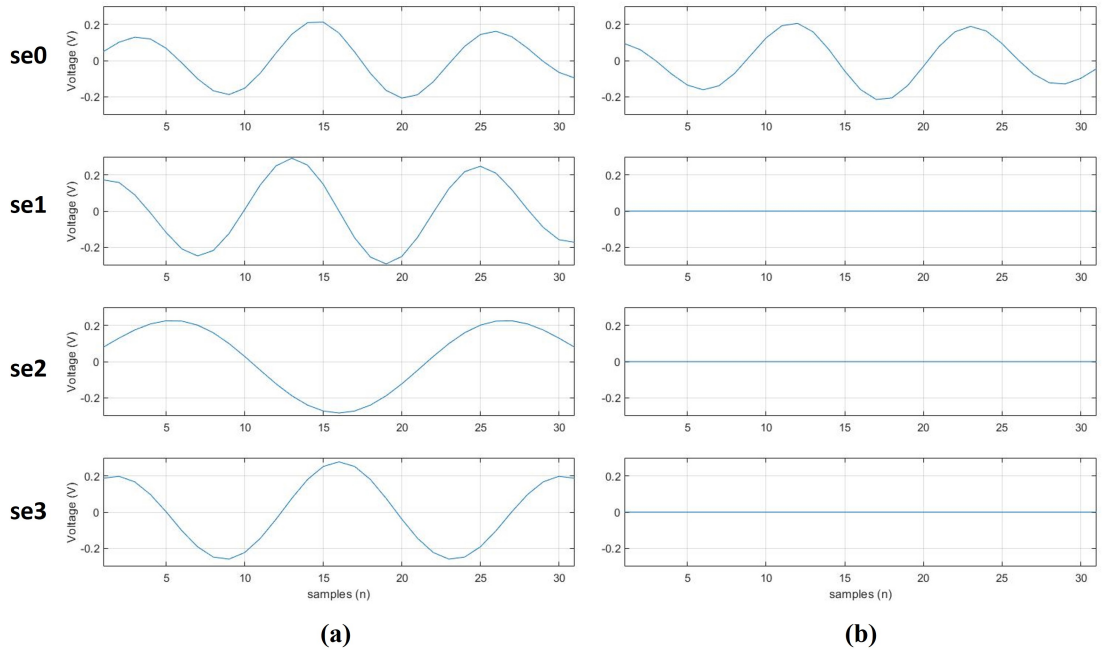


Figure 3.20. Baseband: (a) In-Phase, and (b) Quadrature Target Eigenwaveforms Stored in the System

Table 3.2. Stored Eigenwaveforms

	se0	se1	se2	se3
0	0.0495 + j0.0939	0.1723	0.0795	0.1872
1	0.1011 + j0.0603	0.1579	0.1297	0.1978
2	0.1292 - j0.0015	0.0899	0.1746	0.1673
3	0.1191 - j0.0745	-0.01	0.2081	0.0971
4	0.0679 - j0.1353	-0.1184	0.2255	0.0016
5	-0.0130 - j0.1614	-0.2089	0.2238	-0.1024
6	-0.1005 - j0.1393	-0.2476	0.2015	-0.1915
7	-0.1667 - j0.0708	-0.2177	0.1592	-0.248
8	-0.1877 + j0.0268	-0.1242	0.0998	-0.2595
9	-0.1528 + j0.1249	0.0085	0.0284	-0.223
10	-0.0685 + j0.1923	0.1455	-0.0482	-0.1444
11	0.0420 + j0.2057	0.2504	-0.1229	-0.0387
12	0.1456 + j0.1576	0.2917	-0.1888	0.0761
13	0.2095 + j0.0603	0.2538	-0.2402	0.1802
14	0.2124 - j0.0576	0.1482	-0.2727	0.2521
15	0.1522 - j0.1598	0	-0.2839	0.2776
16	0.0472 - j0.2149	-0.1482	-0.2727	0.2521
17	-0.0704 - j0.2063	-0.2538	-0.2402	0.1802
18	-0.1646 - j0.1378	-0.2917	-0.1888	0.0761
19	-0.2075 - j0.0319	-0.2504	-0.1229	-0.0387
20	-0.1887 + j0.0778	-0.1455	-0.0482	-0.1444
21	-0.1172 + j0.1587	-0.0085	0.0284	-0.223
22	-0.0176 + j0.1888	0.1242	0.0998	-0.2595
23	0.0789 + j0.1630	0.2177	0.1592	-0.248
24	0.1441 + j0.0936	0.2476	0.2015	-0.1915
25	0.1618 + j0.0051	0.2089	0.2238	-0.1024
26	0.1318 - j0.0744	0.1184	0.2255	0.0016
27	0.0686 - j0.1226	0.01	0.2081	0.0971
28	-0.0048 - j0.1291	-0.0899	0.1746	0.1673
29	-0.0651 - j0.0981	-0.1579	0.1297	0.1978
30	-0.0962 - j0.0448	-0.1723	0.0795	0.1872

3.3 Verilog Model Results

This section presents the resulting data output generated from the stand-alone Monte Carlo simulation model. All data points are collected from virtual test points defined in the Verilog testbench module and printed onto the TCL console in Vivado. In this section, the data points are compared to the MATLAB benchmark [16]. The functionality of the PWE algorithm in this design demonstrates the feasibility of implementing this target recognition technique into digital logic hardware.

3.3.1 Vivado Output

Figure 3.21 lists the Vivado simulation printout from the TCL console from the Monte Carlo trials. Per the FSM algorithm presented in Section 3.2.1, the system transmits 4 adaptive waveform iterations based on a randomly selected target response ($target\_sel$) at the beginning of each Monte Carlo trial. After processing the last target return, the system arrives at a target classification decision by selecting the target hypothesis associated with the highest probability.

```
------------------------------------------------------------------------
Output Results:
------------------------------------------------------------------------
ks: 0,  pc[ks]: 0.325912, err: 674
ks: 1,  pc[ks]: 0.352921, err: 647
ks: 2,  pc[ks]: 0.403931, err: 596
ks: 3,  pc[ks]: 0.465881, err: 534
ks: 4,  pc[ks]: 0.548859, err: 451
ks: 5,  pc[ks]: 0.626877, err: 373
ks: 6,  pc[ks]: 0.70993,  err: 290
ks: 7,  pc[ks]: 0.798813, err: 201
ks: 8,  pc[ks]: 0.886826, err: 113
ks: 9   pc[ks]: 0.955841, err:  44
ks: 10, pc[ks]: 0.986816, err:  13
ks: 11, pc[ks]: 0.998795, err:   1
ks: 12, pc[ks]: 1,        err:   0
ks: 13, pc[ks]: 1,        err:   0
ks: 14, pc[ks]: 1,        err:   0
------------------------------------------------------------------------

run: Time (s): cpu = 00:04:22 ; elapsed = 00:22:50 .
Memory (MB): peak = 1720.129 ; gain = 0.000
xsim: Time (s): cpu = 00:04:24 ; elapsed = 00:22:54 .
Memory (MB): peak = 1720.129 ; gain = 0.000
INFO: [USF-XSim-96] XSim completed. Design snapshot 'Target_Recognition_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 610000000ns
launch_simulation: Time (s): cpu = 00:04:25 ;
elapsed = 00:23:08 . Memory (MB): peak = 1720.129 ; gain = 0.000
```

Figure 3.21. Monte Carlo Output from Vivado Simulation

As shown in the Vivado printout, the system performs 1000 classification experiments at each transmit energy starting from -30 dB units (iteration counter $ks = 0$) up to 10 dB units (iteration $ks = 14$). For the purposes of quantifying classification performance, the error counter is incremented if the decision happens to be incorrect, as shown in Figure 3.21 as err. After 1000 trials at each transmit energy level, the probability of correct classification ($pc[ks] = 1 - (err/1000)$) is determined and printed in Vivado. As the transmitted energy and SNR increase, the probability of correct classification increases. The P_{cc} at transmit energy levels greater -5 dB units ($ks = 9$) ranges from 0.955841 to 1, as demonstrated in Figure 3.21.

Figure 3.22 depicts a graphical representation of a Vivado capture further describing the adaptive waveform transmission, **x**, for E_x = -10 dB units and $target\_sel = target0$. Notice that the summation of $ptheta0$ through $ptheta3$ values must equal 1. When the highest probability value corresponds to the randomly selected target, $target\_sel$, the system correctly classifies the target. Otherwise, an error is assigned.

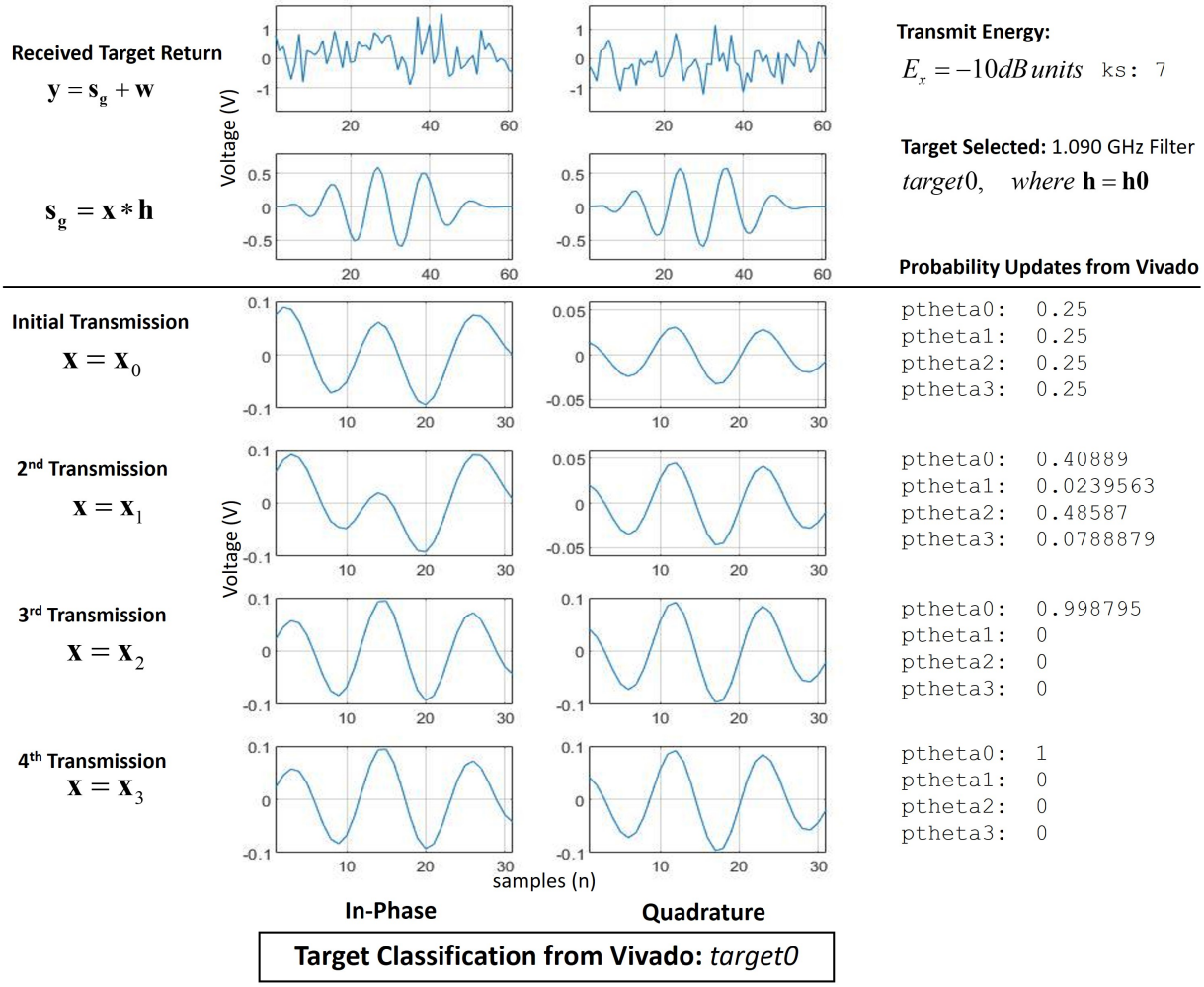


Figure 3.22. Adaptive Waveform Transmission from Vivado Simulation, where E_x = -10 dB units, the Randomly Selected Target is $target0$, and the Correct Classification is **h0**

3.3.2 Results Analysis and Observations

In Figure 3.23, the complex-valued adaptive transmit waveform from the Verilog model is plotted with the transmit waveform from the MATLAB benchmark [16], as denoted in red and blue respectively. This adaptive waveform capture represents the initial transmitted signal at the lower end of the simulation where the transmit energy E_x = -30 dB units to highlight compounded effects of data precision and limitations using 32-bit signed fixed-point values (in addition to approximation errors from the division and square root modules). Note with 32-bit multiplication, some precision is lost since we shift 16 bits to the right, as discussed in Section 3.2.

By inspection, the Verilog waveform amplitude samples are closely aligned with the MATLAB waveform samples, but yield an average percent deviation of 6.32%.

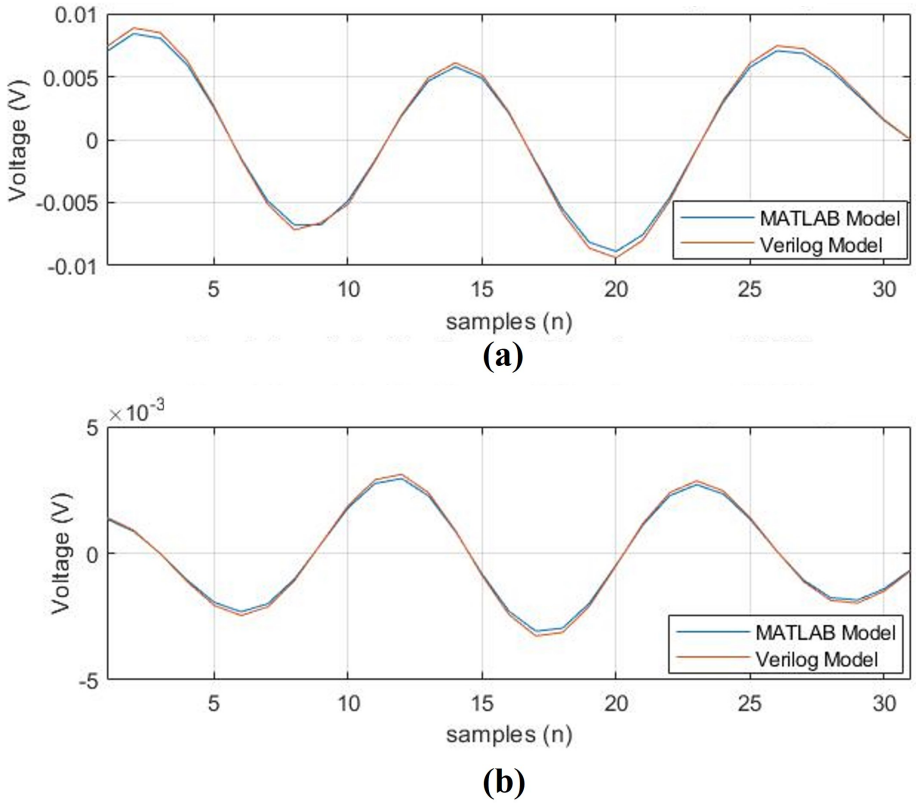


Figure 3.23. (a) In-Phase and (b) Quadrature Adaptive Transmit Waveform, x, E_x = -30 dB units: Verilog Model vs. MATLAB Model

Figure 3.24 shows the in-phase and quadrature output captured from the FIR filter module. This is the convolution of the adaptive transmit waveform, **x**, and the baseband filter response, **h0**. Similar to the adaptive waveform diagram of Figure 3.23, the Verilog FIR filter output is closely aligned with the MATLAB output [16]. On average, the percent deviation is similar to the transmit waveform Verilog output.

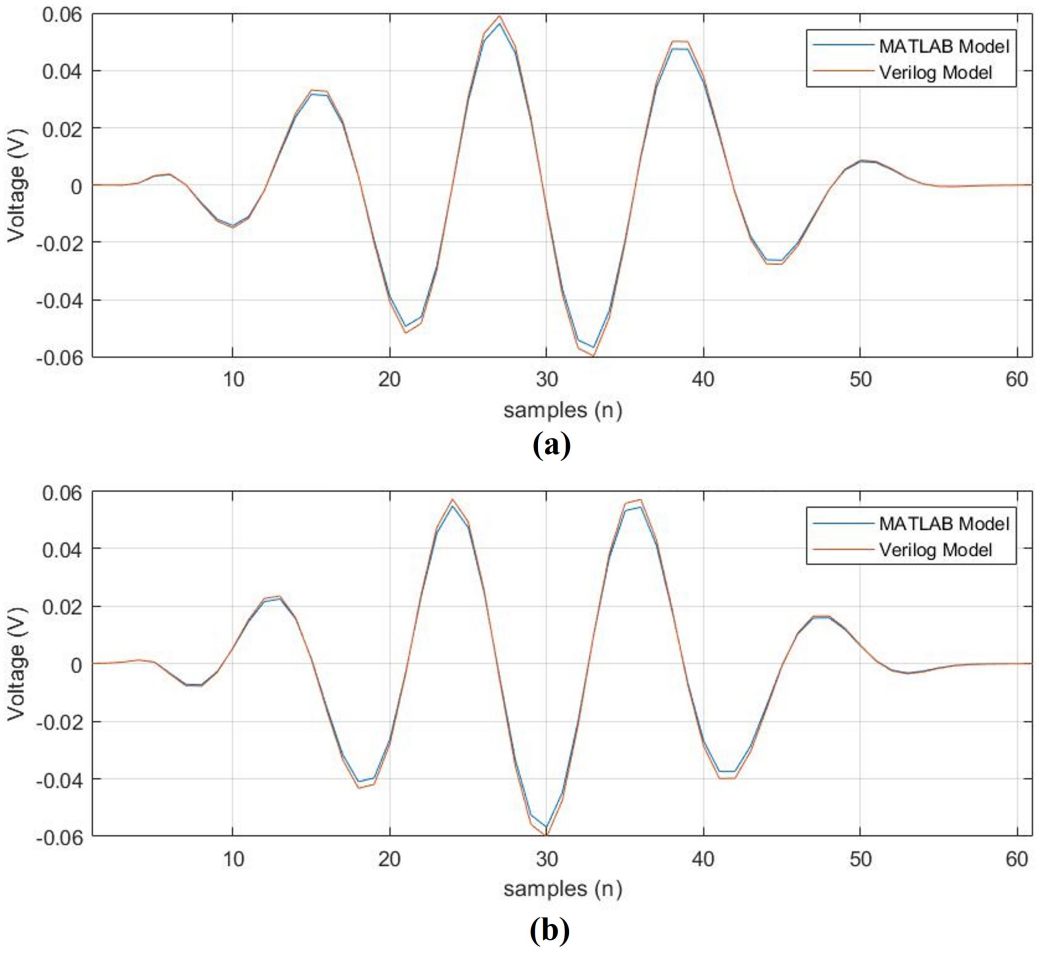


Figure 3.24. (a) In-Phase and (b) Quadrature FIR Filter Output, $\mathbf{S} = \mathbf{x} * \mathbf{h0}$, E_x = -30 dB units: Verilog Model vs. MATLAB Model

To quantify the performance of the Verilog implementation compared to MATLAB implementation, the Verilog P_{cc} is compared with MATLAB P_{cc}. As expected, there is a performance degradation with hardware implementation which is greater for lower transmit energy compared to higher transmit energy. A percent deviation averaging 10% is present in the lower transmit energy ranges starting from -30 dB units to -10 dB units.

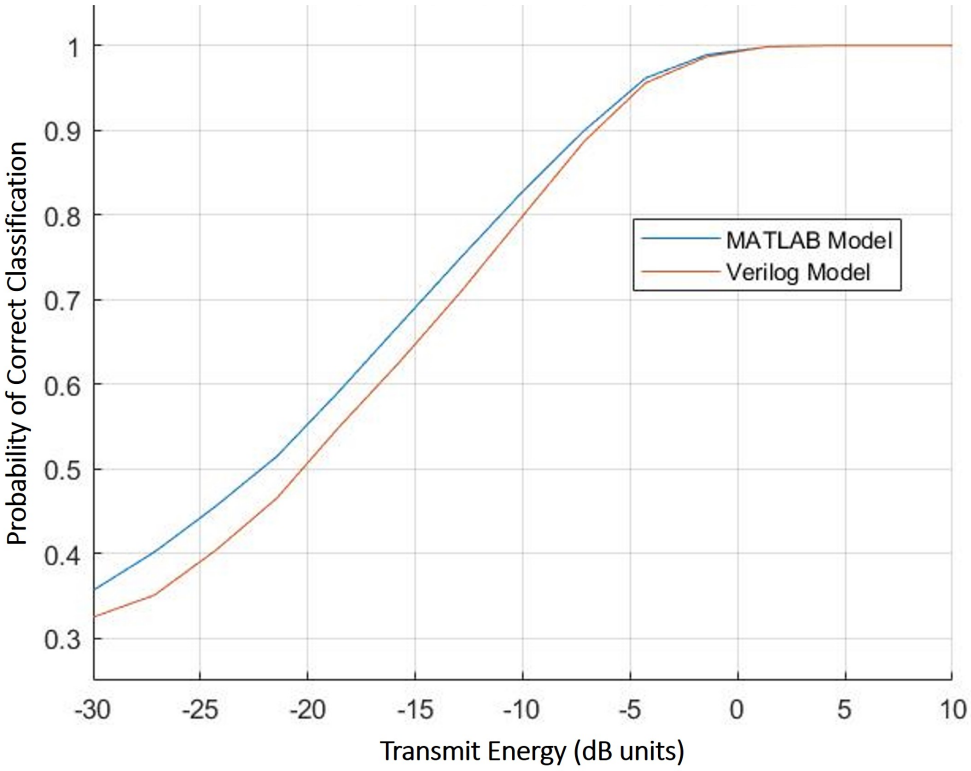


Figure 3.25. Monte Carlo Simulation Results: Verilog Implementation vs. MATLAB Implementation

The performance degradation is due to issues discussed earlier including, data precision limitations, bit rounding errors, custom math functions performed in Verilog (divide, square root, exponential), and/or errors related to multiplication. Additionally, the Gaussian noise distribution of the MATLAB simulation which is more truly random (i.e. uncorrelated samples) in comparison to the pseudo-random LFSR is a contributing factor. Understanding that the feasibility of implementing the PWE algorithm into a digital logic system in Verilog is more paramount than error minimization, the performance degradation is expected.

3.4 Chapter Summary

In this chapter we defined the 32-bit Q15.16 signed fixed-point data structure within the Verilog Monte Carlo Design architectures. We then illustrated the Top Level design description, examined the FSM algorithm, and described each custom Verilog design component within the system. Additionally, stored target response and eigenwaveform parameters were presented, as well as a description of the Vector Network Analyzer procedure used to determine a target filter response, from an actual hardware filter. Finally, we examined the Verilog output and compared it to the original MATLAB PWE benchmark. We concluded that digital hardware implementation of the PWE technique in Verilog is feasible, while at the same time verifying that the Verilog components are functional.

In Chapter 4, we move onto hardware applications and incorporate the proven Verilog modules developed in this chapter into an FPGA-based CRr target recognition system design. We introduce the Xilinx VCU118 Evaluation Board and its integration as a device-in-the-loop processor with Rohde & Schwarz RF equipment.

CHAPTER 4:
Hardware Design and RF Implementation

4.1 Hardware Implementation Overview

In this chapter, we demonstrate the feasibility of implementing the PWE CRr target recognition technique into a functional, real-time hardware application. Specifically, we design a system that incorporates an FPGA-based controller that is integrated with Rohde & Schwarz (R&S) RF equipment as radar transceiver elements. In other words, we actually have to up-convert the adaptive waveform to an RF frequency, transmit to a target (the bandpass filter), receive with the spectrum analyzer, and use the FPGA to perform receive processing and the generation of the next transmit waveform for CRr closed-loop feedback. The Verilog modules that we developed for the Monte Carlo simulation model in Chapter 3 serve as the foundation of the FPGA controller and processing algorithms.

The first part of this chapter begins with an overview of the system hardware components, equipment configuration, and IO interconnections. A system-level functional design description is then presented. We will describe the mixed-language architecture that is nested within the proprietary R&S Very High-Speed Integrated Circuit Hardware Description Language (VHDL) wrapper and IP core [23]. Finally, we close this chapter with a proof-of-concept system design demonstration and provide observations.

4.2 Closed-loop Radar Configuration

The CRr target recognition system design is controlled by a Virtex UltraScale+ FPGA featured on the Xilinx VCU118 Evaluation Board. The VCU118 is configured as a device in-the-loop to accomplish digital signal processing, decision-making requirements, and inter-hardware communications. As depicted in Figure 4.1, the R&S SMW200A Vector Signal Generator and FSW Signal & Spectrum Analyzer function as the CRr Closed-loop Radar Configuration transmitter and receiver.

Digital data communication exchange between the VCU118 and R&S equipment is established through dedicated Quad Small Form-Factor Pluggable+ (QSFP+) interfaces. In this experimental closed-loop radar setup, the target, $target0$, is emulated as a 1.090 GHz narrow-band bandpass filter with a 14 MHz bandwidth, as described previously in Chapter 3.

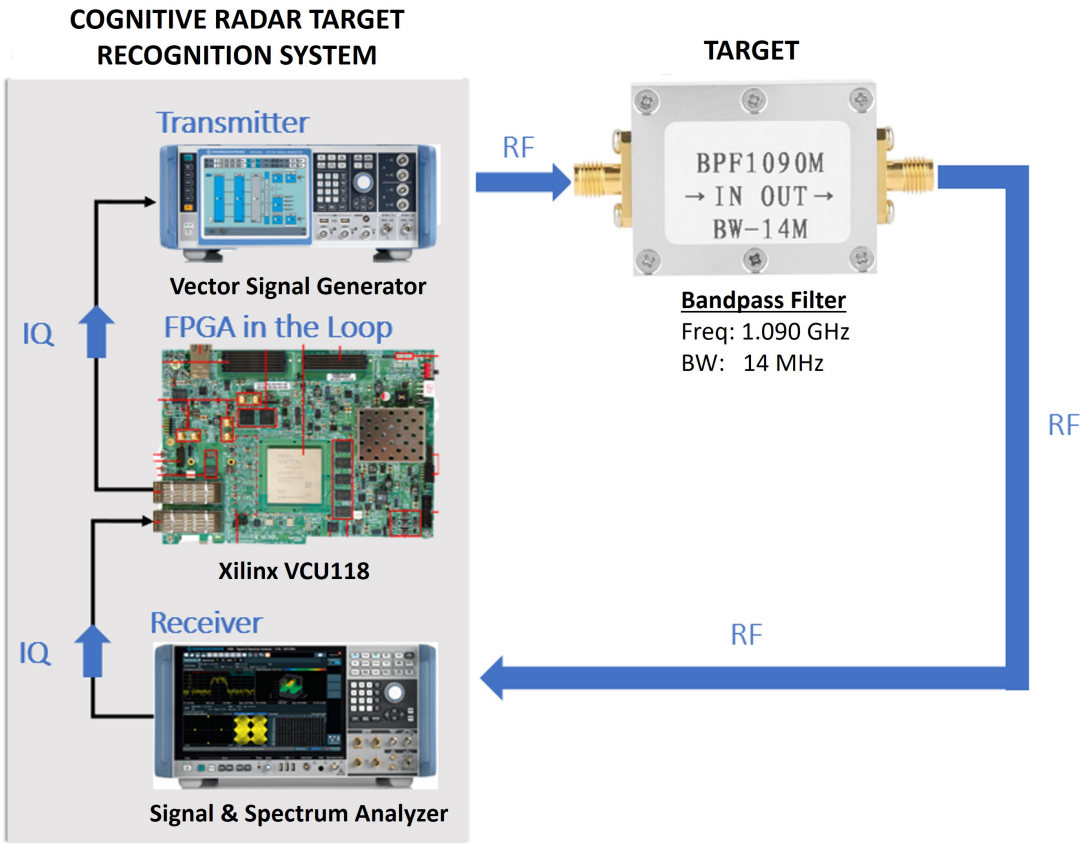


Figure 4.1. Closed-loop RF System Diagram of the Cognitive Radar Target Recognition System Experiment. Processor: VCU118, Transmitter: SMW200A, Receiver: FSW

Upon initialization, the FPGA-generated PWE adaptive transmit waveform, $\mathbf{x}$, is transmitted from the VCU118 in a proprietary R&S high-speed digital IQ (HS DIG IQ) data format over the 56 Gb/s QSFP+ active optical cable to the R&S SMW200A Vector Signal Generator. Serving as the CRr RF transmitter, the SMW200A Vector Signal Generator converts the HS DIG IQ baseband input data into RF that is sent using a set carrier frequency matching

the target center frequency (1.090 GHz). The interaction between transmit waveform and the target response of the bandpass filter results in the RF return waveform that is captured by the FSW Signal & Spectrum Analyzer. Used as the CRr receiver, the FSW converts the received RF return signal into HS DIG IQ format, which is then fed back to the VCU118 through the second QSFP+ optical interface for processing.

4.2.1 Cognitive Radar: FPGA-in-the-Loop

The Xilinx VCU118 Evaluation Board is selected for its high-capacity, high-performance characteristics that are compatible with R&S equipment and proprietary Vivado IP core. In general, this evaluation board provides the hardware systems developer with the ability to implement the Virtex UltraScale+ XCVU9P-L2FLGA2104E FPGA device into digital signal processing and logic designs [24]. As shown in Figure 4.2, multiple general purpose onboard resources integrated into the evaluation board enable hardware prototyping, testing, and concept realization. Specifically, this project utilizes the FPGA General Purpose Input/Output (GPIO) pins accessed through the Xilinx standard peripheral modular interface (PMOD) interfaces, Universal Asynchronous Receiver/Transmitter (UART), onboard general purpose switches, light emitting diodes (LEDs), and QSFP+ ports. Note that this evaluation board also features the Xilinx Zinq 7000 System-on-Chip (SoC) XC7Z101-based controller [24], however, this device, as well as other features such as onboard memory and PCI bus interconnections, are not used in this hardware design.

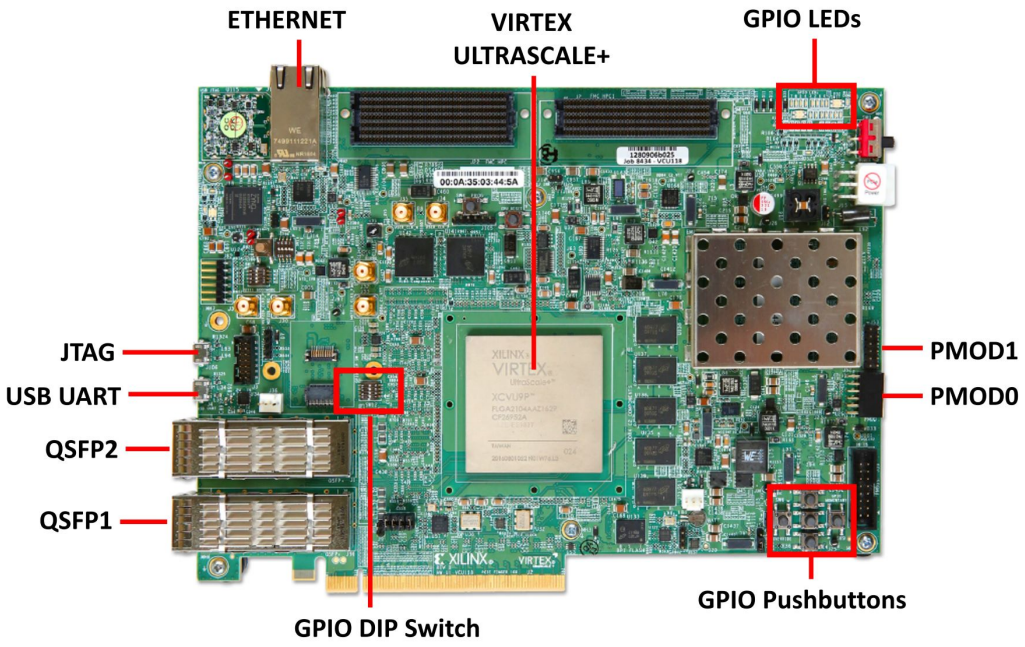


Figure 4.2. Xilinx VCU118 Evaluation Board Featuring the Virtex UltraScale+ XCVU9P-L2FLGA2104E FPGA. Adapted from [23]

The Virtex UltraScale+ XCVU9P-L2FLGA2104E FPGA featured on the VCU118 is packaged as a 47.5 mm x 47.5 mm fine pitch ball grid array (FBGA) with 832 IO pins of its total 2,104 pins [24]. This device is comprised of 2,586,150 logic elements, 2,364,480 configurable logic block (CLB) flip-flops, 1,182,240 CLB LUTs, and 6,840 DSP slices [24]. Additionally, the XCVU9P-L2FLGA2104E FPGA features 345.9 Mb of volatile memory and 120 GTY 32.75 Gb/s transceivers [24]. Further VCU118 features and IO interconnections within the CRr target recognition system experiment will be described later in Section 4.2.3.

4.2.2 Cognitive Radar: RF Transmitter and Receiver

The R&S SMW200A Vector Signal Generator and FSW Signal & Spectrum Analyzer, shown in the equipment setup of Figure 4.3, function as RF transmitter and receiver in this FPGA-in-the-loop CRr target recognition system experiment. These equipment are selected for their highly accurate RF capabilities. They are compatible with the R&S IP core that is readily available for VCU118-based radar design applications. In both R&S systems, the

HS DIG IQ interface is used for data exchange between the VCU118 and equipment via the QSFP+ optical cable connections through a proprietary R&S data bus protocol. Figure 4.4 shows the QSFP+ equipment connections to the VCU118. To establish communications, a PC-based R&S equipment initialization software delivers setup commands to the VCU118 via the Silicon Labs serial USB UART connection. Additionally, remote network operations are used in this project to view and control the equipment over LAN connections during testing.

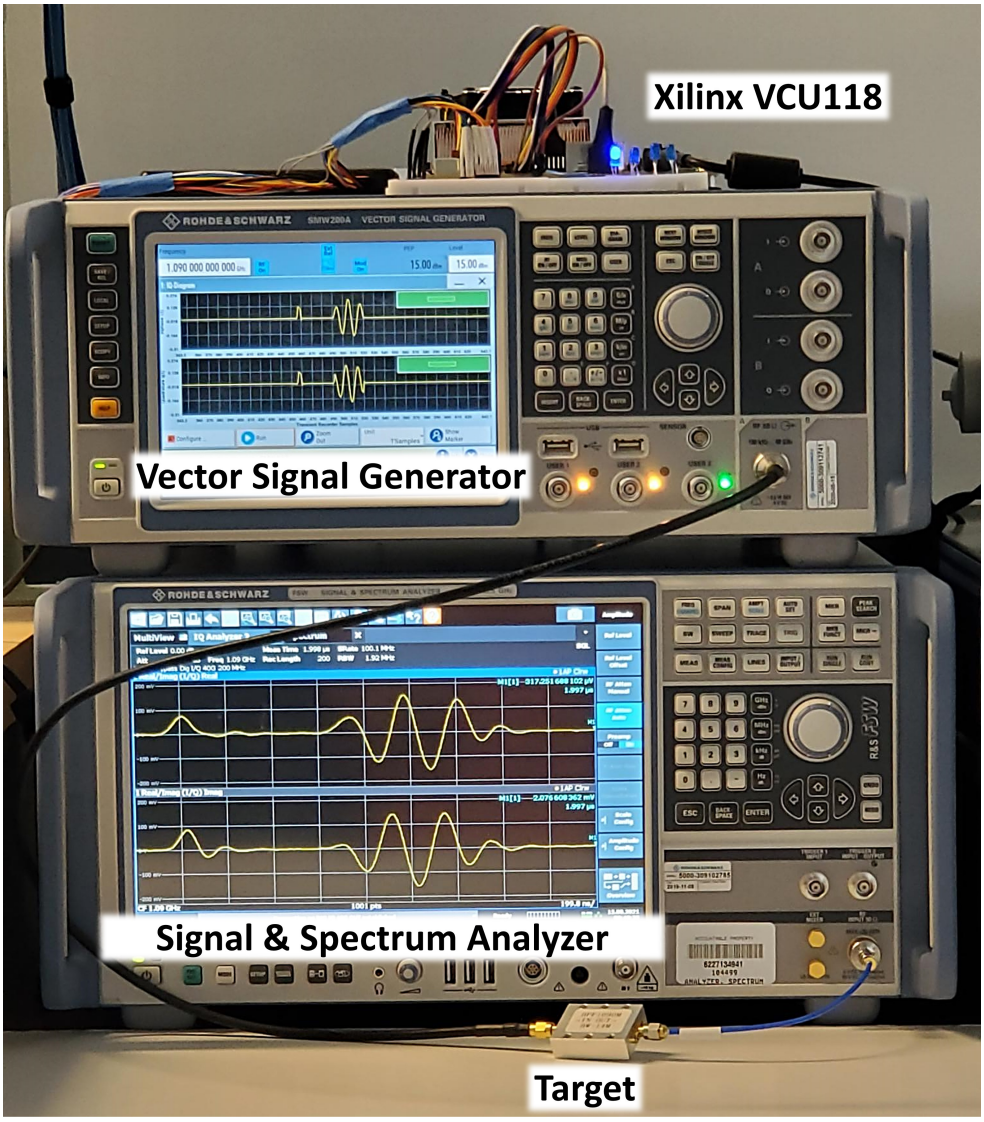


Figure 4.3. Cognitive Radar Target Recognition System Equipment Setup

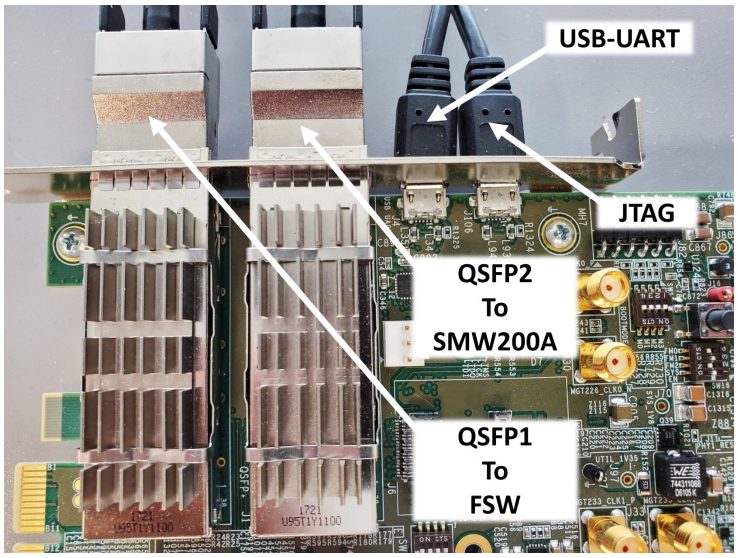


Figure 4.4. VCU118 QSFP+ Connections to Rohde & Schwarz Equipment and PC

R&S equipment synchronization is established through a shared 10 MHz reference frequency sourced from the FSW. The RF carrier frequency is set to 1.090 GHz center frequency of the bandpass filter, $target0$. On the transmitter side, Figure 4.5 shows the signal flow block diagram of the SMW200A Vector Signal Generator. Note the active input modules within the RF path highlighted in blue. The baseband input, $BB\ Input$, represents the HS DIG IQ data from the VCU118 that undergoes IQ Modulation, via $I/QMod$,. This IQ modulated baseband signal is then upconverted when the RF block is activated. A configurable additive white Gaussian noise generator, shown as the $AWGNA$ block within the signal flow path, provides the option of adding adjustable noise into the receiver that can be greatly higher than the thermal noise floor when selected [25]. The $AWGNA$ output bandwidth is set to 14 MHz and the SNR setting [25] is adjustable ranging from -30 dB to 45 dB. We set the transmit output power to 15 dBmW. On the receiver side, the FSW is configured with a low noise amplifier (preamp). When the preamp is used, the noise figure provided by the manufacturer is approximately $F = 2$ dB units. The estimated thermal noise power, $P_n = kT_0FB$, is approximately -100.5 dBmW, where k is the Boltzmann's constant and $T_0 = 29$ K, Additional system losses are present due to the bandpass filter, cable length, and FPGA.

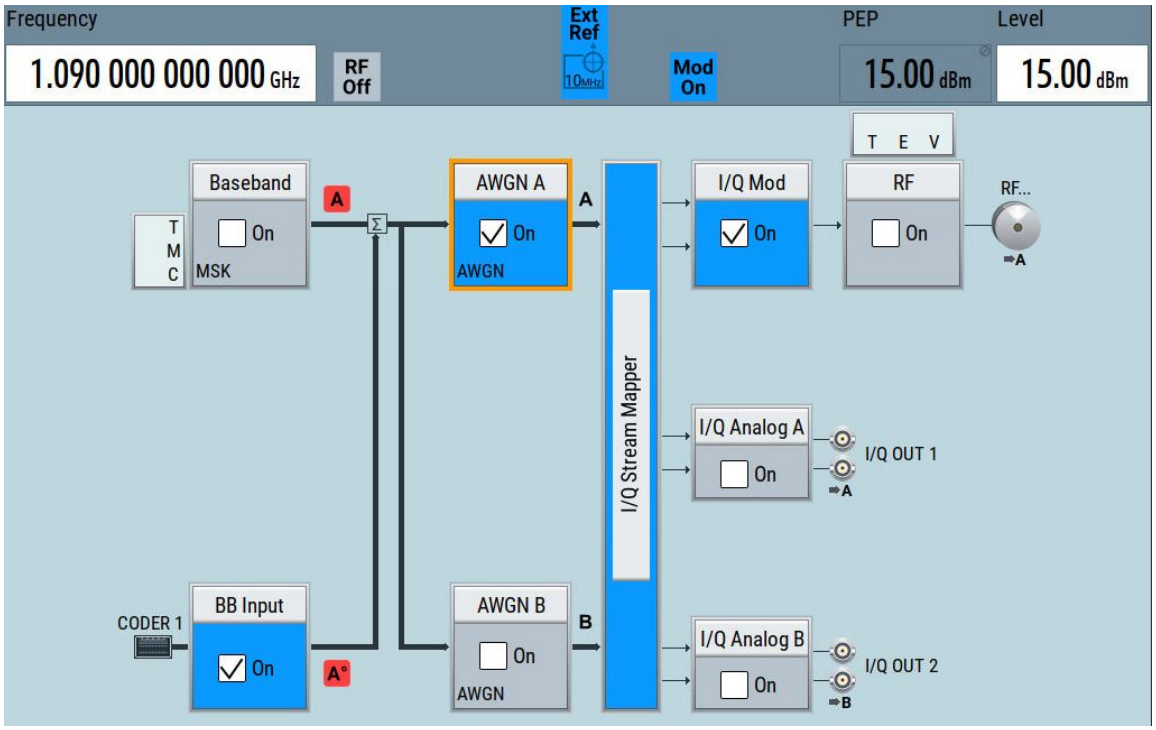


Figure 4.5. SMW200A Signal Flow Block Diagram Configuration: Baseband High Speed Digital IQ with Optional Additive White Gaussian Noise

4.2.3 System IO

The Xilinx VCU118 system IO in this design is represented in Figures 4.6 and 4.7. Internal components include the USB JTAG interface for device configuration, system clocks, and onboard reset circuitry. For simplicity, non-volatile memory featured on the VCU118 is not used in this demonstration, but is available to provide design portability.

Onboard VCU118 general purpose input ports include the $GPIO\_SW\_C$ momentary push-button to initiate the PWE transmit waveform and the 4-bit $GPIO\_DIP\_SW$ ports used for system testing. The peripheral modular interface ports, $PMOD0$ and $PMOD1$, are configured to observe data over a logic analyzer and target classification decision through a simple active-low configured LED circuit. Sections 4.2.4 through 4.2.7 will provide a schematic level overview of each IO feature described.

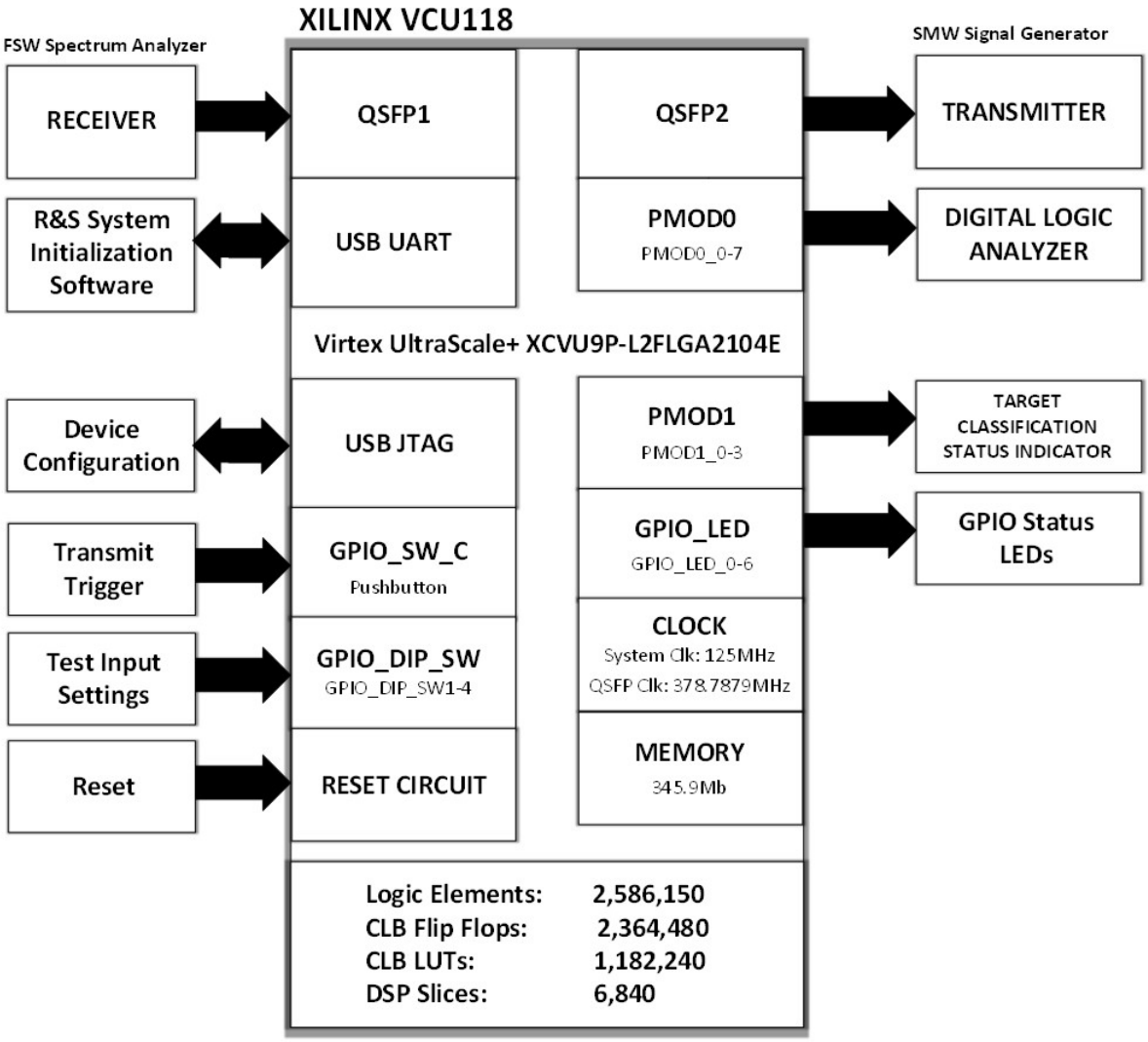


Figure 4.6. Xilinx VCU118 Block Diagram and Hardware System IO

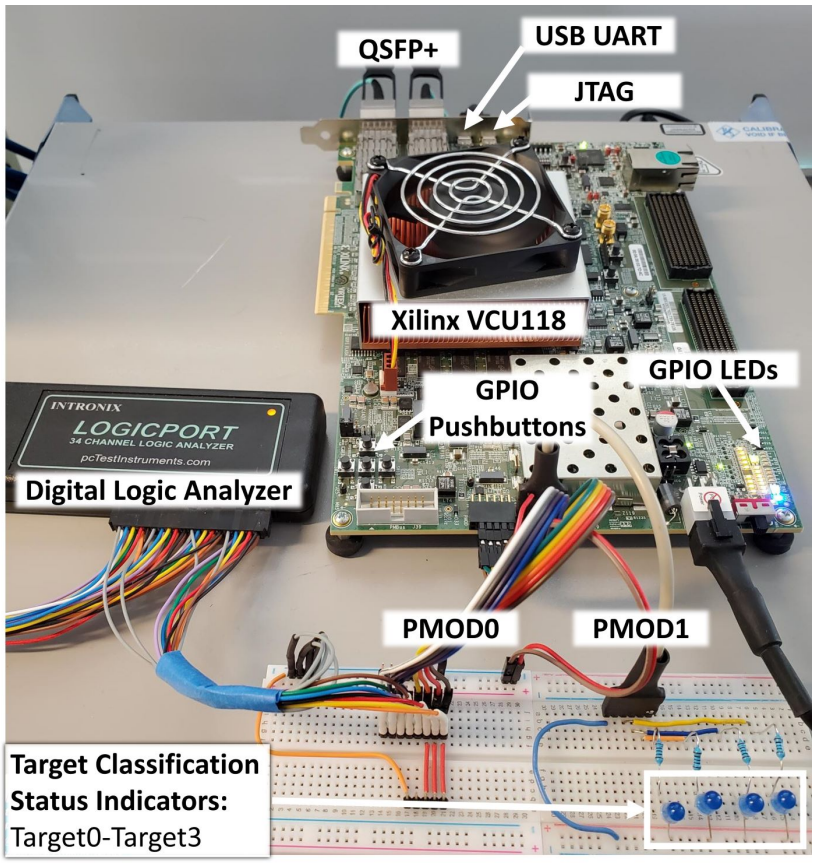


Figure 4.7. Hardware System IO Configuration

4.2.4 System IO: Clock Signals

The VCU118 is clocked by a 125 MHz system clock, generated by the onboard Silicon Labs Si55335A 1.8 V low-voltage differential signaling (LVDS) clock generator device (U122) [24]. Additionally, the programmable user clocks: Silicon Labs Si570 (U32) and Si53340 clock buffer (U104), provide the designer with the option of generating an application-specific clock signal ranging from 10 MHz to 810 MHz [24]. For this design, the QSFP+ clock is defined in the constraints file to support HS DIG IQ data transmission synchronization between the VCU118 and RF equipment. All QSFP+ data handling is configured within the proprietary R&S VHDL wrapper. Finally, the QSFP+ Jitter attenuated clocks are supplied to the system by the Silicon Labs Si5328B LVDS precision clock (U5) [24].

4.2.5 System IO: GPIO Input Switches

Figures 4.8 and 4.9 are images and schematic diagrams [24] of the onboard GPIO momentary pushbutton switch and GPIO duel inline package (DIP) switches, respectively. In Verilog, the normally-open pushbutton actuation is captured using a switch debouncing algorithm. An alternative and more effective method of achieving switch debouncing incorporates cross-coupled NAND gates, however, this involves modifying existing hardware. This pushbutton switch is used to trigger the initial transmit signal, **x**, upon system initialization, which is discussed later in Section 4.2.8. The 4-bit GPIO DIP switch input is used for system-level testing and development purposes.

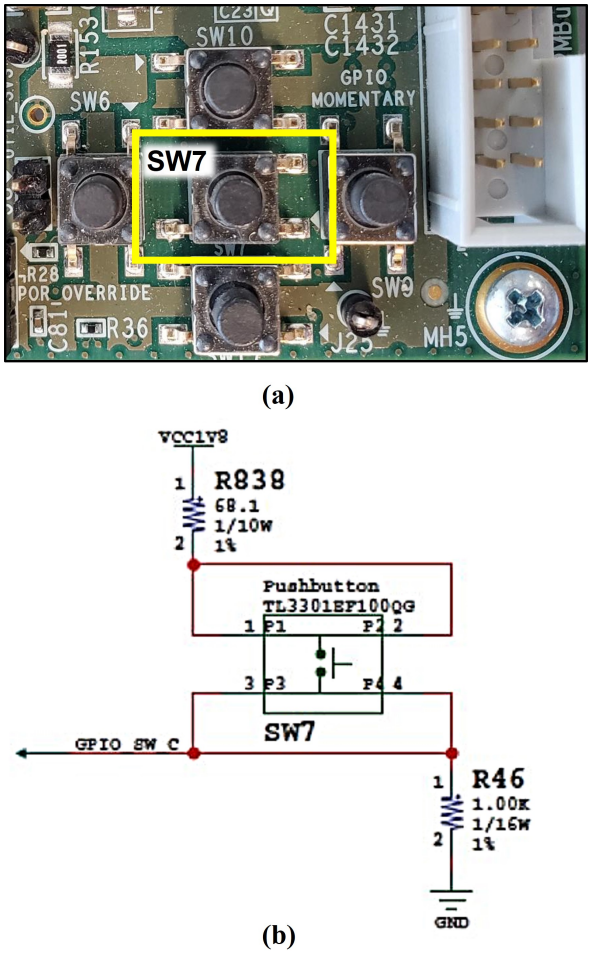


Figure 4.8. (a) Xilinx VCU118 Evaluation Board GPIO Momentary Pushbutton Input, SW7: Adaptive Waveform Trigger. (b) SW7 Schematic Diagram. Source [23]

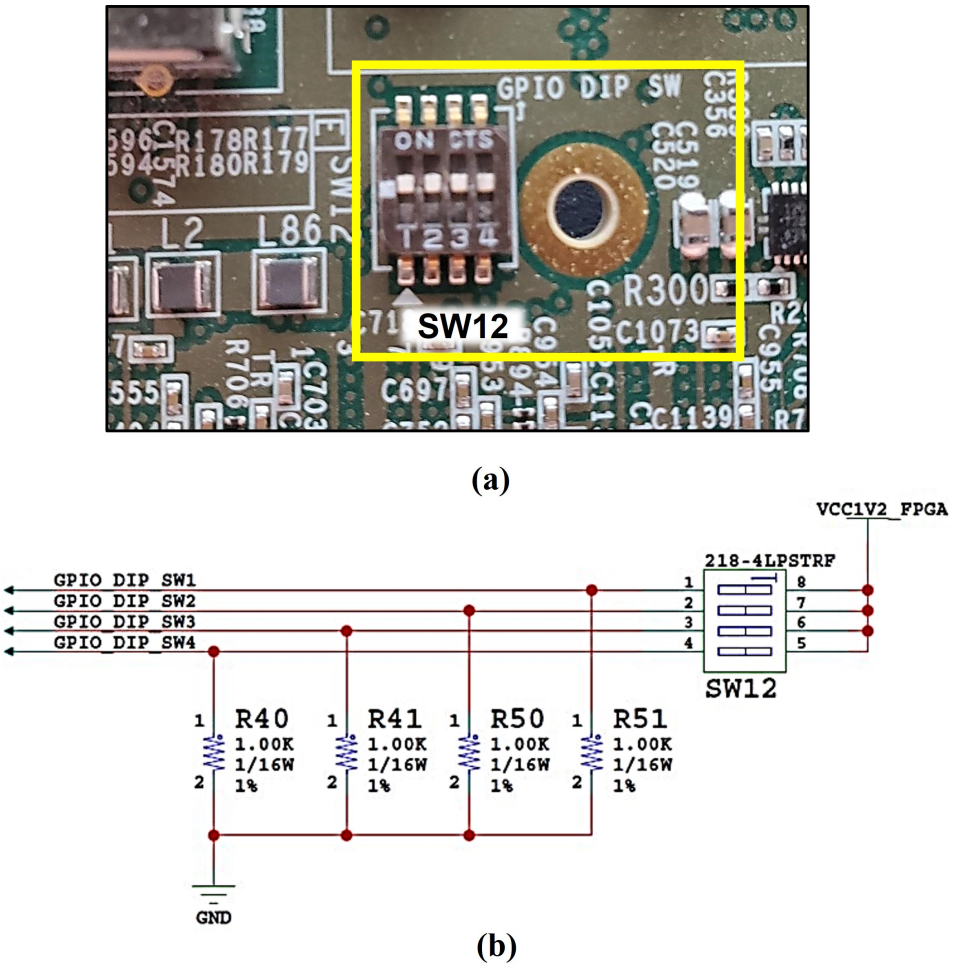


Figure 4.9. (a) Xilinx VCU118 Evaluation Board GPIO DIP Switch Input. (b) SW12 Schematic Diagram. Source: [23]

4.2.6 System IO: GPIO Status LEDs

The GPIO Status LEDs, featured in Figure 4.10, are used as system and communication status indicators. $LED0$-$LED3$ illuminate when the system is properly programmed and configured by the R&S equipment initialization software. $LED4$ and $LED7$ strobe when QSFP+ data communications are established. These functions are governed by the R&S IP core and VHDL wrapper.

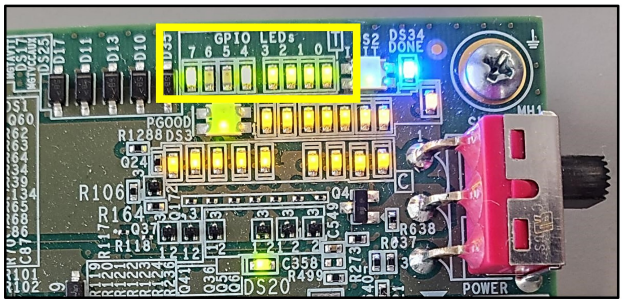
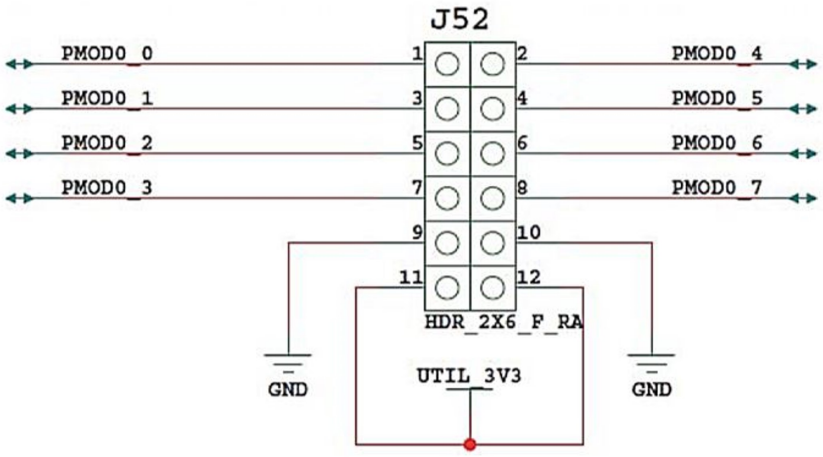


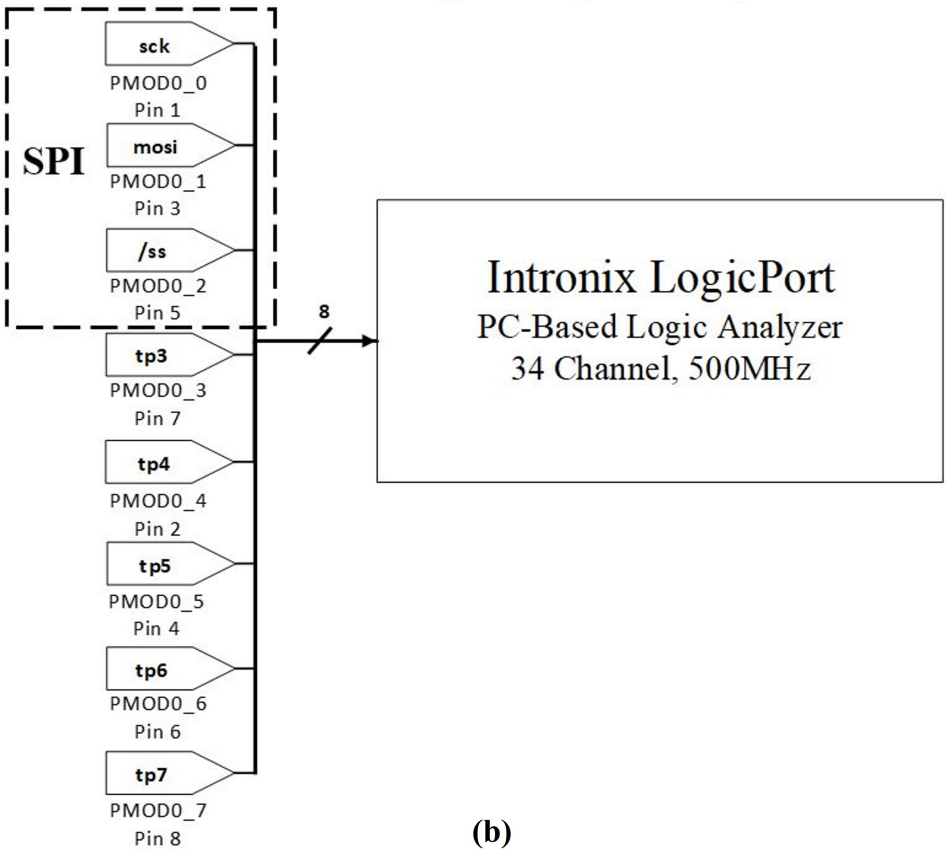
Figure 4.10. Xilinx VCU118 Evaluation Board GPIO Status LEDs.

4.2.7 System IO: PMOD Output Configuration

Figures 4.11 and 4.12 depict the two PMOD ports, $PMOD0$ and $PMOD1$. The PMOD pinout standard is generally used for Xilinx-based, application-specific hardware peripherals that can be purchased as modules, however, these ports can also be accessed and configured as custom IO in Verilog with respect to constraints file definitions. In this project, eight $PMOD0$ pins are assigned as test point that are accessed and interpreted by a PC-Based, 500 MHz Digital Logic Analyzer, shown in Figure 4.11. $PMOD0\_0 - PMOD0\_2$ are dedicated for 3-wire serial peripheral interface (SPI) signals used to display Q15.16 formatted data. $PMOD\_3 - PMOD\_7$ are used for signal and timing analysis.



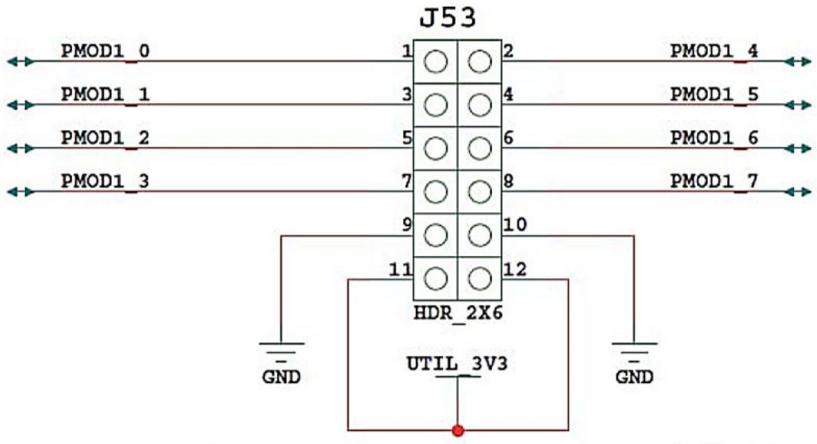
(a)



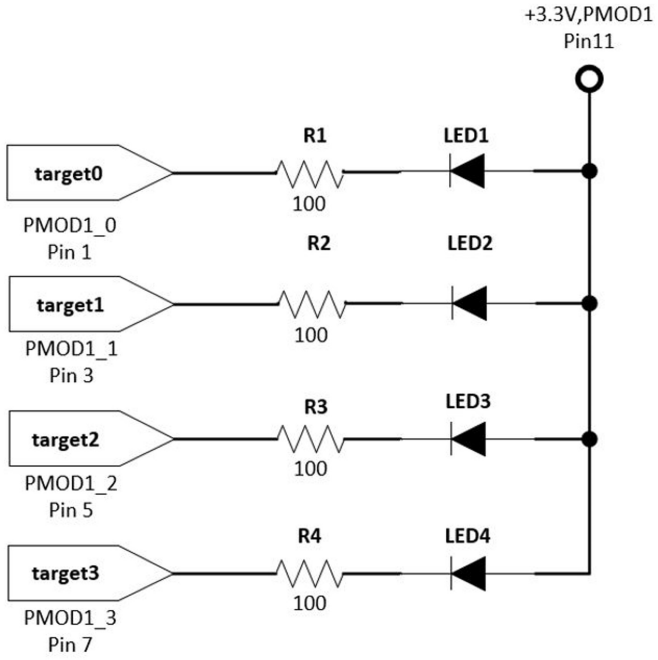
(b)

Figure 4.11. (a) Xilinx VCU118 Evaluation Board PMOD0 Right Angle Header Female Receptacle. (b) Output Test Signals to Logic Analyzer. Source: [23]

Shown in Figure 4.12, the $PMOD1$ port is used to display the target classification output decided by the system. $PMOD1\_0 - PMOD1\_3$ are devised in a simple active low LED schematic configuration. $LED1 - LED4$ correspond with $target0 - target3$ indicators.



(a)



(b)

Figure 4.12. (a) Xilinx VCU118 Evaluation Board PMOD1 Male Pin Header Output Signals. (b) Target Classification Indicator LEDs. Source: [23]

The complete VCU118 $PMOD0$ and $PMOD1$ connections and corresponding circuitry configuration described in Figures 4.11 and 4.12 are captured in Figure 4.13. Note that only eight channels of the 34 channel Digital Logic Analyzer are accessed. A standard 4 channel 200 MHz oscilloscope is also used during development to read PMOD outputs.

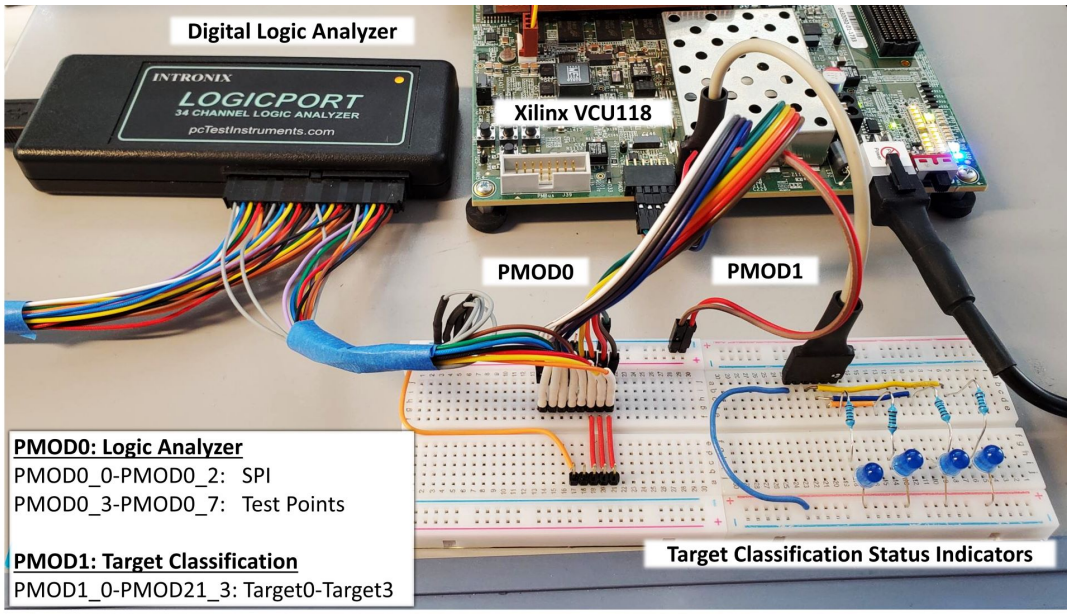


Figure 4.13. Xilinx VCU118 Evaluation Board, PMOD0 and PMOD1 Connections

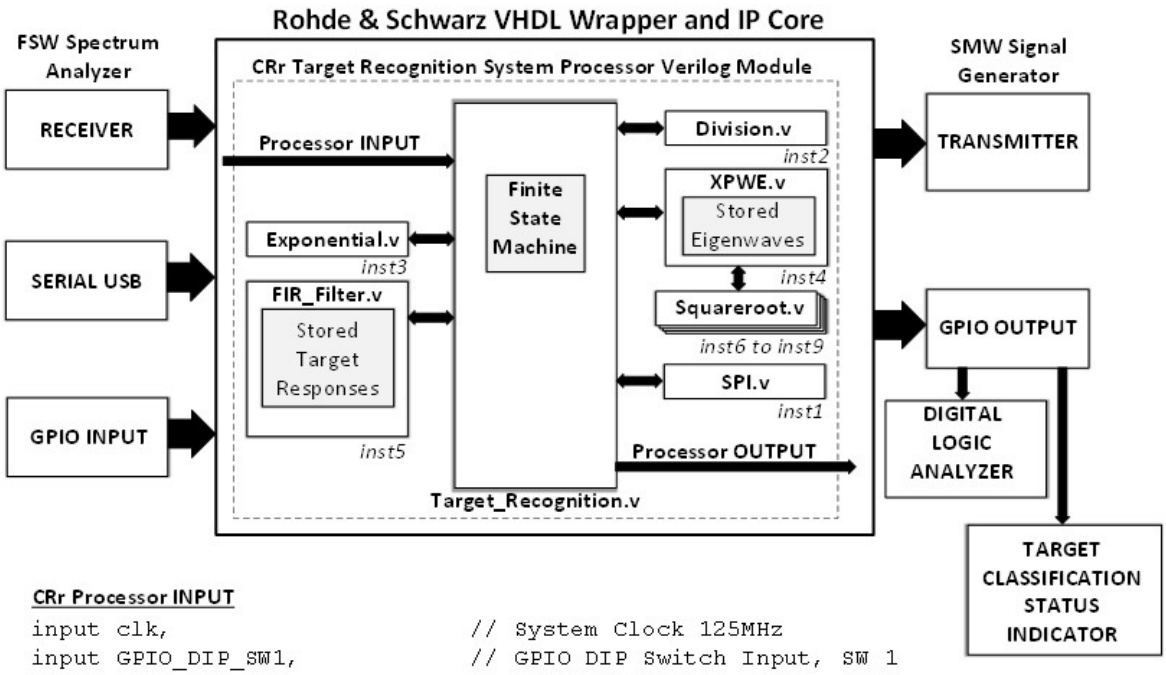
4.3 Functional Design Description and Demonstration

In this section, we provide a functional design description and demonstration of the CRr target recognition system, beginning with the top level Verilog design that is instantiated within the R&S VHDL wrapper. We first present an overview of the system level FSM and characterize the real-time sequential execution of the PWE algorithm in hardware. The Verilog modules that were developed in the Monte Carlo simulation model design are reused and modified. Modifications to the Verilog architecture include replacing the simulated Monte Carlo simulation model signals and test IO with the hardware system interconnections described in Section 4.2. We verify system functionality through target recognition status LEDs, the FSW and SMA200A graphical user interface displays, and VCU118 PMOD test points that are interpreted by the PC-based digital logic analyzer.

4.3.1 Top Level System Design Description

The mixed-language system architecture is illustrated in Figure 4.14 as a top level block diagram. The CRr target recognition system processor Verilog module is comprised of the division ($inst2$), exponential ($inst3$), XPWE ($inst4$), FIR filter ($inst5$), and square root ($inst6 - inst9$) modules described in Chapter 3. The SPI data handling module ($inst1$), adapted from [20], is added to provide the capability of viewing stored data and register states via the standard 3-wire, SPI data-write protocol. This custom module accepts a 32-bit Q15.16 formatted parallel input and streams a 32-bit serial data output ($mosi$) with respect to a 244 kHz clock signal (sck) and select signal ($/ss$). Additionally, the simulated pseudo-random noise generator of the Monte Carlo design is removed to allow the effective receiver noise (resulting from the spectrum analyzer and FPGA being used as a receiver) with the option of adding higher noise power with the $AWGNA$ generator well above the effective receiver thermal noise floor of the spectrum analyzer. The square root and division instances that were used to scale and normalize the adaptive transmit waveform, $\mathbf{x}$, in the original design are also omitted, as the SMW200A Signal Generator RF level is used to manually adjust the transmission power level.

The CRr target recognition system processor hardware module IO definitions are listed in Figure 4.14. All GPIO switch inputs and PMOD output pins are directly routed to the VCU118 per the standard constraints file definitions. The R&S equipment transceiver data passed through the CRr target recognition system processor module, RX_DATA and TX_DATA, are managed by the R&S VHDL wrapper and IP core, in addition to the serial USB communication to the PC-based R&S equipment initialization software.



CRr Processor INPUT

```
input clk,                      // System Clock 125MHz
input GPIO_DIP_SW1,             // GPIO DIP Switch Input, SW 1
input GPIO_DIP_SW2,             // GPIO DIP Switch Input, SW 2
input GPIO_DIP_SW3,             // GPIO DIP Switch Input, SW 3
input GPIO_DIP_SW4,             // GPIO DIP Switch Input, SW 4
input GPIO_SW_C,                // GPIO Momentary Push Button, Center
input RESET,                    // VCU118 System Reset
input [239:0] RX_DATA           // RX DATA, received waveform from QSFP1
```

CRr Processor OUTPUT

```
output reg [239:0] TX_XPWE,     // TX XPWE, transmit waveform to QSFP2
output reg PMOD0_0_LS,          // PMOD0_0, tp0, SPI scl
output reg PMOD0_1_LS,          // PMOD0_1, tp1, SPI mosi
output reg PMOD0_2_LS,          // PMOD0_2, tp2, /ss
output reg PMOD0_3_LS,          // PMOD0_3, tp3, Busy
output reg PMOD0_4_LS,          // PMOD0_4, tp4
output reg PMOD0_5_LS,          // PMOD0_5, tp5
output reg PMOD0_6_LS,          // PMOD0_6, tp6
output reg PMOD0_7_LS,          // PMOD0_7, tp7
output reg PMOD1_0_LS,          // PMOD1_0, LED0, Target 0
output reg PMOD1_1_LS,          // PMOD1_1, LED1, Target 1
output reg PMOD1_2_LS,          // PMOD1_2, LED2, Target 2
output reg PMOD1_3_LS           // PMOD1_3, LED3, Target 3
```

Figure 4.14. Top Level Block Diagram of the CRr Target Recognition System design in Verilog Instanced in a VHDL Wrapper

4.3.2 FSM, Synchronization Issues, and Functional Demonstration

The system FSM described in Figure 4.15 modifies the initial Monte Carlo simulation model by removing Monte Carlo-related iterative loops and incorporating RX_DATA and TX_DATA and other related system hardware IO into the algorithm. The FSM behavioral algorithm is defined within $Target\_Recognition.v$ and is comprised of 13 states.

State	Description
0	Scan for Trigger and DIP Switch Test Settings. Proceed to State 1 when SW7 is depressed. **INPUTS: SW7 & SW12** SW7 – Momentary Pushbutton, SW12 – 4 position DIP Switch
1	Generate Adaptive Transmit Waveform, **x** **XPWE.v** Probability update input *ptheta*[0:3] = 0.25
2	**FOR LOOP: *iter*[0:3]** Read Results from XPWE Module - Convert Q15.16 to R&S IQ Data Packet Format **OUTPUT: QSFP2, HS DIG IQ to Signal Generator** Transmit Waveform, **x**, via Signal Generator via QSFP2
3	Read Target Return Waveform, **y**, from Spectrum Analyzer **INPUT: QSFP1, HS DIG IQ from Spectrum Analyzer** Convert Received Input from R&S IQ Data Packet format to Q15.16
4	**FOR LOOP: *il*[0:3]** For each Target Response, Update: $\mathbf{S} = \mathbf{x} * \mathbf{h}_h$ **FIR_Filter.v**
5	Read Results from FIR_Filter.v, **S**
6	Likelihood Calculation (a): SS = re[**S**'**S**]
7	Likelihood Calculation (b): b = 2re[**S**'**y**]
8	Likelihood Calculation (c): re[b]-SS **If *il* < 4, Return to State 4**
9	Calculate PDF value **Exponential.v**
10	Calculate Probability Update for each Target Hypothesis **Division.v**
11	Update Adaptive Transmit Waveform, **x** **XPWE.v** Increment *iter* counter: **If *iter* < 4, Return to State 2**
12	Target Classification **OUTPUT: *GPIO LED*[0:3] (PMOD1), Logic Analyzer (PMOD0)** Display Target Selection based on highest probability **Return to State 0**

Figure 4.15. CRr Finite State Machine in Verilog

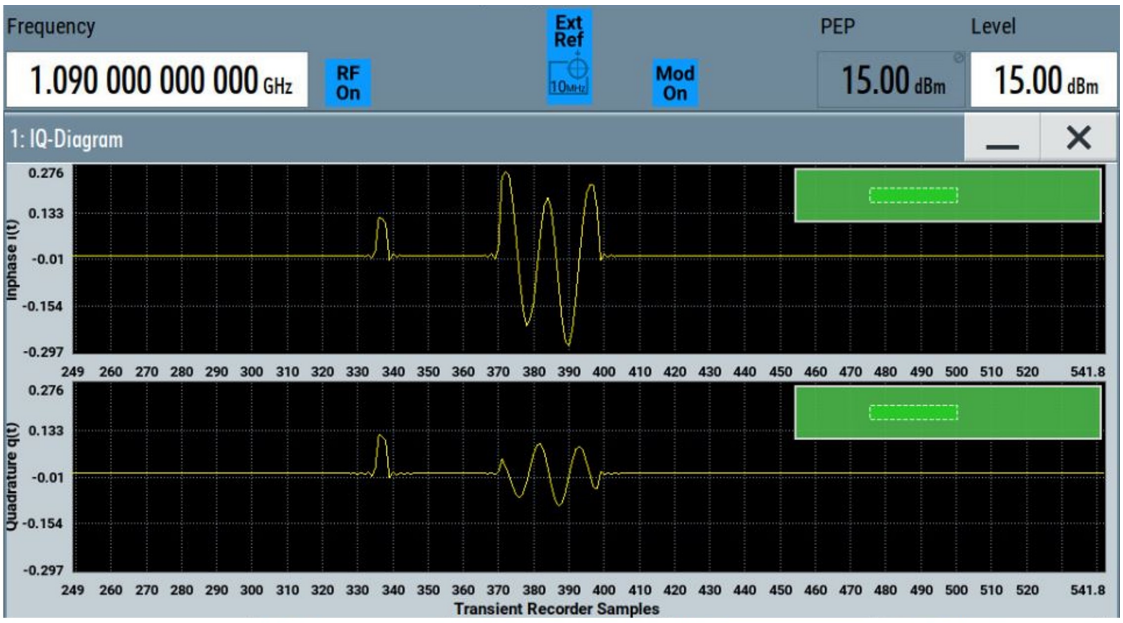
During the idle state, State 0, the system scans for the asynchronous input trigger (SW7) and captures the DIP switch settings (SW12). Using a Verilog defined debouncing algorithm, the state machine transitions to State 1 when SW7 is depressed. The initial complex-valued adaptive transmit signal, **x**, is then generated in the XPWE module using the stored eigenwaveforms and initial probability values, where registers $ptheta$[0:3] are set to 0.25.

In State 2, the data received from the XPWE module are translated in accordance to the R&S standard HS DIG IQ data bus format and sent to the QSFP2 output via the R&S VHDL wrapper. The HS DIG IQ baseband data are received by the SMW200A Vector Signal Generator, and then then transmitted as IQ modulated RF. Figure 4.16a shows the initial adaptive waveform, **x**, displayed on the IQ diagram of the SMW200A graphical user interface. The transmitter power level is set to 15 dBmW with a carrier frequency of 1.090 GHz.

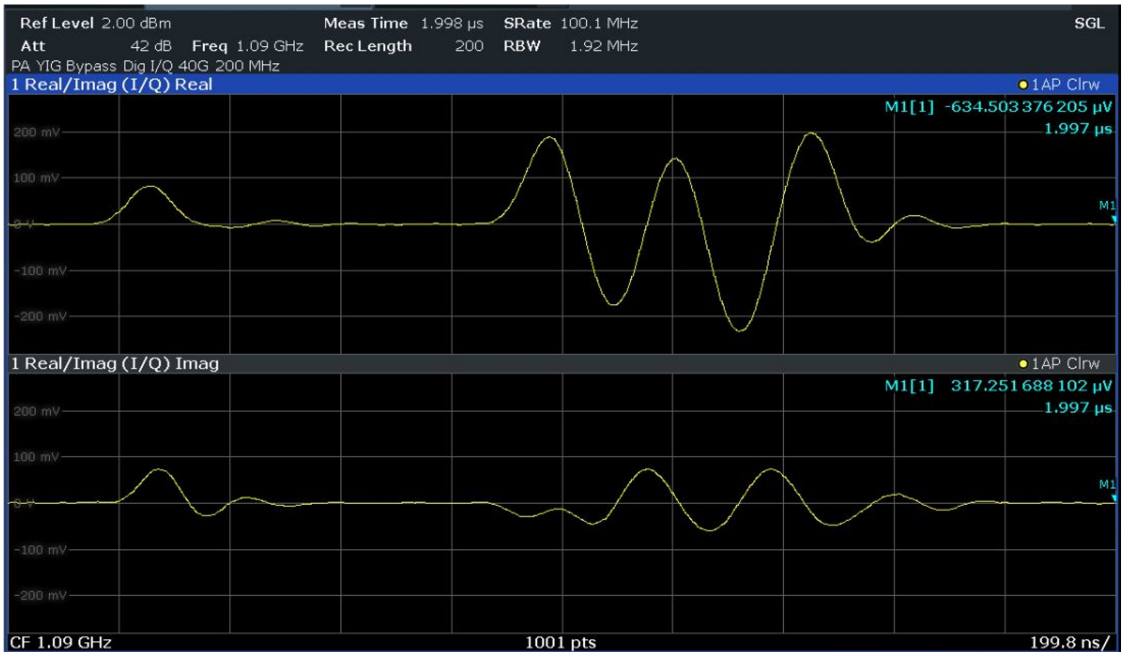
Note that the Vivado implementation in Chapter 3 assumes full synchronization of the MATLAB implementation of the PWE. For the FPGA-in-the-loop in an RF setting, discrete time synchronization of the FPGA with the RF equipment can be very difficult to achieve unlike a true dedicated receiver where total synchronization can be truly designed such that all receiver component timing can be aligned.

As shown in Figure 4.16 (a), an initial reference pulse is appended at the beginning of each of the IQ adaptive waveform signals to facilitate target recognition module synchronization and enable receiver detection. This technique can help ensure that the received target return signal aligns with the matched waveform values that are calculated in the FPGA. However, because the IQ phase outputs from the RF signal generator are observed to drift over time after system initialization, manual RF phase calibration is often required. The RF phase output is adjusted at the signal generator until the IQ reference pulses approximately match in amplitude.

A screen capture of the FSW Signal and Spectrum Analyzer interface is depicted in Figure 4.16 (b) by displaying the complex-valued target return signal, resulting from the transmitted waveform interaction with the baseband response, **h0**. The FSW converts the received RF signal into HS DIG IQ baseband data format via QSFP1. The data first enter the receiver module of the R&S VHDL wrapper, which is then sent to an internal first-in, first-out (FIFO) receiver buffer, RX_FIFO.



(a)



(b)

Figure 4.16. (a) Initial Transmitted Waveform, **x**, with Reference Pulse Transmitted by SMW200A and (b) Target Return Waveform, **y** with Reference Pulse Captured by FSW, where $ptheta[0{:}3] = 0.25$

In State 3, the target recognition module receiver monitors the RX_FIFO output and searches for the initial reference pulse of the transmitted signal using a conditional statement with predefined thresholds. As shown in the Logic Analyzer display of Figure 4.17, once the reference pulse is detected, a read window enable signal, RX_READ, is asserted, capturing and storing 240 samples. The complex-valued HS DIG IQ data is translated into Q15.16 format through a sign extension process, down-sampled to the 61 sample design format, and stored in registers yy[60:0] and yy_j[60:0].

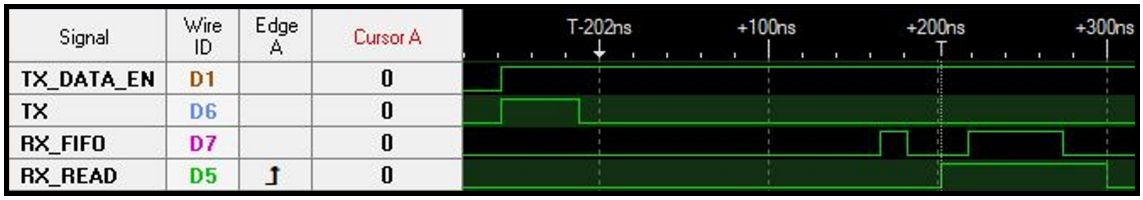
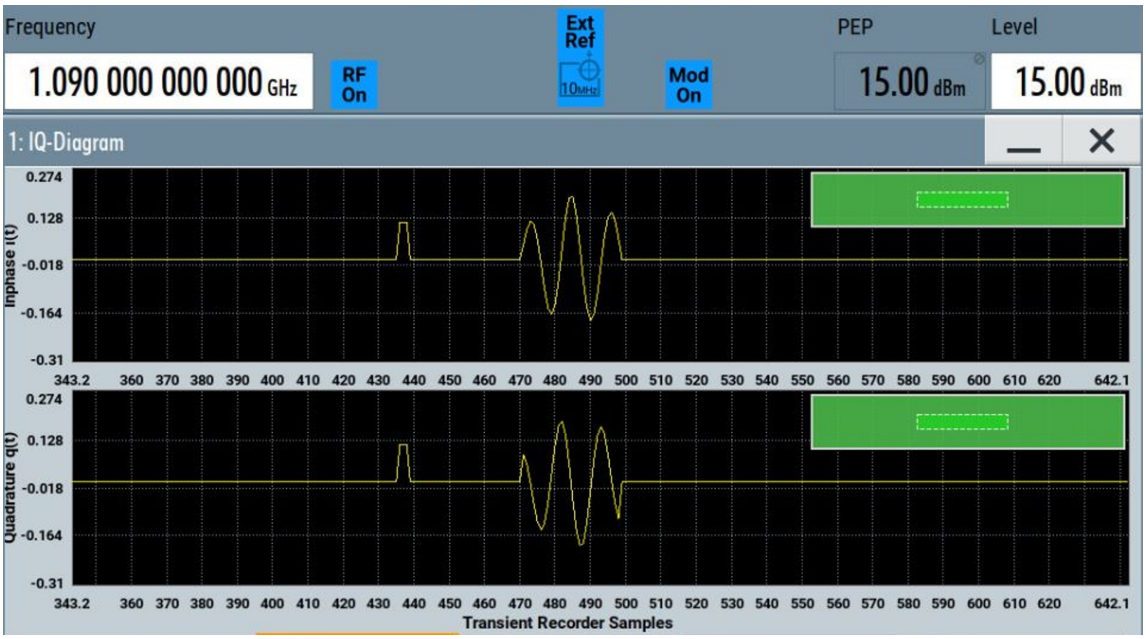


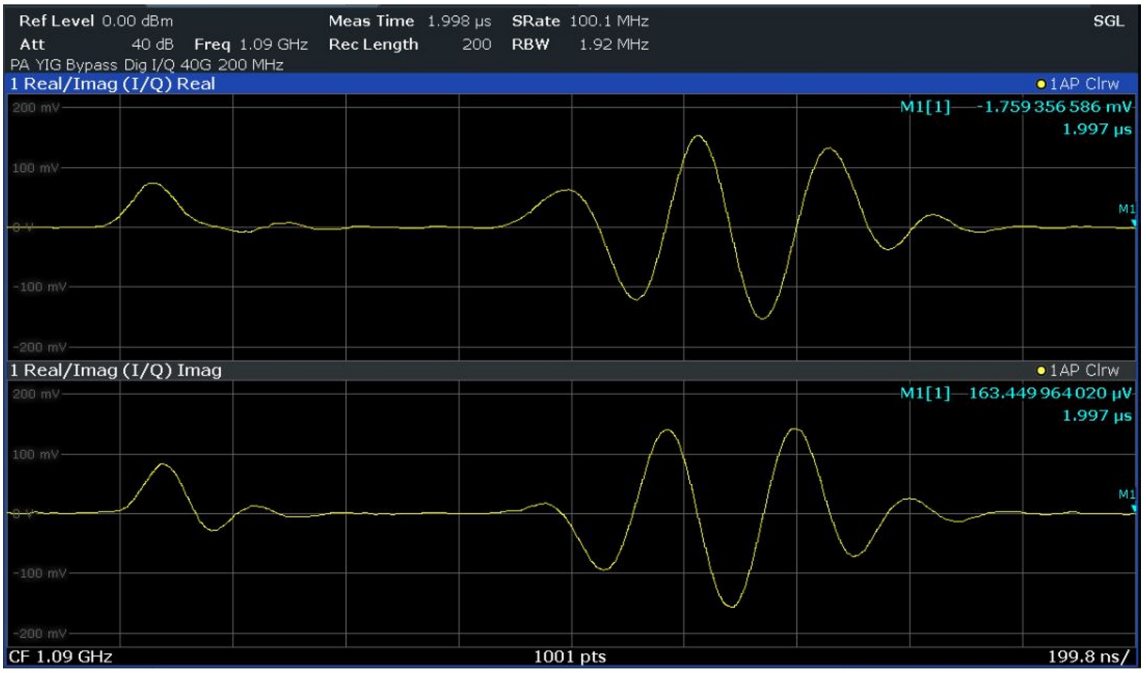
Figure 4.17. Timing Analysis of TX/RX HS DIG IQ Data Captured on the Digital Logic Analyzer

In a separate process internal to the FPGA, **x** is convolved with each of the stored eigen-waveforms, represented as $\mathbf{h}_h$, during States 4 and 5, which results in **S**. In States 6 through 9, **S** and the received data, **y**, are used to calculate the likelihood values by evaluating each target hypothesis PDF. The updated probabilities, $ptheta$[0:3], are then calculated in State 10.

Using the updated probabilities calculated in State 10, the complex-valued adaptive waveform is generated and re-transmitted during State 11 using the XPWE module and R&S VHDL support architecture. Figure 4.18 provides an example of subsequent transmit/receive waveforms, where the probability update determined by the system for **h0** is $ptheta$0=0.9987. The corresponding probability update is displayed on the top Logic Analyzer SPI capture of Figure 4.19, and the $ptheta$0 history for target0 is shown at the bottom. The state machine is then redirected to State 3, where the adaptive transmit, receive, and process cycle of States 3 through 11 are repeated for a total of 4 transmissions, as shown in Figure 4.20.



(a)



(b)

Figure 4.18. (a) Subsequent Adaptive Waveform, $\mathbf{x}$, Transmitted by the SMW200A Signal Generator and (b) Target Return Waveform, $\mathbf{y}$, Captured by the FSW Spectrum Analyzer (Bottom), where $ptheta0 = 0.9987$.

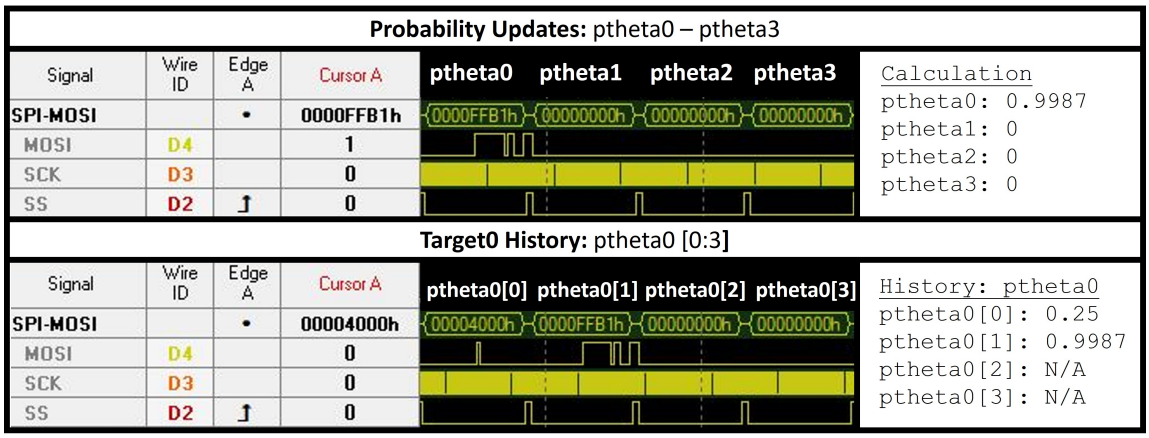


Figure 4.19. Digital Logic Analyzer Displaying the Update Probabilities (Top) and $ptheta0$ History for $target0$ after 2 Transmit/Receive Iterations (Bottom)

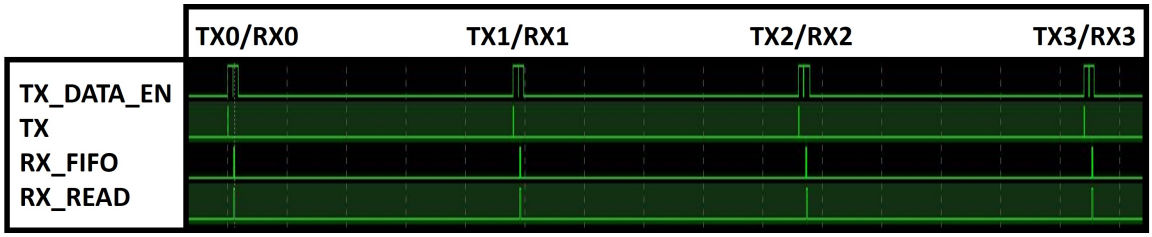


Figure 4.20. Digital Logic Analyzer Displaying 4 Adaptive Waveform HS DIG IQ Transmit and Receive Cycles

After the last transmission, the maximum probability value is selected by the system from the $ptheta[0:3]$ values, in State 12 through a set of conditional statements. The corresponding target LED is then turned ON based on the target classification decided by the system. In this demonstration, Figure 4.21, confirms that the system correctly classifies the bandpass filter as $target0$. Additionally, the logic analyzer screen capture of Figure 4.22 depicts the last probability values corresponding to each target hypothesis, as well as the probability update history of $target0$ corresponding to each transmission. Finally, the last adaptive waveform is displayed in Figure 4.23. The system returns to the idle state, State 0, where this last waveform is re-transmitted in periodic intervals for user observation.

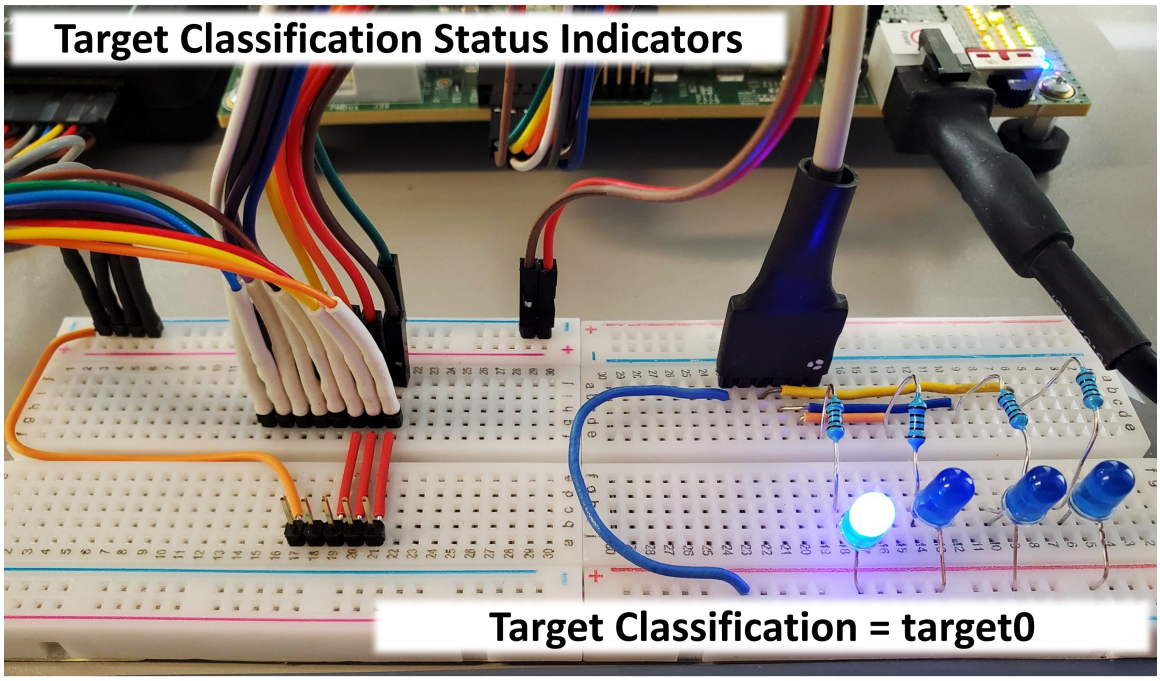


Figure 4.21. Target Classification Status Indicators, $target0$ Determined by System

Target Classification Decision: ptheta0 – ptheta3

Signal	Wire ID	Edge A	Cursor A	ptheta0	ptheta1	ptheta2	ptheta3
SPI-MOSI		•		00010000h	00000000h	00000000h	00000000h
MOSI	D4		0				
SCK	D3		0				
SS	D2	↑	1				

```
Final Calculation
ptheta0: 1
ptheta1: 0
ptheta2: 0
ptheta3: 0
```

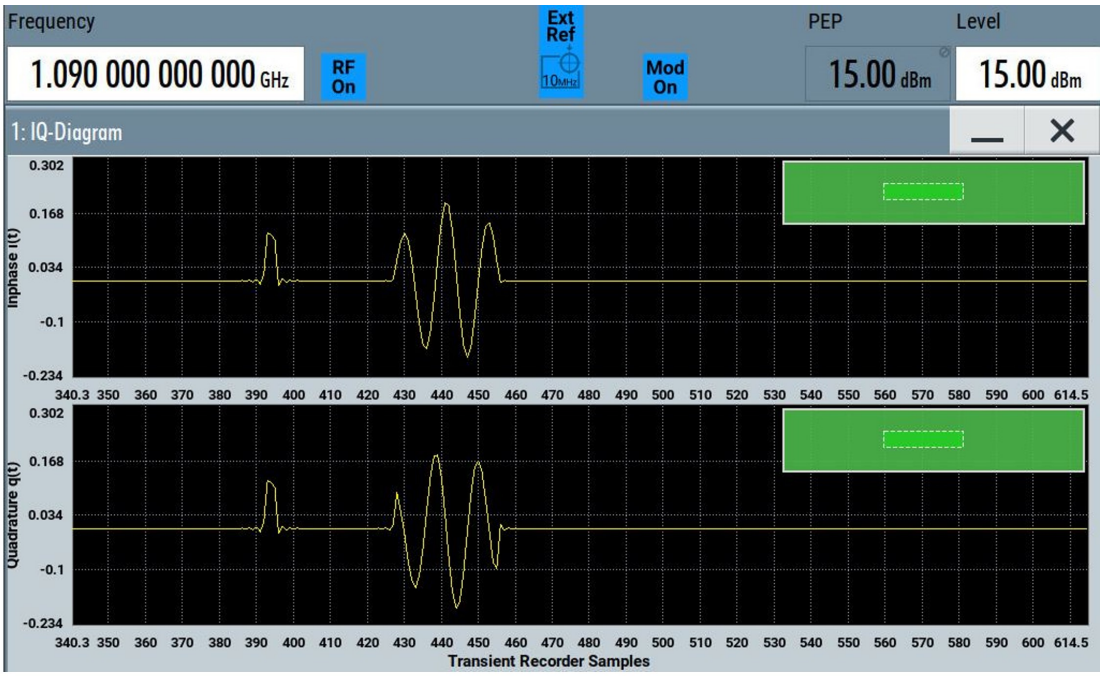
Target0 History: ptheta0 [0:3]

Signal	Wire ID	Edge A	Cursor A	ptheta0[0]	ptheta0[1]	ptheta0[2]	ptheta0[3]
SPI-MOSI		•		00004000h	0000FFB1h	00010000h	00010000h
MOSI	D4		0				
SCK	D3		0				
SS	D2	↑	1				

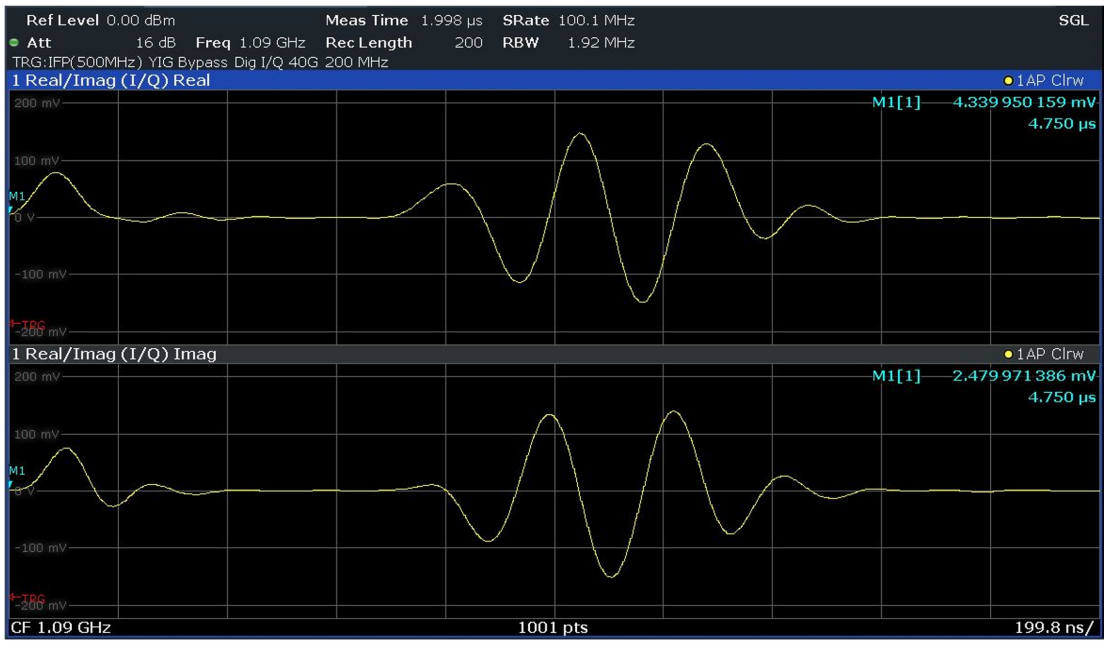
```
History: ptheta0
ptheta0[0]: 0.25
ptheta0[1]: 0.9987
ptheta0[2]: 1
ptheta0[3]: 1
```

Target Classification Decision: target0, ptheta0 = 1

Figure 4.22. Digital Logic Analyzer Data: Target Classification Decision (Top) and $ptheta0$ History of $target0$ Determined by System



(a)



(b)

Figure 4.23. (a) Last Adaptive Waveform, **x**, Transmitted by the SMW200A Signal Generator and (b) Target Return Signal, **y**, Captured by the FSW Spectrum Analyzer (Bottom), where $ptheta0 = 1.0$

4.4 Hardware Implementation Observations

The functional demonstration described in Section 4.3 confirms that the PWE CRr target recognition theory and technique can be implemented into real-time hardware applications. In this proof-of-concept demonstration, an FPGA-based design is used to perform all cognitive radar decision-making and processing requirements within a fully adaptive, closed-loop transceiver system using R&S RF test equipment. This real-world digital hardware demonstration also confirms that it is possible to incorporate this technique into other processor technologies, such as digital signal processor (DSP), system on chip (SoC), or application specific integrated circuit (ASIC) based designs. Most importantly, the PWE technique can be directly applied in modern radar systems.

The importance of receiver synchronization is critical in this design to properly read received HS DIG IQ data. As stated in Section 4.3.2, the initial reference pulse of the transmit waveform is used to dictate the read window at the FIFO receiver buffer. Misinterpreting the reference pulse introduces synchronization error, which results in recording erroneous data. This, of course, leads to incorrect target classification if the rest of the processing continues. The reference pulse in this design is observed to be sensitive to noise. When $AWGNA$ generator noise is added to the transmit signal significantly above the effective receiver thermal noise floor, the additional noise intermittently trips the read window threshold and introduces synchronization error. Notice that the reference pulse has lower energy than the adaptive waveform. Thus, even if a large carrier to noise (SNR) is set in the RF signal generator, the reference pulse has lower SNR. In the implementation, we did not have a matched filter to the pulse since that would have to be matched to the RF carrier phase (which depends on pressing the initial trigger to the experiment). That level synchronization to ensure maximum energy is captured from the reference pulse is beyond the scope of this work. Future designs can increase the energy of the reference pulse to reduce noise sensitivity. Other considerations include reducing additional layers of synchronization by removing the FIFO receiver buffer from the design and capturing data directly from the QSFP+ receiver, since each target return waveform is only 240 samples wide.

In the Vivado implementation, it had already been shown that the PWE technique is robust via the Monte Carlo simulation method where the performance degradation is dictated by hardware implementation. For more robust RF demonstrations, synchronization is more of an issue.

4.5 Chapter Summary

In this chapter, we demonstrated a hardware realization of the PWE CRr target recognition technique and theory through the implementation of an FPGA-in-the-loop CRr design integrated with the R&S RF equipment. From this demonstration, we confirmed the functional feasibility of incorporating PWE concepts into real-time hardware applications. In the beginning of the chapter, we first provided a description of the system hardware, equipment, and IO components. We then provided a functional demonstration of the CRr target recognition system by stepping through the FSM process. Finally, we concluded this chapter with design implementation observations, as well as recommendations for future design optimization. In the next and final chapter, we summarize this complete body of work and provide recommendations for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Summary and Conclusion

The objective of this study was to design an FPGA implementation of a cognitive radar (CRr) target recognition system utilizing the probability of weighted energy (PWE) adaptive waveform transmission technique. This thesis served a proof-of-concept investigation to determine the feasibility of incorporating the PWE adaptive matched waveform theory into a real-time hardware application and/or demonstration.

We first defined the design objectives and background in Chapter 1 by dividing this study into two main focus areas: (1) develop and simulate the PWE CRr target recognition method as a Verilog digital system logic design through a Monte Carlo simulation model using the Xilinx Vivado development environment and (2) design and demonstrate a functional digital hardware implementation of a PWE CRr target recognition system in a dynamic closed-loop, fully adaptive transceiver design using an FPGA-based processor integrated with R&S RF equipment. In Chapter 2, we provided the conceptional background of the CRr perception-action cycle model and introduced the PWE theory.

In Chapter 3, we designed the Verilog architecture of a PWE Monte Carlo simulation model consisting of 15,000 trials. The MATLAB PWE algorithm [16] was employed as a benchmark to develop the required digital hardware logic and compare performance. Data structure standardization incorporated throughout the Verilog system modules used 32-bit signed fixed-point Q15.16 representation to accommodate data precision requirements with the Xilinx VCU118 and R&S proprietary data bus formats. The overall outcome of the Verilog Monte Carlo model simulation confirmed that the PWE target recognition technique can be realized in a hardware description language. Additionally, through this design model, we produced and verified the functional Verilog module components suitable for FPGA implementation. Considerations for optimizing this design include incorporating the CORDIC algorithm for exponential functions [21], improving the square root function using techniques such as the "Babylonian Method," and implementing more advanced division algorithms beyond the elementary Euclidean method to gain precision.

Finally, in Chapter 4, we designed and exhibited an integrated FPGA-in-the-loop RF hardware demonstration of the PWE CRr target recognition technique with R&S RF equipment. In this chapter, we first detailed the equipment features and hardware system interconnections of the Xilinx VCU118 Evaluation Board, SMA200A Vector Signal Generator, and FSW Signal & Spectrum Analyzer. Additionally, we described the mixed language architecture, illustrating the Verilog CRr target recognition system design instantiated within the R&S VHDL wrapper. We then provided a functional demonstration of the system FSM. The design outcome produced a proof-of-concept demonstration confirming the feasibility of employing the PWE CRr target recognition technique into real-time hardware applications. This small-scaled hardware approach exhibited that the PWE concepts can be directly applied in modern radar systems using processor technology extending beyond FPGAs, such as DSPs, SoCs, or custom ASICs. Recommendations that we discussed for optimizing this test system design include improving transmit and receiver synchronization of the experiment setup.

5.1 Recommendations for Future Work

Recommendations for future work can be divided into two focus areas based on levels complexity: (1) further proof-of-concept PWE CRr development expanding from the test system presented in this thesis and (2) PWE integration into a true radar system.

The test system setup and environment presented in this project can be further optimized and expanded using practical target responses, such as military aircraft, naval warship hulls, or ground utility vehicles, to fully demonstrate a realistic application of PWE target recognition. For example, in [8], high fidelity EM-simulated radar cross section (RCS) response models are incorporated into the PWE simulation to represent actual military target sets. Using the test setup design of this thesis, the $target0$ bandpass filter can be replaced with a second FPGA configured as a programmable FIR filter that is positioned in series between the SMA200A and FSW connections. This programmable FIR filter can incorporate the Verilog filter module developed in this project with the stored military RCS responses. Adding a layer of complexity, aspect angles should be considered in this approach, as demonstrated in [8].

This test equipment setup can also be utilized to apply the jammer nulling adaptive waveform EM countermeasure technique developed in [9] as a hardware demonstration. This countermeasure incorporates PWE concepts and RCS models [8] to classify aircraft in the presence of jammer interference [9].

In addition to the test system setup improvements and optimization recommendations from Section 4.4, future work recommendations include incorporating the pulse descriptor word package feature on the R&S SMA200A to provide enhanced streaming IQ generation and direct control of waveform parameters over the local area network Ethernet connection between the VCU118 and SMA200A [25]. This method would provide increased flexibility and automation to shape FPGA generated waveforms, as well as eliminate manual adjustments required in the current system design.

Furthermore, implementing parallel processing architecture features of the VCU118 Evaluation Board is recommended in future improvements to this design to incorporate concurrent computing and enhance Virtex Ultrascale+ FPGA utilization. The onboard Xilinx Zynq 7000 SoC features a dual core ARM processor [24], which can be used to share the sequential behavioral load of the FSM currently residing in the target recognition Verilog module. This would free up processing resources and real estate on the Virtex Ultrascale+ and improve overall system performance. Additionally, stored target responses, eigenwaveforms, and LUTs can be stored onto the onboard non-volatile memory devices featured on the VCU118.

Finally, in a more complex application, to fully demonstrate PWE CRr target recognition methods at the operational level, future research recommendations involve incorporating the algorithm in a modern radar system, expanding beyond the test environment presented in this thesis, and using existing and proven transceiver designs in the field. The hardware realization and demonstration presented in this body of work at the simplistic level provides confidence that a functional application of the PWE method in a true radar system is possible.

List of References

[1] *Joint Electromagnetic Spectrum Operations*, Joint Publication 3-85, US Department of Defense, Washington, DC, USA, 22 May 2020.

[2] S. Gurbuz, H. Griffiths, A. Charlish, M. Rangaswamy, M. Greco, and K. Bell, "An overview of cognitive radar: Past, present, and future," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, pp. 6–18, 12 2019.

[3] S. Haykin, "Cognitive radar: a way of the future," *IEEE Signal Processing Magazine*, vol. 23, no. 1, pp. 30–40, 2006.

[4] *Cognitive Radar*, Science and Technology Organization Technical Report, TR-SET-227, North Atlantic Treaty Organization, Washington, DC, October 2020.

[5] R. A. Romero and N. A. Goodman, "Improved waveform design for target recognition with multiple transmissions," in *2009 International Waveform Diversity and Design Conference*, 2009, pp. 26–30.

[6] R. Romero, "Matched waveform design and adaptive beamsteering in cognitive radar applications," Ph.D. dissertation, The University of Arizona, Tucson, AZ, USA, 2010. Available: http://hdl.handle.net/10150/194499

[7] Q. J. O. Tan and R. A. Romero, "Jammer-nulling transmit-adaptive radar against knowledge-based jammers in electronic warfare," *IEEE Access*, vol. 7, pp. 181 899–181 915, 2019.

[8] Q. J. O. Tan, R. A. Romero, and D. C. Jenn, "Target recognition with adaptive waveforms in cognitive radar using practical target rcs responses," in *2018 IEEE Radar Conference (RadarConf18)*, 2018, pp. 0606–0611.

[9] Q. J. O. Tan and R. A. Romero, "Jammer nulling adaptive waveforms with cognitive radar for aircraft rcs recognition in presence of frequency sweep and base jammers," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 1339–1343.

[10] J.-Y. Nieh and R. A. Romero, "Integrated range-doppler map and extended target identification with adaptive waveform for cognitive radar," in *2015 IEEE Radar Conference (RadarCon)*, 2015, pp. 1644–1649.

[11] J.-Y. Nieh, "Integrated range-doppler map and extended target classification with adaptive waveform for cognitive radar," Ph.D. dissertation, Department of Electrical

and Computer Engineering, Naval Postgraduate School, Monterey, CA, USA, 2014 [Online]. Available: https://calhoun.nps.edu/handle/10945/44632

[12] B. M. Bey and R. A. Romero, "Clutter-compensating adaptive waveforms with cognitive radar for target classification using em-simulated ground-based rcs responses," in *2019 16th European Radar Conference (EuRAD)*, 2019, pp. 9–12.

[13] E. Mourtzakis, "Waveform design compensation for moving extended targets," M.S. thesis, Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, USA, 2013. [Online]. Available: https://calhoun.nps.edu/handle/10945/34710

[14] R. Romero and E. Mourtzakis, "Transmit energy efficiency of two cognitive radar platforms for target identification," *Aerospace*, vol. 2, no. 3, pp. 376–391, 2015. Available: https://www.mdpi.com/2226-4310/2/3/376

[15] Xilinx, *Vivado Design Suite 2019.2*, [Software]. San Jose, CA, USA, 2019.

[16] R. A. Romcero, *Deterministic Targets in AWGN, Waveform Design using Probability of Weighted Energy*, [Software]. Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, USA, 2010.

[17] N. A. Goodman, P. R. Venkata, and M. A. Neifeld, "Adaptive waveform design and sequential hypothesis testing for target recognition with active sensors," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 1, pp. 105–113, 2007.

[18] R. J. Abad, M. H. Ierkic, and E. I. Ortiz-Rivera, "Basic understanding of cognitive radar," in *2016 IEEE ANDESCON*, 2016, pp. 1–4.

[19] O. A. Yakimenko, *Engineering Computations and Modeling in MATLAB/Simulink*, 1st ed. Reston, Va: American Institute of Aeronautics and Astronautics, 2011, ch. 8.

[20] C. Ünsalan and B. Tar, *Digital System Design with FPGA: Implementation Using Verilog and VHDL*, 1st ed. New York: McGraw Hill, 2017, ch. 6, 12.

[21] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. New York: Oxford University Press, 2010, ch. 13, 21, 22.

[22] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design*, 3rd ed. Toronto, Canada: McGraw Hill, 2014, ch. 5,11, pp. 326,670–673.

[23] Rohde & Schwarz USA, Inc, *VCU118 Intellectual Property Files*, [Software]. Columbia, MD, USA, 2020.

[24] *VCU118 Evaluation Board User Guide*, UG1224 (v1.4), Xilinx, Inc, 2018. [Online]. Available: https://www.xilinx.com/support/documentation/boards_and_kits/vcu118/ug1224-vcu118-eval-bd.pdf

[25] *R&S SMW200A Vector Signal Generator User Manual*, 1175.6632.02 Version 29, Rohde & Schwarz GmbH & Co. KG, Munchen, Germany, 2021. [Online]. Available: https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/pdm/cl_manuals/user_manual/1175_6632_01/SMW200A_UserManual_en_29.pdf

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California