MEE10:81

Observer Implementation in an FPGA Using Simulink and Xilinx System Generator

Omer Farooq

Bodiuzzaman Molla

This thesis is presented as part of Degree of Master of Science in Electrical Engineering

Blekinge Institute of Technology

August 2010

Blekinge Institute of Technology
School of Engineering
Department of Electrical Engineering
Supervisors: Anders Hultgren
: Carina Nilsson
Examiners: Anders Hultgren
: Carina Nilsson

This thesis is submitted to the Department of Signal Processing, School of Engineering at Blekinge Institute of Technology in partial fulfillment of requirement for the degree of Master of Science in Electrical Engineering with emphasis on Signal Processing.

Contact Information:

Author:

Omer Farooq
Email: omerfarooq_13@hotmail.com, omfa08@student.bth.se

Bodiuzzaman Molla
Email: engr.bzaman@gmail.com, bomo08@student.bth.se

University advisor(s):

Anders Hultgren
Email: anders.hultgren@bth.se

Carina Nilsson
Email: carina.nilsson@bth.se

School of Engineering
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

web: www.bth.se/tek
Phone: +46 457 385 000
Fax: +46 457 271 29

Abstract

In order to reduce the cost of the current switched resonant power converters, a state observer is introduced to provide the controller with the required information for the effective switching of the converter without the need of expensive sensory inputs from the resonant circuit.

This Master's thesis investigates the usage of Matlab/Simulink and Xilinx System Generator as an implementation tool for Field Programmable Gate Array (FPGA) development of the Observer Model. The main aim is to keep the model size to a minimum while keeping the error within a reasonable range, allowing the observer to converge.

Keywords: State observer, Hamiltonian Modelling, Stability of observer, FPGA.

Acknowledgment

We would like to thank our parents and friends who supported us throughout our thesis work in Sweden.

We would like to thank our supervisor from Blekinge Tekniska Hogskolan (BTH) Anders Hultgren, who provided us with valuable time and addressed the problems that we faced.

We would also like to thank our co-supervisor Carina Nilsson, who provided us with all the support which enabled us to finish our thesis work. Her priceless guidance through the difficult phases of the work ensured the completion of this project.

We would also like to thank the staff at ALSTOM POWER SYSTEMS AB, VAXJÖ who provided us with a very friendly environment.

Finally we would like to thank everyone who was involved in this project.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem statement

This thesis work tries to answer the following research questions:

- How to use Matlab Simulink and Xilinx System Generator as an FPGA implementation tool?
- How to keep the observer model size small while keeping the error to a minimum?
- Is the current available hardware sufficient enough for the 4th order Hamiltonian observer implementation?

1.2 Thesis outline

We have tried to systematically describe the background, different phases, steps and the conclusion regarding this thesis work.

Chapter 1 describes the problem that is at hand.

Chapter 2 explains briefly the whole system and state of the art i.e. the previous work done in this area.

Chapter 3 gives an idea about how we shall tackle the problem, our approach and the tools we decided to use for implementation.

Chapter 4 explains the design and the implementation work we performed, the challenges we faced, the results we acquired and the recommendations we made in a detailed way.

Chapter 5 is basically a manual explaining in detail all the steps required for the VHDL code generation and its implementation.

Finally in chapter 6 we concluded all of our experiences and a few recommendations regarding the future work.

Chapter 2

Background and Related work

2.1 Background

The observer is part of a new generation of power converters, being developed by ALSTOM POWER SYSTEMS, to be used in an industrial application.

The Power converter consists of a controller, a switched resonant circuit and an observer.

The controller controls the switching of the switched resonant circuit. The controller can supply either positive or negative conduction depending on a comparison between the current and the voltage in the resonant circuit and a reference table. While switching between the positive and negative supply and vice verse, the controller switches to a non conduction state for $1\mu s$ to prevent any short circuiting [1].

The power converter is based on a resonant circuit as shown by the Figure 2.1. Some basic analysis of the power converter can be found in [2]. The load is connected in series with the resonant circuit using a full wave bridge rectifier as a capacitative load. The voltage across the resonant circuit is controlled with the help of four switches Z_1, Z_2, Z_3 and Z_4. The switching of these four transistors determine the supply mode of the power converter. The power converter can be in three conducting states, either conducting in positive direction, negative direction or not conducting state [3]. The power converter acts as a positive supply if the switches Z_1 and Z_4 are closed. Similarly power converter acts as a negative supply if the switches Z_2 and Z_3 are closed. The converter does not supply any power while all the four switches are open.

The main purpose of the observer, is to mimic the switched resonant power converter and provide the controller with all the necessary parameters to effectively control the switching of the resonant circuit without the need of expensive sensory inputs from the resonant power converter. The Figure 2.2 gives an idea of how the whole system is integrated.

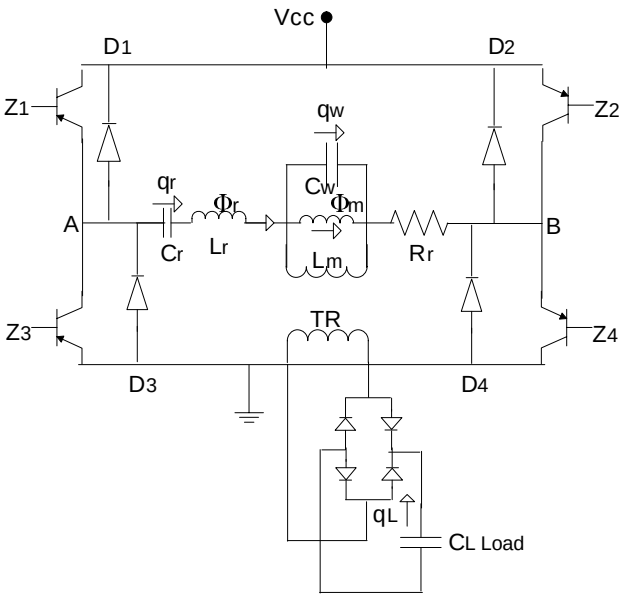


Figure 2.1: Switched Resonant Power Converter

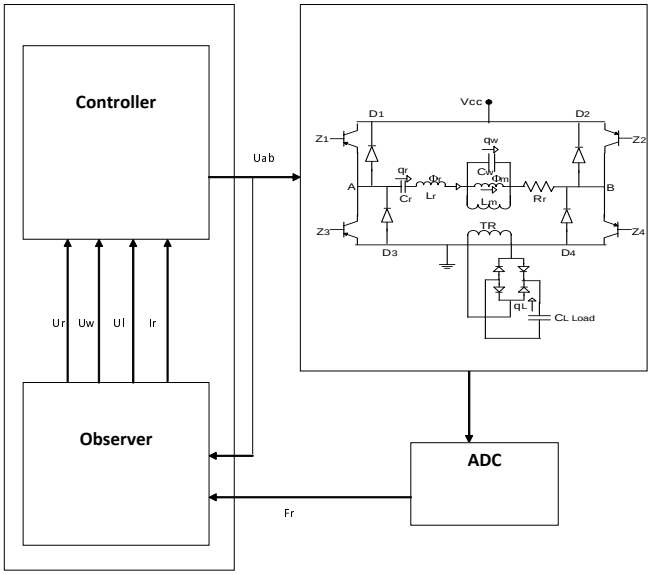


Figure 2.2: The System Overview

2.2 Related work

In [4] a Simulink model was designed and tested for 3rd and 4th order Hamiltonian observer. Both models were analyzed for different sampling times and Hamiltonian constant (K) values. There was recommended that a 4th order Hamiltonian model working at a 10 MHz sampling frequency was adequate for implementation on the FPGA platform.

In [1] some progress was made regarding the feasibility of using Matlab Simulink and Xilinx System Generator over hand written VHSIC (Very-High-Speed Integrated Circuit) Hardware Description Language (VHDL) code. The resource utilization of the hardware was compared when the controller was designed using the Simulink and Xilinx System Generator versus the hand written VHDL implementation already done by Sonny Bui at ALSTOM. It was realized that the Simulink and Xilinx System Generator used the resources more efficiently as compared to the hand written VHDL code.

Chapter 3

Methodology

3.1 Objective

The main objective is to implement the existing Fourth order Hamiltonian observer model on a currently available Xilinx Virtex-II prototype board fitted with a xc2v1000 fg256 chip.

Matlab Simulink and Xilinx System Generator is used as implementation tools for this development project. This report will serve as a guideline for all future implementation projects who will prefer to use Simulink and Xilinx System Generator over hand written VHDL code for the further development of new systems.

3.2 Software

There are several different options available regarding the development tools that can be selected and used. We are using Matlab as the main development tool. Simulink is a widely used tool for modeling, simulating and analyzing large and complex dynamic systems which can be easily integrated with Matlab. It provides a graphical overview of the whole system and helps in reducing the design time.

In order to convert the Simulink model to VHDL code, an add on, Xilinx System Generator is used which adds Xilinx block set to the Simulink library. The new model developed using Xilinx block set can then be used to generate the VHDL code. Project navigator is used to check the schematics of the model developed, map the design over the available chip and generate a programming file that can be used to actually program the FPGA.

After the programming file is transferred on to the FPGA, a software called the Xilinx ChipScope Pro is used which captures the internal signals generated deep within the implemented design. These signals can then be analyzed using the virtual Logic Analyzer through a programming interface, hence freeing the pins of the hardware used.

3.3 Hardware

The hardware available for implementation is a Xilinx VIRTEX-II FG256 PROTO BOARD. This development board is used to test the capabilities of the Virtex 2 series. The board is provided with an external power supply and a Parallel cable IV for JTAG configuration. The board has a 256 pin user configurable FPGA and a service FPGA which controls different functions on the board. The user FPGA Device Under Test (DUT) provides the testing ground for the developed code.

Chapter 4

Design and Implementation

In continuous form, the process model which is going to be implemented in this thesis work can be described by the following equation [3]:

$$\dot{x}(t) = (J(s) - R)Dx(t) + Bu_{AB}(t) \tag{4.1}$$

$$J(s) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 - s^2 I \\ 0 & 0 & 0 & sI \\ -1 & -(1 - s^2 I) & -sI & 0 \end{bmatrix}, \qquad R = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Rr \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

where, $s = \{-1, 0, 1\}$ i.e when the rectifier conducts in the negative direction, no conduction and the conduction in the positive direction.

The Hamiltonian observer in discrete form can be expressed as [5]:

$$\hat{x}(k+1) = F\hat{x}(k) + Gu_{AB}(k) + K[y(k) - C\hat{x}(k)] \tag{4.2}$$

where, F=Phi
G=Gama
K=Hamiltonian gain
$y(k)$=$Cx(k)$=Physical system model output

4.1 Physical system model

The physical system models the resonant power converter and is based on the equation 4.1. The logic unit defines the current state of the power converter that is either positive supply, negative supply or non conduction mode. The output of the controller serves as input to the physical system model. The physical system model outputs four different variables the resonance capacitance(u_r), winding capacitance(u_w), load capacitance(u_l) and the resonance current(i_r).

The controller output acts as input to the physical system model which is simulated using a square wave generator. The physical system model provides the input to the observer. Figure 4.1 shows the Simulink model of the physical system where Uab is a square wave input from the controller, Fr_Phy is fed to the observer which can be later used for error correction. The physical system model output contains the four actual output variables. The logic unit decides the current conduction state of the physical system model depending on the relation between u_l, u_w and i_r. $J(s) - R$ from equation 4.1 can be represented as A_0, A_1 and

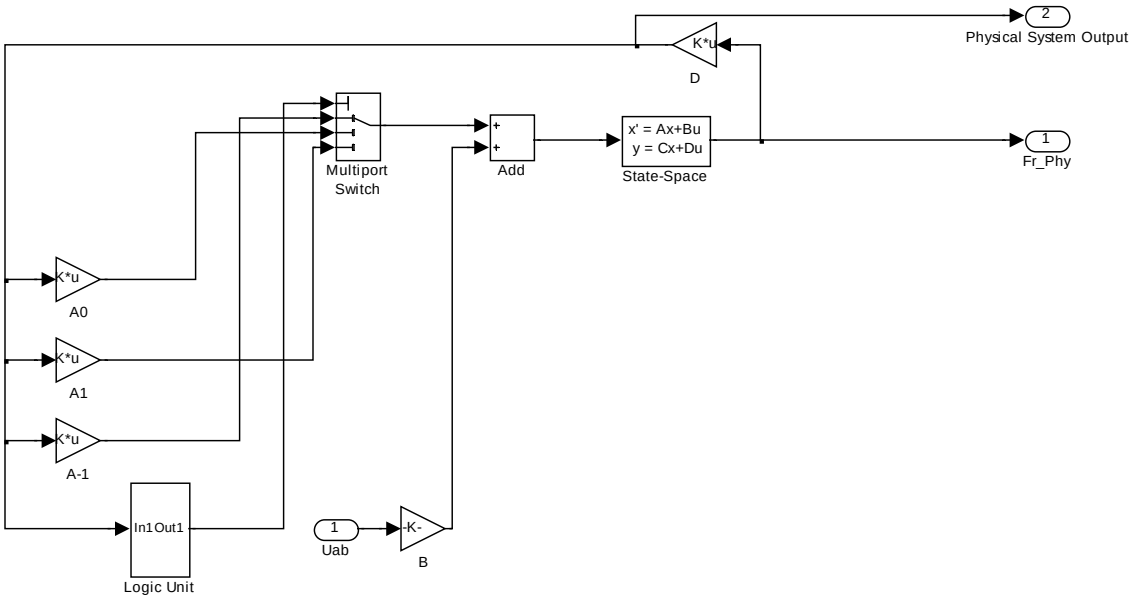


Figure 4.1: Simulink Physical System Model

A_{-1} for the each conduction state in the physical system model. D is the inverse of the M matrix which can be represented as $M = diag(Cr\ Cw\ Cl\ Lr)$.

4.2 Observer

Figure 4.2 shows the Simulink model of the fourth order observer. It also estimates the same four variables u_r, u_w ,u_l and i_r being simulated by the physical system model. The observer model is similar to the physical model and is based on the equation 4.2. However the physical model is in continuous time while the observer model operates in discrete time. The controller output and the physical system model output serves as the input to the observer. The job of the observer is to estimate all the states of the physical system model. The output from the physical system model and the observer model should be similar.

In figure 4.2 Uab is the input signal from the controller and Fr_Phy is one of the states of the physical system model which is later used for error correction. The logic unit is similar to the one in physical system model and is used to determine the current condition state of the observer. The observer output contains the four estimated variables similar to the physical system model outputs. D is same as in the physical system model. A_0, A_1 and A_{-1}in the physical system model can be represented in discrete time as $gama_0\ phi_0$, $gama_1\ phi_1$ and $gama_{-1}\ phi_{-1}$ in observer model.

Figure 4.3 shows the controller, physical system model and the observer connected together in Simulink environment. The controller provides input to both the physical system model and the observer, while the physical system model provides a measurement signal to the observer model.

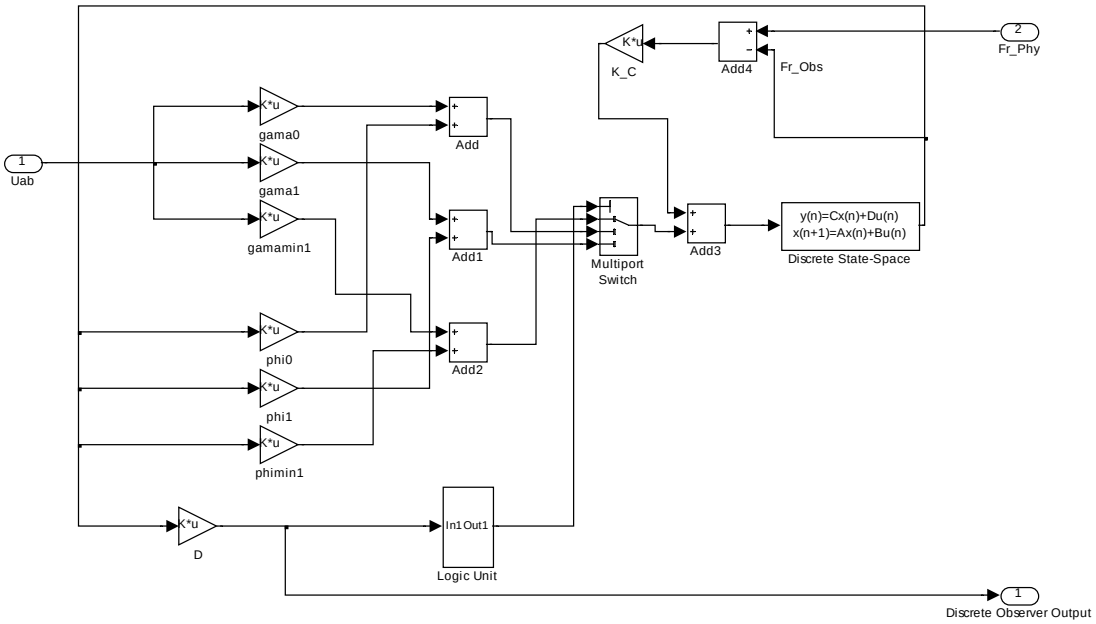


Figure 4.2: Simulink Observer Model

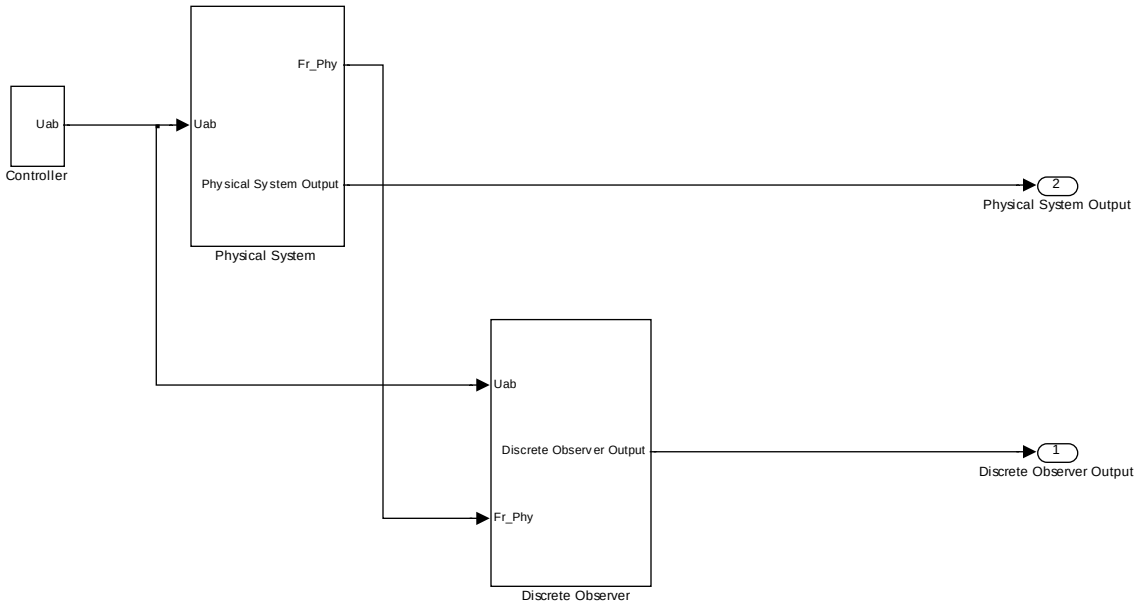


Figure 4.3: Controller, Physical System and Discrete Observer model

Figure 4.4 shows the simulation results for all the four variables being estimated by the discrete observer.

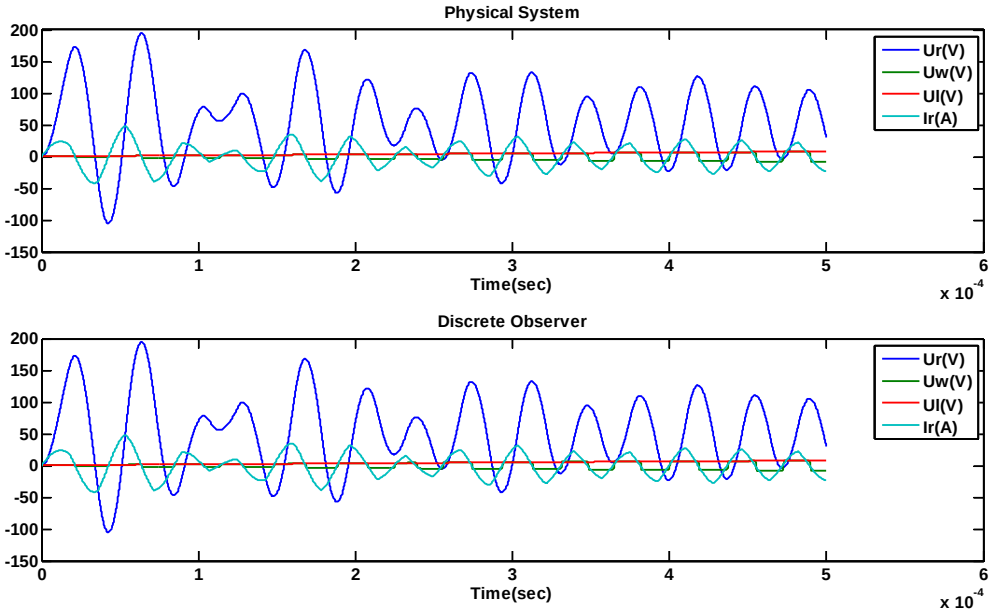


Figure 4.4: Physical System and Discrete Observer Model Output

4.3 4th Order Observer Implementation

In order to obtain VHDL code from the Simulink environment, we need an add on called the Xilinx System Generator. This add on supplements the current Simulink library with Xilinx block sets which can be easily converted into VHDL code.

4.3.1 Library Augmentation

Before converting the Simulink observer model into the Xilinx model, the current Xilinx library has to be augmented since the Xilinx block set does not support mathematical operations like matrix multiplication, vector multiplication, vector addition and discrete state space. The elements in the current Xilinx block set library are used to implement the above mentioned operations [1].

4.3.2 Xilinx Observer

The Simulink observer model is now converted into a Xilinx observer model by simply replacing the Simulink blocks with the equivalent Xilinx blocks. Figure 4.6 shows the equivalent Xilinx observer model. The observer model is provided with two inputs. One is the controller output that is the control voltage U_{ab}. The second input is provided by the physical system model which is the magnetic flow Fr_Phy through the inductor and is later used for error correction. Gateway In and Gateway Out are used to convert the Simulink data type into the Xilinx fixed point data type and vice verse [6].

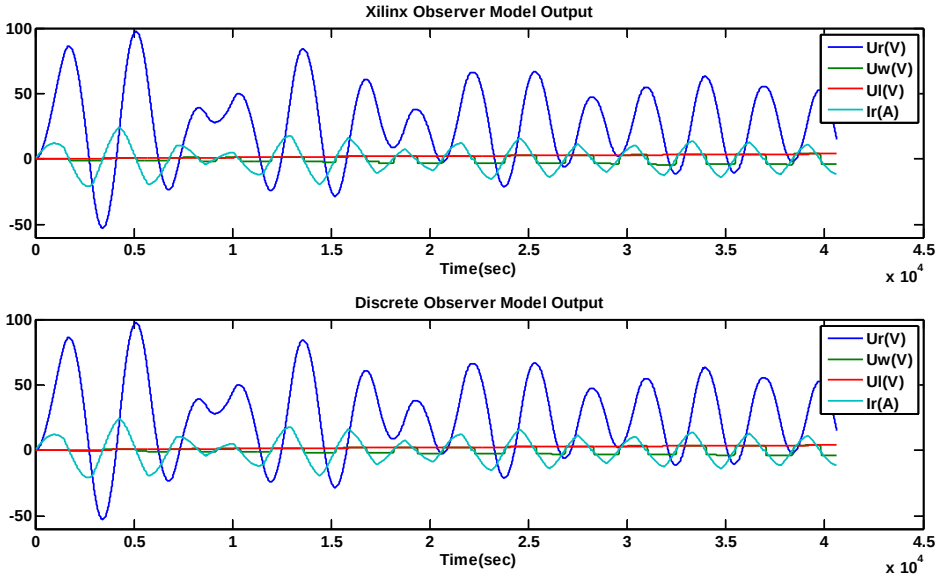


Figure 4.5: Discrete and Xilinx Observer Outputs for 200 Word length

4.3.3 Testing

In order to check the correct working and performance of the Xilinx observer, a simulation was run using the controller, physical system, discrete observer and the Xilinx observer models all connected together. The Xilinx observer was initialized with very large word lengths. Figure 4.5 shows the Discrete and Xilinx observer outputs while figure 4.7 shows the connections between the modules.

4.3.4 Word Length

The Xilinx observer when initialized at very large word lengths displayed good performance. However the drawback is the increase in the implementation size. One way to reduce the size is to reduce the word length. Since every part of the observer had different numeric values being processed, a generalized word length could not be allocated to every Xilinx block.

In order to calculate the best-fit for the word lengths, the simulation was run once for an appropriate amount of time [1] and data was saved in a workspace. Then for each node in the model the largest numeric values on the positive and negative side and the smallest numeric values on either side of the zero value were calculated. The largest and the smallest numeric values provided us with the number of non fractional and fractional bits required to represent the data correctly with in a reasonable range of accuracy through out the simulation.

The above mentioned method provided us with a rough estimate of the number of bits required. These initial word lengths are shown in figure A1.7. These word lengths could then be further reduced by using a for loop. During every iteration one bit was subtracted from all the blocks and the maximum error from the simulation was saved. This data when plotted provides us with a comparative graph between the reduction in bits from the starting value versus the increase in error in each parameter being estimated. Figure 4.8 shows the increase in error with the decrease in word length.

4.3.5 Results

The Xilinx observer works within a reasonable range of error after further decreasing the word lengths. However the model size is not small enough for the current chip set (xc2v1000 fg256). To try and further reduce the size, scaling is applied.

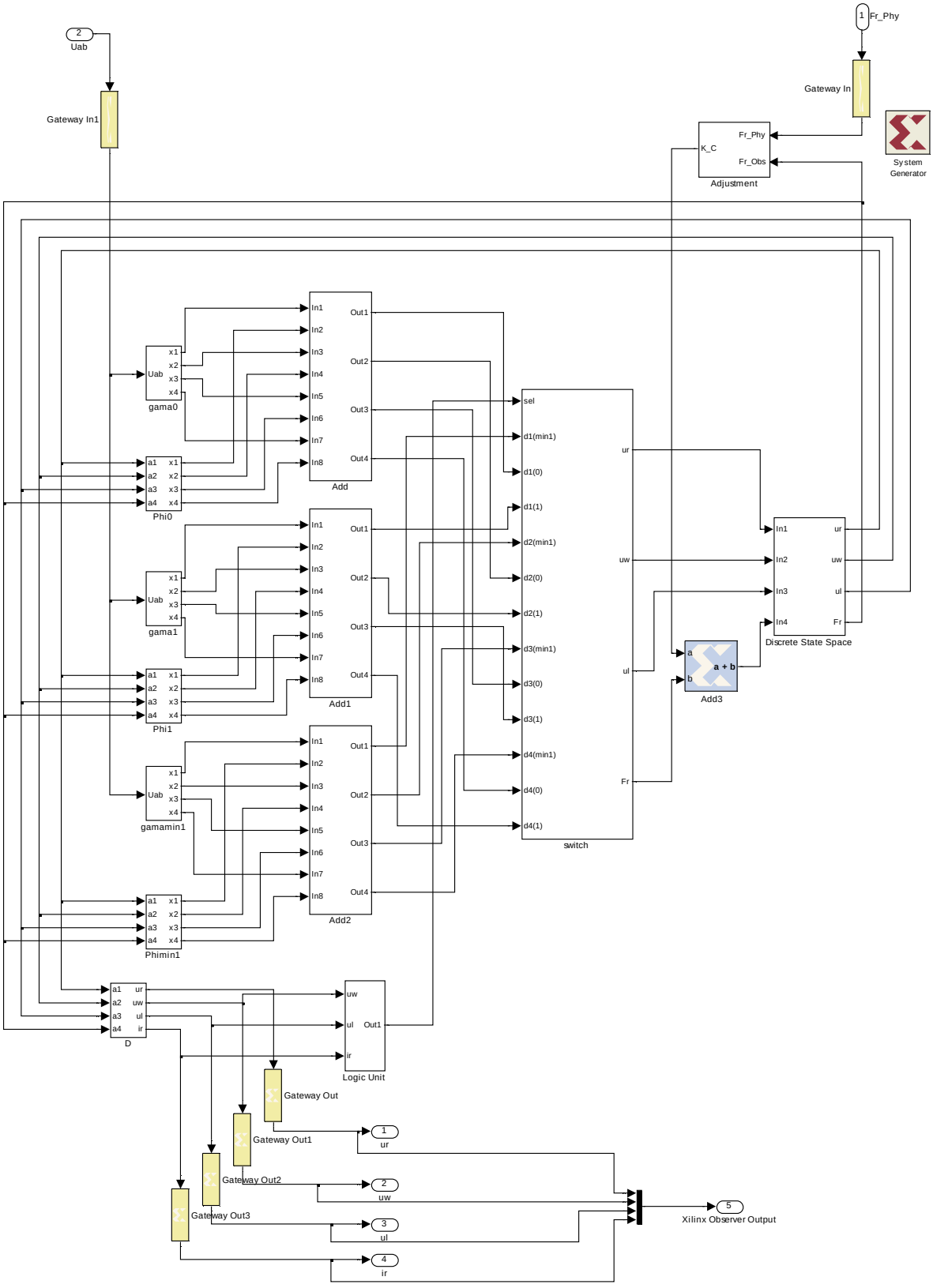


Figure 4.6: Xilinx Observer Model

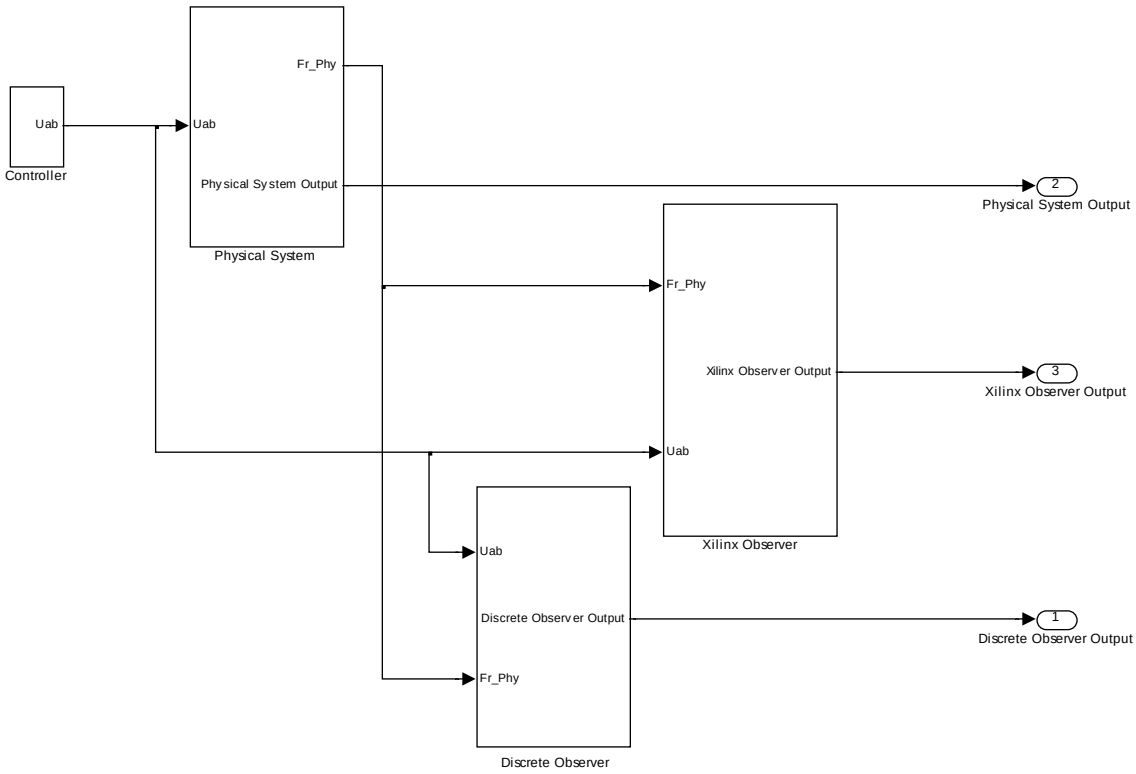


Figure 4.7: Controller, Physical system, Discrete and Xilinx observer models

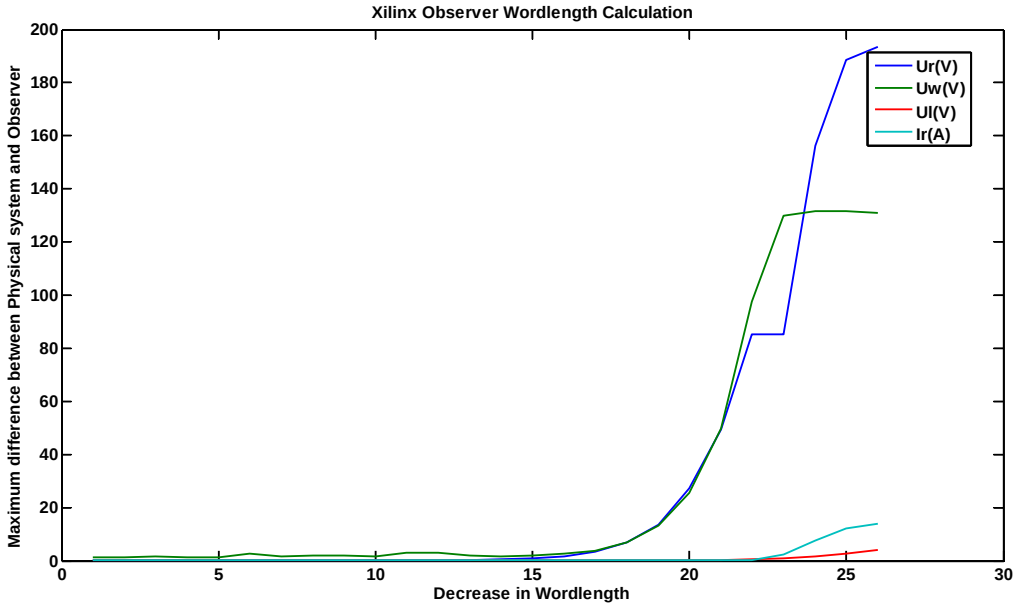


Figure 4.8: Xilinx Observer Word length vs Error

4.3.6 Scaling

The internal states of the observer are very small fractional values. These small fractions require a large number of fractional bits for representation. One way to reduce the word length requirement is to scale these small fractions to a larger number so that they require less fractional bits for representation.

While scaling, all the inputs to the observer have to be scaled up and all the outputs from the observer have to be scaled down as well. A scaling matrix is calculated to apply scaling to the internal observer states.

To calculate the scaling matrix, a simulation is run and all the states are saved in a workspace. The maximum value from all the states is taken. The scaling matrix can be represented in general terms as $T_{scaling} = diag(2^{x_{U_r}} 2^{x_{U_w}} 2^{x_{U_l}} 2^{x_{I_r}})$ where x_{U_r}, x_{U_w}, x_{U_l} and x_{I_r} can be calculated using the mathematical calculations described below. The diagonal is chosen to be a power of 2 as it can alternatively be implemented using arithmetic shift which can help reduce the resource utilization. The following mathematical calculations are only for x_{U_r}, but the others can also be calculated in the same way. U_r is the maximum value of the resonance voltage throughout the whole simulation. After scaling, the values need to be close to 1 [7]. Hence

$U_r * 2^{x_{U_r}} = 1$
$log(U_r) + x_{U_r} * log(2) = log(1)$
$x_{U_r} * log(2) = -log(U_r)$
$x_{U_r} = -log(U_r)/log(2)$

Table 4.3.6 shows the matlab code for calculating the scaling matrix.

```
maxX=max(simout); xlog2=1/log(2);
xa=(-1*log(maxX(1,1)))*xlog2;xb=(-1*log(maxX(1,2)))*xlog2;
xc=(-1*log(maxX(1,3)))*xlog2;xd=(-1*log(maxX(1,4)))*xlog2;
xdiag=[2^xa 2^xb 2^xc 2^xd];
Tscaling=floor(diag(xdiag))
```

Table 4.1: Matlab code for $T_{scaling}$ Matrix

After calculations, the scaling matrix $T_{scaling}$ can be applied to the observer in the following way.

The state of the unscaled physical system model can be represented as [5]:

$$x(k+1) = Fx(k) + Gu(k)$$

$$y(k) = Cx(k) + Du(k) \tag{4.3}$$

The state of the scaled physical system model can be represented as [7]:

$x_{scaled}(k+1) = TFT^{-1}x(k) + TGu(k)$
$y_{scaled}(k) = CT^{-1}x(k) + Du(k)$

The equivalent scaled state observer can then be represented as [5]:

$\hat{x}_{scaled}(k+1) = T\hat{x}(k+1) = TFT^{-1}\hat{x}(k) + TGu(k) + TK[y(k) - \hat{y}_{scaled}(k)]$
$\hat{y}_{scaled}(k) = T\hat{y}(k) = CT^{-1}\hat{x}(k) + Du(k)$

The scaled equation can be obtained as follows. Here $y(k)$ is from equation 4.3 since the output from the physical system model is unscaled

$\hat{x}_{scaled}(k+1) = T\hat{x}(k+1) = TFT^{-1}\hat{x}(k) + TGu(k) + TK[y(k) - \hat{y}_{scaled}(k)]$
$\hat{x}_{scaled}(k+1) = T\hat{x}(k+1) = TFT^{-1}\hat{x}(k) + TGu(k) + TK[Cx(k) + Du(k) - CT^{-1}\hat{x}(k) - Du(k)]$
$\hat{x}_{scaled}(k+1) = T\hat{x}(k+1) = TFT^{-1}\hat{x}(k) + TGu(k) + TK[Cx(k) - CT^{-1}\hat{x}(k)]$
$\hat{x}_{scaled}(k+1) = T\hat{x}(k+1) = TFT^{-1}\hat{x}(k) + TGu(k) + TKCT^{-1}[Tx(k) - \hat{x}(k)]$

The above equation shows how each parameter should be scaled in the observer.

4.3.7 Scaling Results

Scaling provided larger internal states for the observer which required a smaller number of fractional bits as compared to the unscaled version. The Figure 4.9 shows the comparison between the states before and after the scaling.

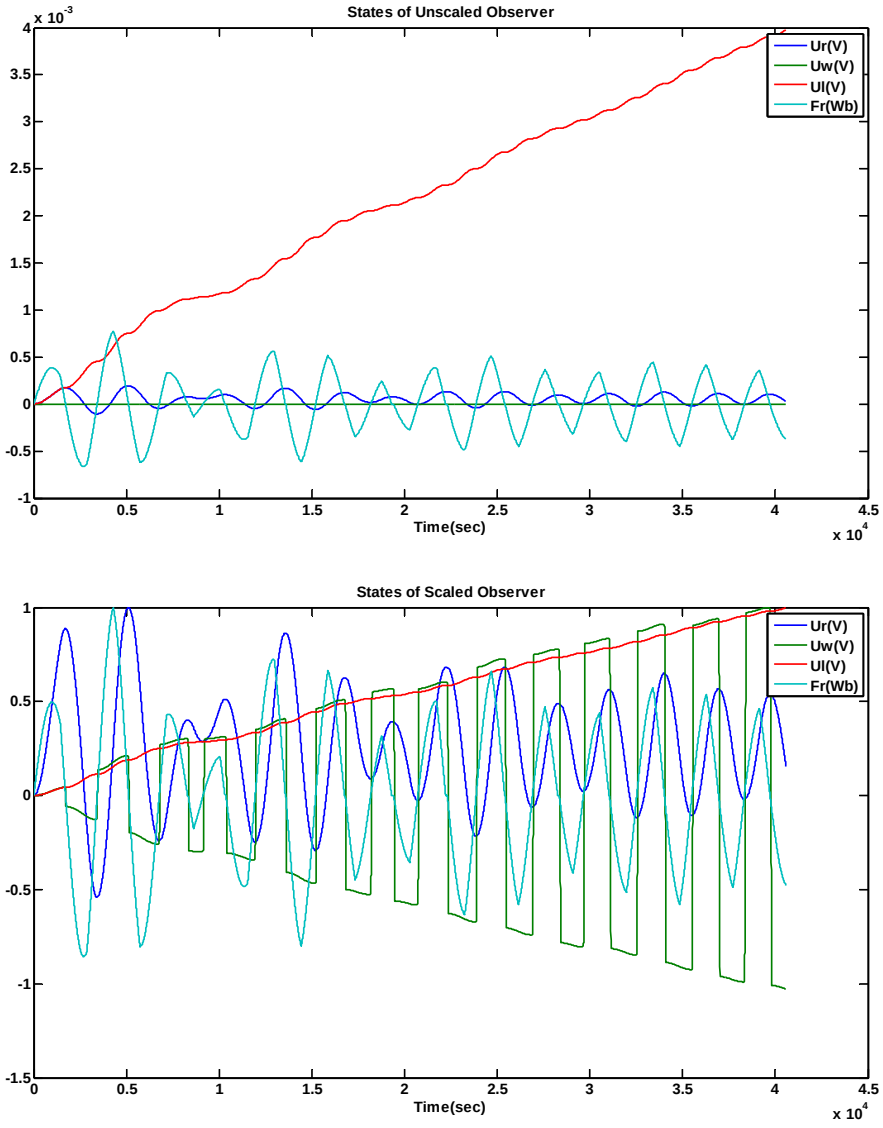


Figure 4.9: Unscaled and Scaled Observer States

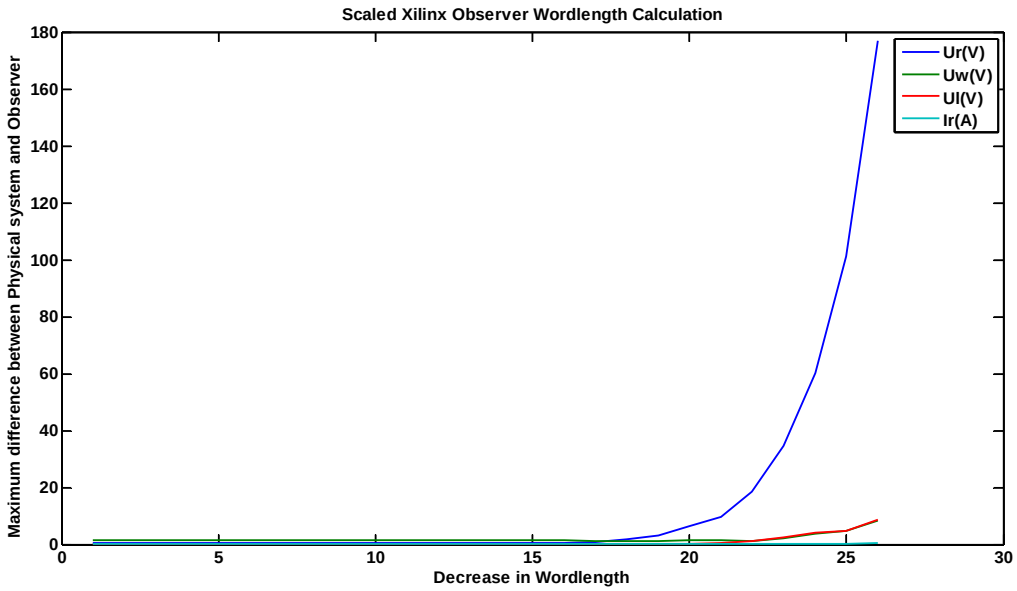


Figure 4.10: Scaled Xilinx Observer Word length vs Error

After scaling, new word lengths for the scaled observer were calculated by the procedure already described in subsection 4.3.4 and are shown in figure A1.8. Figure 4.10 shows the increase in error as the word length is decreased by one bit at a time from the starting scaled word lengths.

To have a comparison between the word lengths before and after scaling, let us consider the word length for one observer parameter Gama. Before scaling the appropriate word length for Gama output was calculated as 61_60 where 61 are the total number of bits out of which 60 are fractional bits and 1 in a non fractional bit. We further reduced it by 19 bits as explained in 4.3.4 to achieve a reduced word length of total 42 bits. After scaling, the starting word length for Gama output was 41_40. After further reducing 21 bits, the new word length was 20_19. After appropriate mathematical calculations, the reduction in word length was calculated to be 52.38 percent.

It was also realized that in general, the error was more dependent on the word lengths of gama, phi and D as compared to Fr_Phy, Tscaling and Adjustment parameters in the scaled Xilinx observer model shown in figure A1.2. Hence Fr_Phy, Tscaling and Adjustment were further reduced by 39 bits after the initial 21 bit reduction, allowing us to reduce a total of 60 bits from these parameters.

With scaling, we were able to reduce the word lengths to quite an extent and achieved quite reasonable results. The reduction in hardware requirements due to scaling was confirmed by synthesizing and mapping the generated VHDL code using Xilinx Project Navigator. However the observer size still exceeded the capabilities of the current chip set (xc2v1000 fg256).

4.3.8 Data Analysis

In order to try and reduce the observer size requirements, some comparative study was done. Different sampling frequencies and initial conditions were applied to the observer, results were compared and recommendations were made.

Sampling Frequencies

All the previous calculations were done at a sampling frequency of 80MHz which is 12.5 ns of sampling time. In [4] it was recommended that 10MHz is the optimum frequency for any future observer implementation on FPGA platform. Our current platform supports a minimum frequency of about 25MHz i.e 40ns of sampling time. Here we will compare the results between using a 25MHz and 80MHz sampling frequency. Figure 4.11 and 4.12 show a comparison between the two frequencies for a Xilinx observer with all the same parameters. It seems from the results that for the same variables error is four times less using 25MHz sampling frequency as compared to 80MHz.

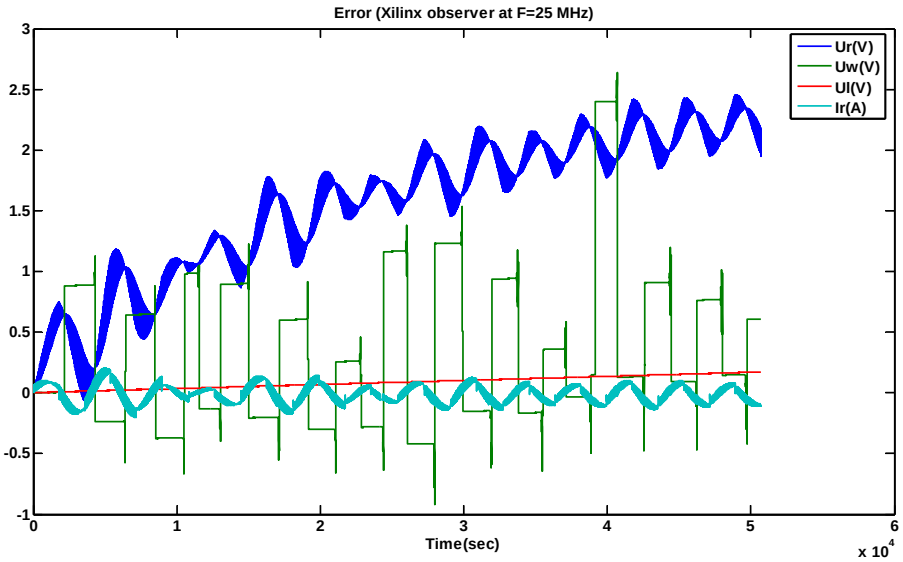


Figure 4.11: 25 MHz Sampling Frequency

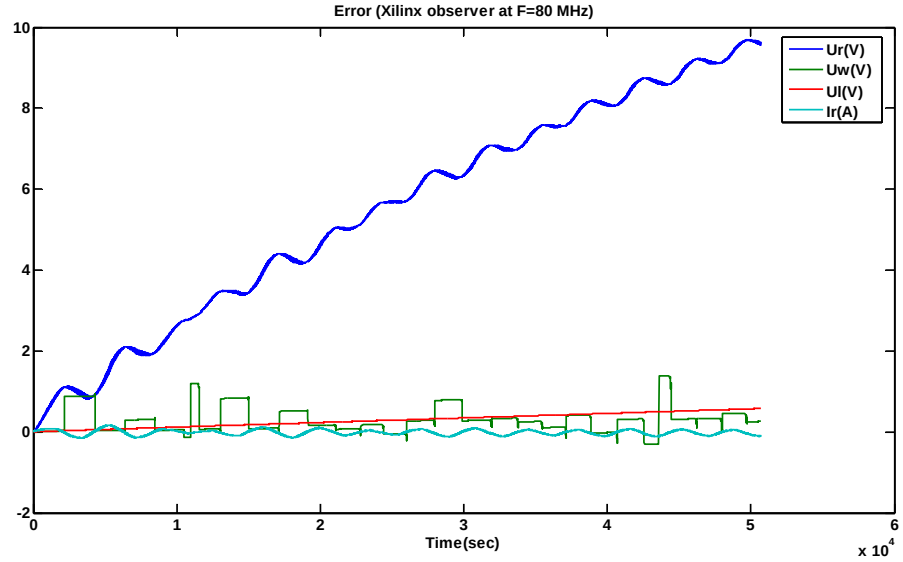


Figure 4.12: 80 MHz Sampling Frequency

Figure 4.13 and 4.14 shows the error for large word length in Xilinx observer. A comparison between the figures 4.10, 4.11, 4.12 and 4.13 shows the limitation of the Xilinx observer due to word length restriction. Increasing the word length greatly decreases the error between the physical system model's actual and the observer's estimated variable values. A way to increase the performance is to clip u_w within the model so that it does not exceed a certain level since it is not possible due to certain physical reasons.

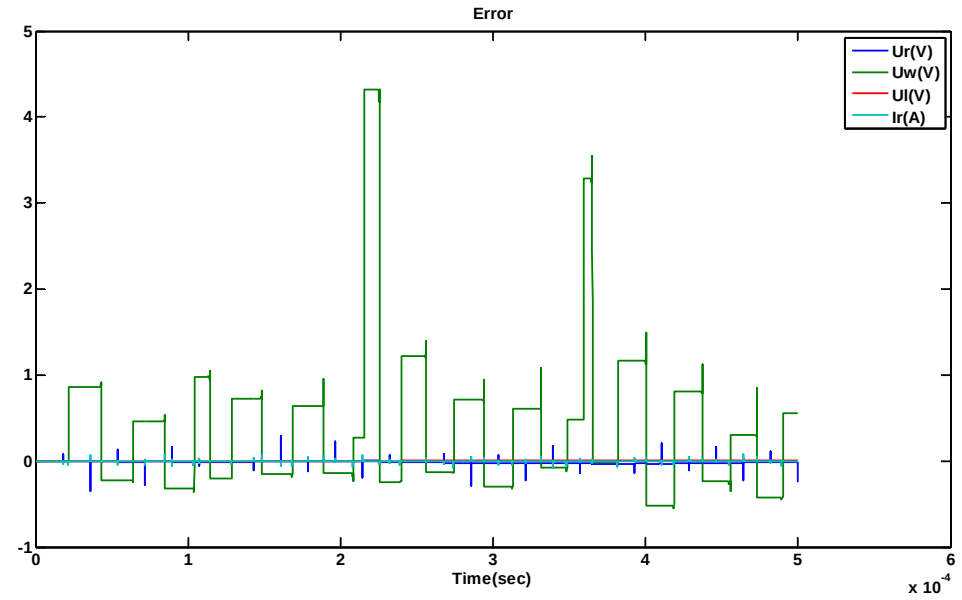


Figure 4.13: 25 MHz Sampling Frequency with 200 Word length

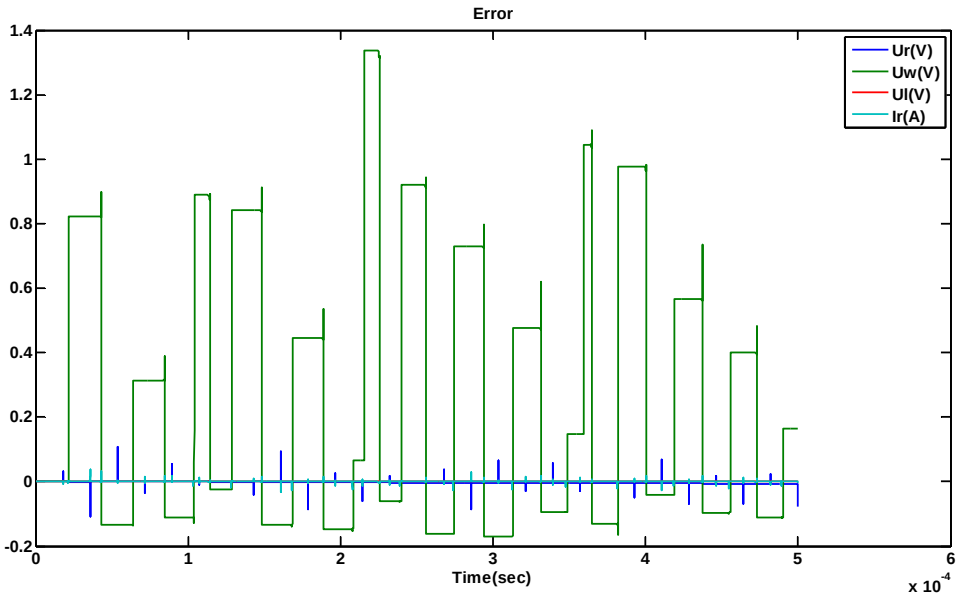


Figure 4.14: 80 MHz Sampling Frequency with 200 Word length

Initial conditions
We also experimented with different initial conditions to try and improve the performance of the observer while keeping the word lengths to a minimum. Following initial conditions which are similar to [4]:

Physical system: $u_r = 0, u_w = 0, u_l = 0, i_r = 0$
Observer: $u_r = 0, u_w = 0, u_l = 0, i_r = 0$

Physical system: $u_r = 50, u_w = 0, u_l = 300, i_r = 20$
Observer: $u_r = 50, u_w = 0, u_l = 300, i_r = 20$

Physical system: $u_r = 0, u_w = 0, u_l = 300, i_r = 0$
Observer: $u_r = 0, u_w = 0, u_l = 300, i_r = 0$

The comparison and analysis of different simulation results from different initialization conditions lead to the conclusion that the error was significantly less when both the physical system model and the observer were initialized with the same initial conditions. The output error was quite large when an error was introduced between the initial conditions of the physical system model and the observer. Also the word length had to be increased within the Xilinx observer when the observer was initialized with large values.

Recommendation
After some analysis of the above results, it was recommended that a lower sampling frequency of 25 MHz and zero initial conditions for both the observer and the physical system model should be used.

4.3.9 Result

By using a sampling time of 40 ns we were able to further reduce the large word lengths in the error correction part of the observer while keeping the error to within the acceptable range. However the observer size was still not small enough for the current chip set. Figure 4.15 shows the current observer exceeding the available resources.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	97	10,240	1%	
Number of 4 input LUTs	11,643	10,240	113%	OVERMAPPED
Logic Distribution				
Number of occupied Slices	6,212	5,120	121%	OVERMAPPED
Number of Slices containing only related logic	3,997	6,212	64%	
Number of Slices containing unrelated logic	2,215	6,212	35%	
Total Number 4 input LUTs	12,244	10,240	119%	OVERMAPPED
Number used as logic	11,643			
Number used as a route-thru	601			
Number of bonded IOBs	133	172	77%	
Number of GCLKs	1	16	6%	
Number of RPM macros	64			
Total equivalent gate count for design	111,596			
Additional JTAG gate count for IOBs	6,384			

Figure 4.15: 25 MHz 4th Order Observer For xc2v1000-6fg256

4.3.10 Hardware Upgrade

In order to implement the current observer model we recommend a hardware upgrade for the board. If the current xc2v1000-6fg256 is replaced with xc2v1500-6fg676, observer implementation is possible. Figure 4.16 shows the resource utilization for xc2v1500-6fg676. This upgrade will also support the ChipScope blockset which can measure two signals upto 4096 samples each.

SCALED_XILINX_OBSERVER_25MHZ_CW Project Status			
Project File:	scaled_xilinx_observer_25mhz_cw.ise	**Current State:**	Programming File Generated
Module Name:	scaled_xilinx_observer_25mhz_cw	**• Errors:**	No Errors
Target Device:	xc2v1500-6fg676	**• Warnings:**	858 Warnings (0 new)
Product Version:	ISE 8.2.02i	**• Updated:**	fr 20. aug 22:12:39 2010

SCALED_XILINX_OBSERVER_25MHZ_CW Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	97	15,360	1%	
Number of 4 input LUTs	11,643	15,360	75%	
Logic Distribution				
Number of occupied Slices	7,678	7,680	99%	
Number of Slices containing only related logic	6,929	7,678	90%	
Number of Slices containing unrelated logic	749	7,678	9%	
Total Number 4 input LUTs	12,244	15,360	79%	
Number used as logic	11,643			
Number used as a route-thru	601			
Number of bonded IOBs	133	392	33%	
Number of GCLKs	1	16	6%	
Number of RPM macros	64			
Total equivalent gate count for design	111,596			
Additional JTAG gate count for IOBs	6,384			

Performance Summary			
Final Timing Score:	0	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	fr 20. aug 22:05:24 2010	0	855 Warnings (0 new)	0
Translation Report	Current	fr 20. aug 22:07:04 2010	0	1 Warning (0 new)	0
Map Report	Current	fr 20. aug 22:08:17 2010	0	2 Warnings (0 new)	3 Infos (0 new)
Place and Route Report	Current	fr 20. aug 22:11:42 2010	0	0	0
Static Timing Report	Current	fr 20. aug 22:12:01 2010	0	0	1 Info (0 new)
Bitgen Report	Current	fr 20. aug 22:12:39 2010	0	0	0

Secondary Reports		
Report Name	**Status**	**Generated**
Xplorer Report		

Figure 4.16: 25 MHz 4th Order Observer For xc2v1500-6fg676

4.4 3rd Order Observer Implementation

4.4.1 Xilinx Implementation

Since the current hardware available was not sufficient enough for the implementation of a 4th order observer, we decided to reduce the model by one variable (u_w) to a 3rd order model estimating only u_r, u_l and i_r.

According to [4], equation 4.1 can be modified for a 3rd order model in the following way,

$$J(s) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & s \\ -1 & -s & 0 \end{bmatrix} \qquad \text{where, } s = \{-1, 1\}, \qquad R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & R_r \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The logic unit for the 3rd order observer can be implemented using the following equations:

$\Omega_1 : s = 1$:Positive conduction $i_r > 0$
Ω_{-1} :$s = -1$:Negative conduction $i_r \leq 0$

The Xilinx observer after the appropriate modifications is shown in the figure 4.17. This Xilinx observer is connected to the 4th order physical system model and operates at a sampling frequency of 25MHz with zero initial conditions.

Figure 4.18 shows the three estimated outputs of the observer while figure 4.19 shows the error between the three corresponding states of the Simulink physical system model and the 3rd order Xilinx observer model.

This 3rd order Xilinx observer was successfully synthesized, mapped, route & placed using ISE Project Navigator. A programming file was generated and the FPGA was successfully programmed using iMPACT.

4.4.2 Verification Using ChipScope

In order to check the performance of the FPGA, ChipScope Pro Analyzer was used to verify the internal signals of the FPGA implementation using a virtual scope. For this purpose the implementation model was modified. In order to provide input to the observer without external sources, a square wave generator was developed using the Xilinx System Generator block set. This square wave generator was then connected to the observer to act as an input from the controller and the physical system model. A ChipScope block was connected to retrieve the desired data signals. The modified model is shown in figure A1.4. A bit file was generated for this new model and the FPGA was configured.

Since the input to the observer model was arbitrary, therefore the output waveform would be different from the one obtained during the simulation process. However it should be similar to the output obtained during the on chip verification process. ChipScope provided quite similar results between the simulation and the actual FPGA calculations. Results from the Simulink simulation and ChipScope are shown in Figure 4.20 and 4.21 and are discussed later in detail.

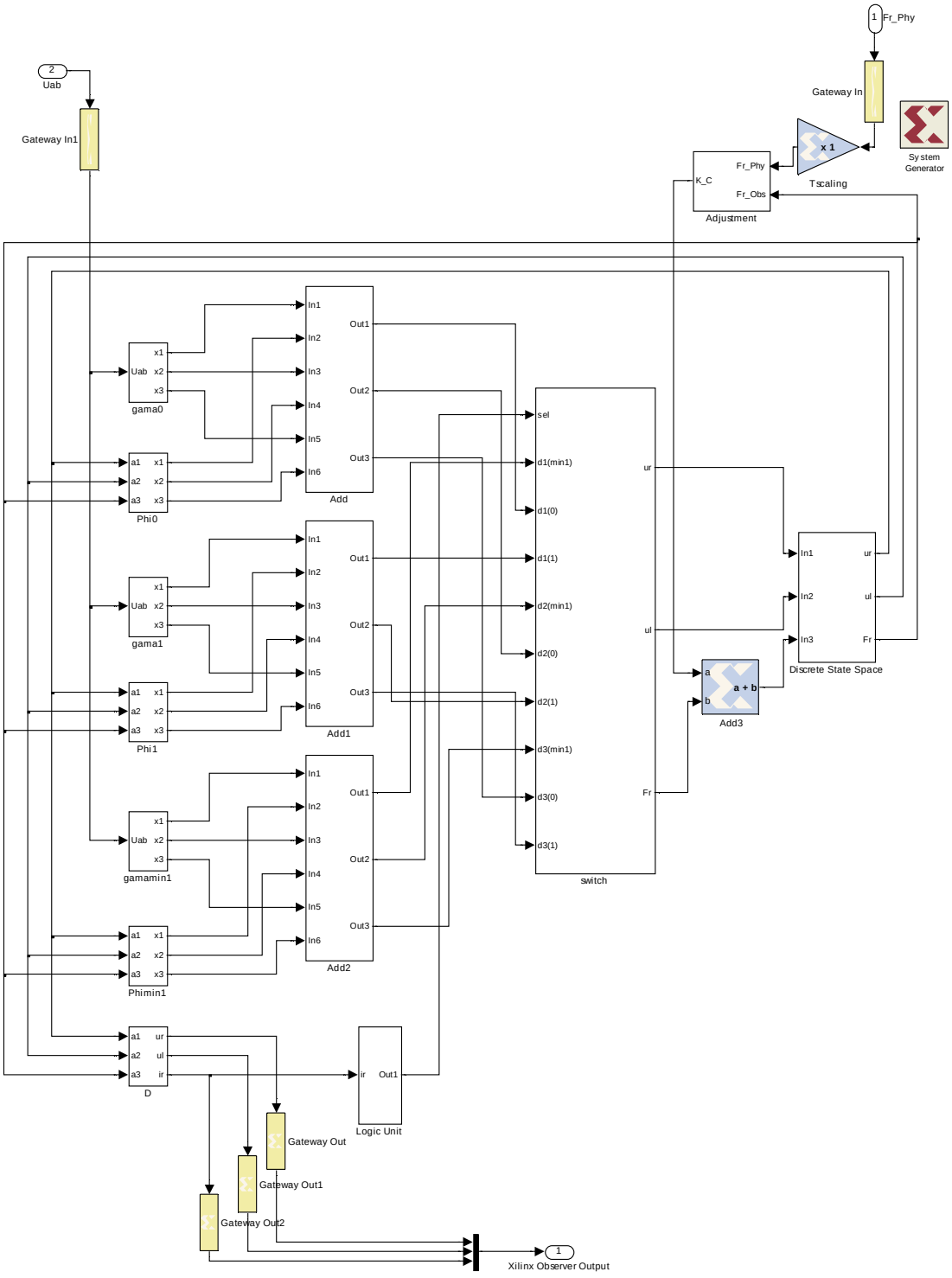


Figure 4.17: Xilinx 3rd Order Observer Model

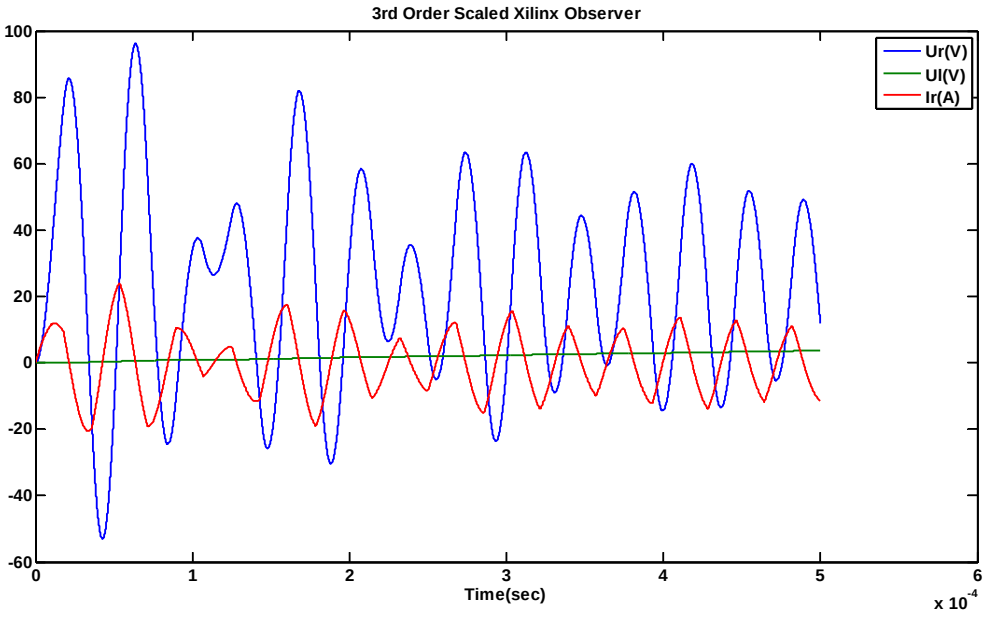


Figure 4.18: Output of 3rd Order Xilinx Observer

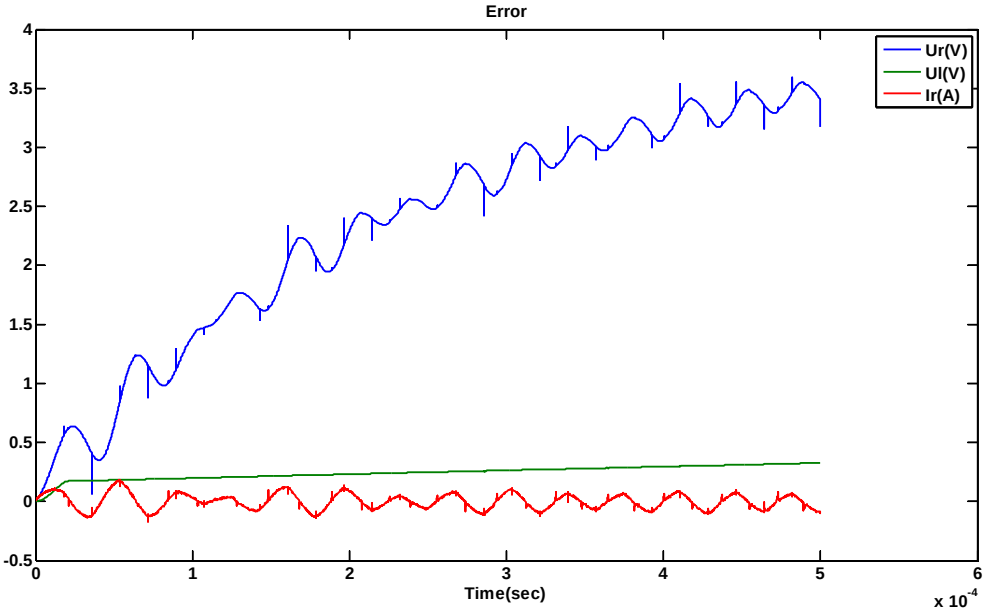


Figure 4.19: Error for 3rd Order Observer

Figure 4.20 shows the results from the Simulink simulation while figure 4.21 shows the results obtained by ChipScope. Due to the hardware limitations ChipScope was set to measure only i_r and u_r which are the two prominent variables in the simulation. Since the FPGA is constantly performing the calculations, the system stabilizes by the time we trigger to get the waveforms. Hence the plots from ChipScope shows i_r and u_r in the stabilized state. A comparison between the figures show similarity between the simulated and the actual results obtained.

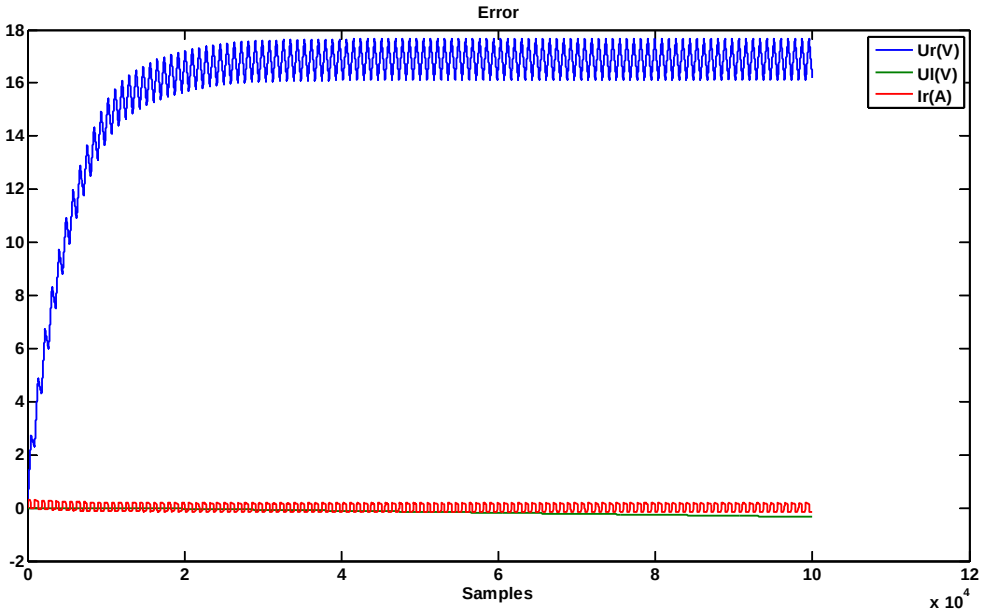


Figure 4.20: Simulink Error for a Square Wave Input Signal

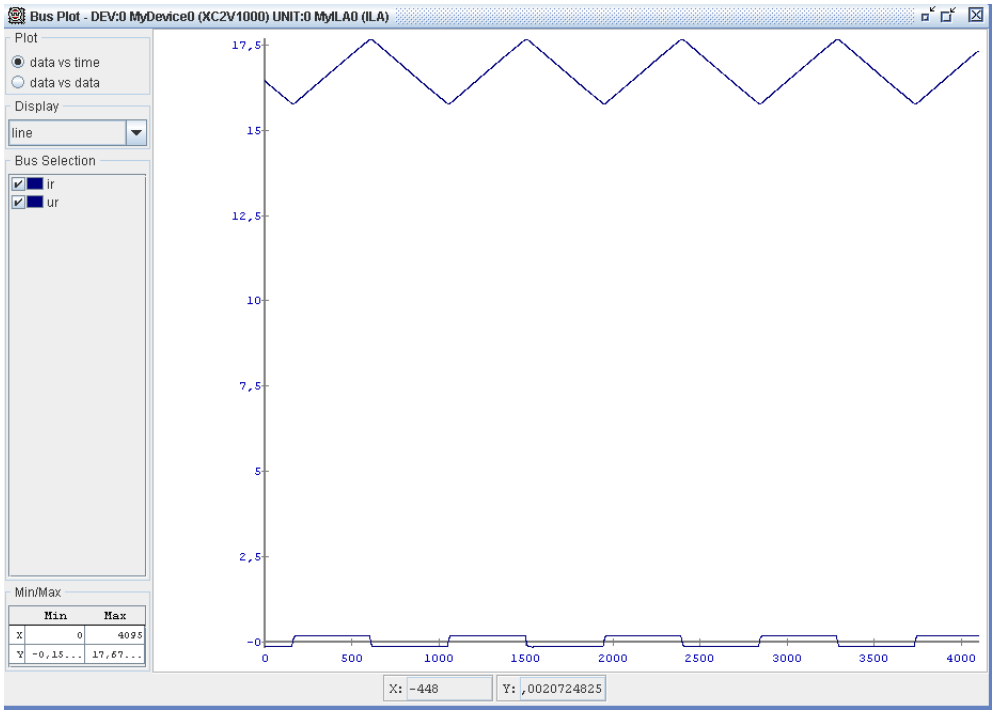


Figure 4.21: ChipScope Output for ir and ur

Figure 4.22 and 4.23 shows the ChipScope output for ir and ur separately. Both the signals bear reasonable resemblance to the Simulink results. Here since only one signal was measured at a time, therefore the number of samples for each signal to be saved were increased showing more details in the figure.

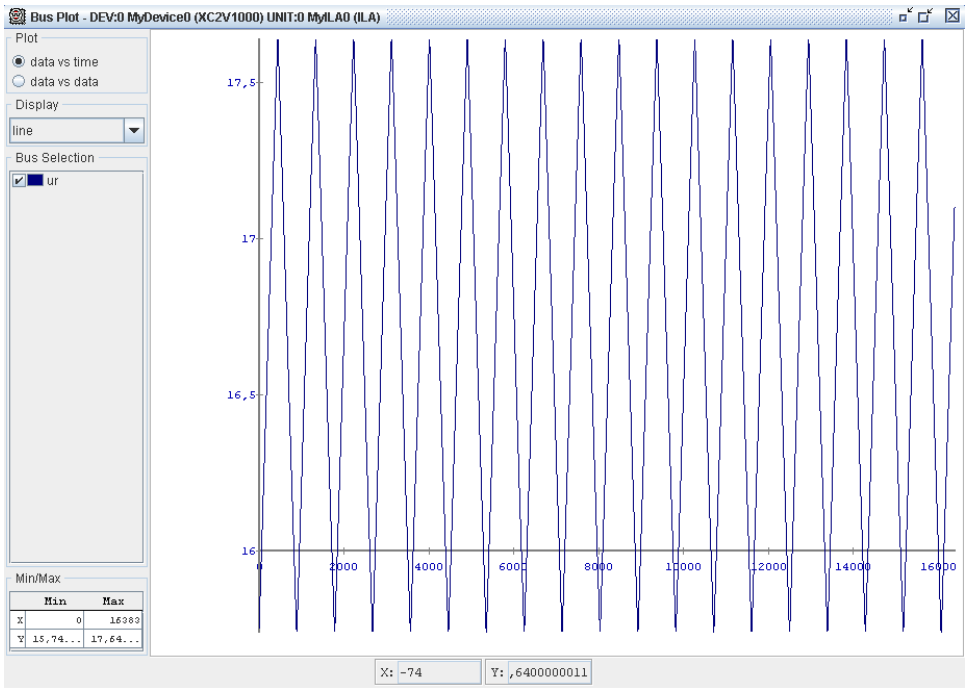


Figure 4.22: ChipScope Output for ur

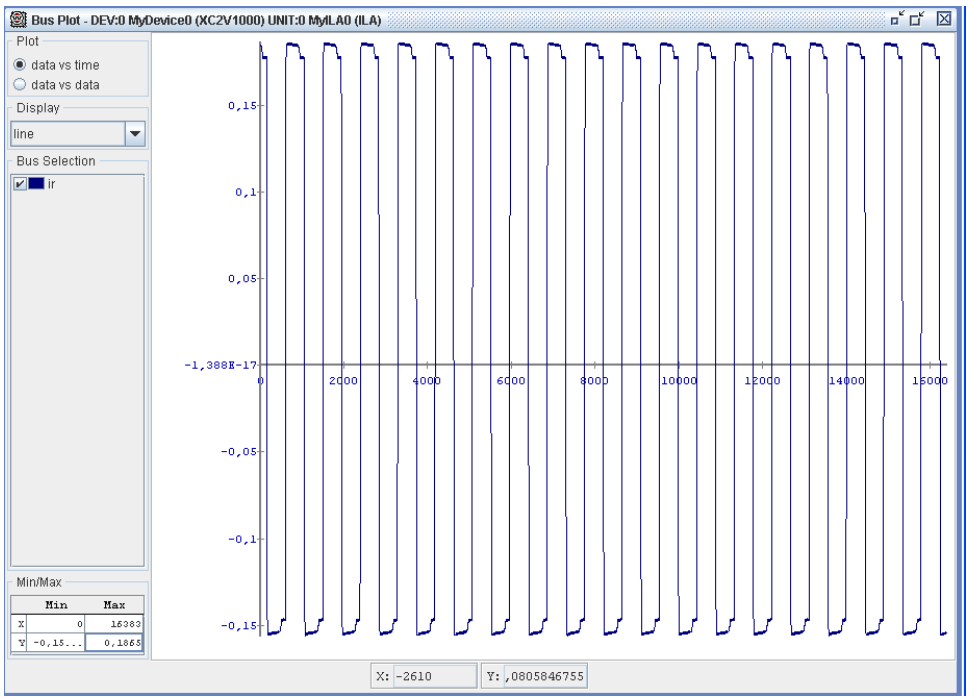


Figure 4.23: ChipScope Output for ir

Chapter 5

Simulink Model to VHDL Code Conversion

This chapter provides a guideline on how to use Matlab Simulink and Xilinx System Generator as implementation tools for development of any future system. We will start from the model designing in Simulink and cover all the steps required for VHDL implementation in FPGA platform.

5.1 Simulink

Simulink is a widely used simulating tool with a graphical overview that can be easily integrated with Matlab. The graphical nature of the tool greatly reduces the time for designing and analyzing of large dynamic systems. The physical system and observer models are first designed and tested in the Simulink environment using the default block set from the Simulink library. The Matlab M-file provides all the necessary parameters, values and initial conditions for the Simulink models.

5.2 Xilinx System Generator

Xilinx System Generator is an add-on to the Simulink. It adds Xilinx block set to the current Simulink library. After the models are analyzed through Simulink, an equivalent Xilinx model is implemented using the Xilinx block sets. The performance limitation in the Xilinx model is due to the fixed word length.

After all the performance parameters are set including the System Generator Token containing the FPGA clock period and Simulink system period, Xilinx model is then run once with all the necessary inputs connected to the appropriate Simulink models. This provides the System Generator with all the necessary parameters required for the implementation. For hardware implementation files, double click the System Generator Token which looks similar to Figure 5.1 and select Hardware Description Language (HDL) netlist as the compilation tool. It allows for the Register Transfer Level (RTL) synthesis and place and route to be performed using a tool such as Project Navigator. Select the target part which in our case is Virtex2 xc2v1000-6fg256. Select the hardware description language as VHDL and enter the appropriate clock pin location according to the board specifications which in our case is $R9$. Click the Generate button to generate the HDL Netlist [6].

5.3 Project Navigator

After the above steps a netlist folder is created in the parent directory. This folder contains a file with the same name as the Xilinx model in the Simulink .mdl file with a .ise extension. Double clicking this

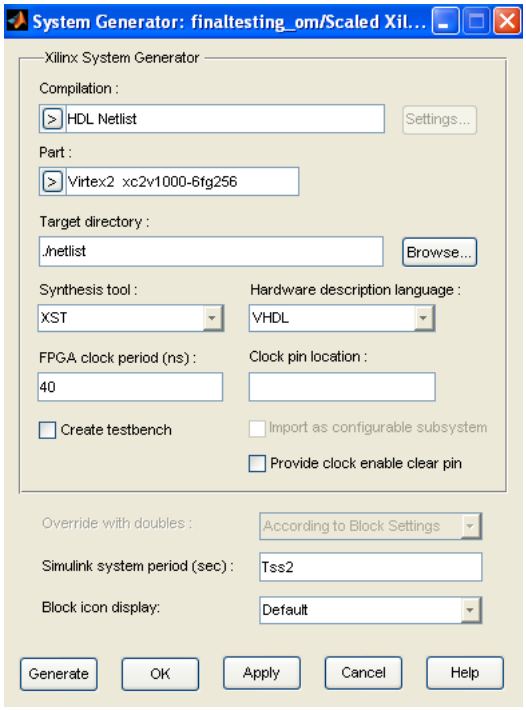


Figure 5.1: System Generator Token

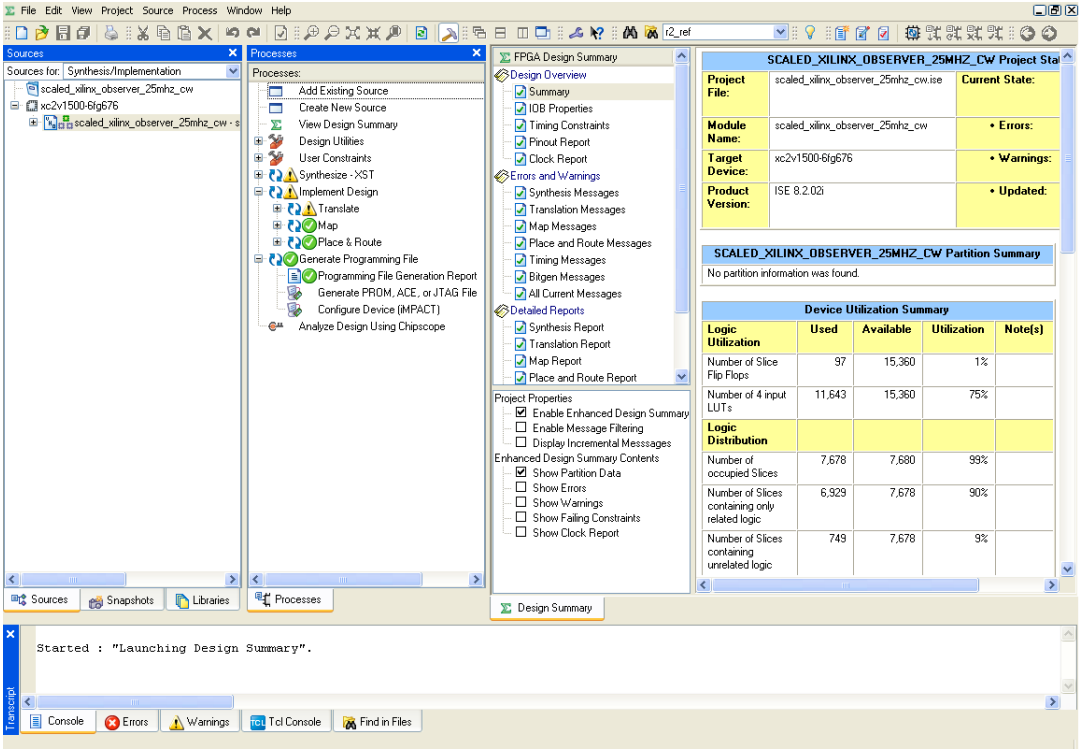


Figure 5.2: ISE Project Navigator

file opens the ISE Project Navigator which is similar to Figure 5.2. Sources show the hardware selected for synthesis and implementation. Processes contains Synthesis - XST, Implement Design and Generate Programming File. Right clicking Synthesis-XST and selecting the run option turns the circuit behavior, in our case RTL, into an implementable design in terms of logic. After successful synthesis schematics can be viewed. After synthesis, the implementation which consists of Translation, Mapping and Place & Route, converts the implementable design into a file that can be downloaded onto the chosen device. Generate Programming file produces a bit steam which can be used to configure the Xilinx device. To configure the device, run Configure Device (iMPACT). Select Configure Device using Boundary Scan (JTAG) in the new window opened and click Finish. Select the generated bit file which will be used to configure the FPGA. After the bit file is selected, right click on the device icon and select program. After successful configuration of the device a message will appear, Program succeeded. Also on the board, the led will change from INT to DONE[8][9].

5.4 ChipScope Pro

In ISE Project Navigator run the Analyze Design using ChipScope. This will open a new window for the ChipScope Pro. Open the JTAG chain. Device would be detected automatically. Select the appropriate device. Now in the top of the window select the Device pull down menu and select the bit file for the device. A .cdc file can be imported to assign the appropriate signal names that were set during the ChipScope insertion phase. Different trigger conditions could be set to capture the waveforms and data buses. Different dataports can be combined into a databus for signal analysis. Figure 5.3 shows the graphical interface for ChipScope Pro.

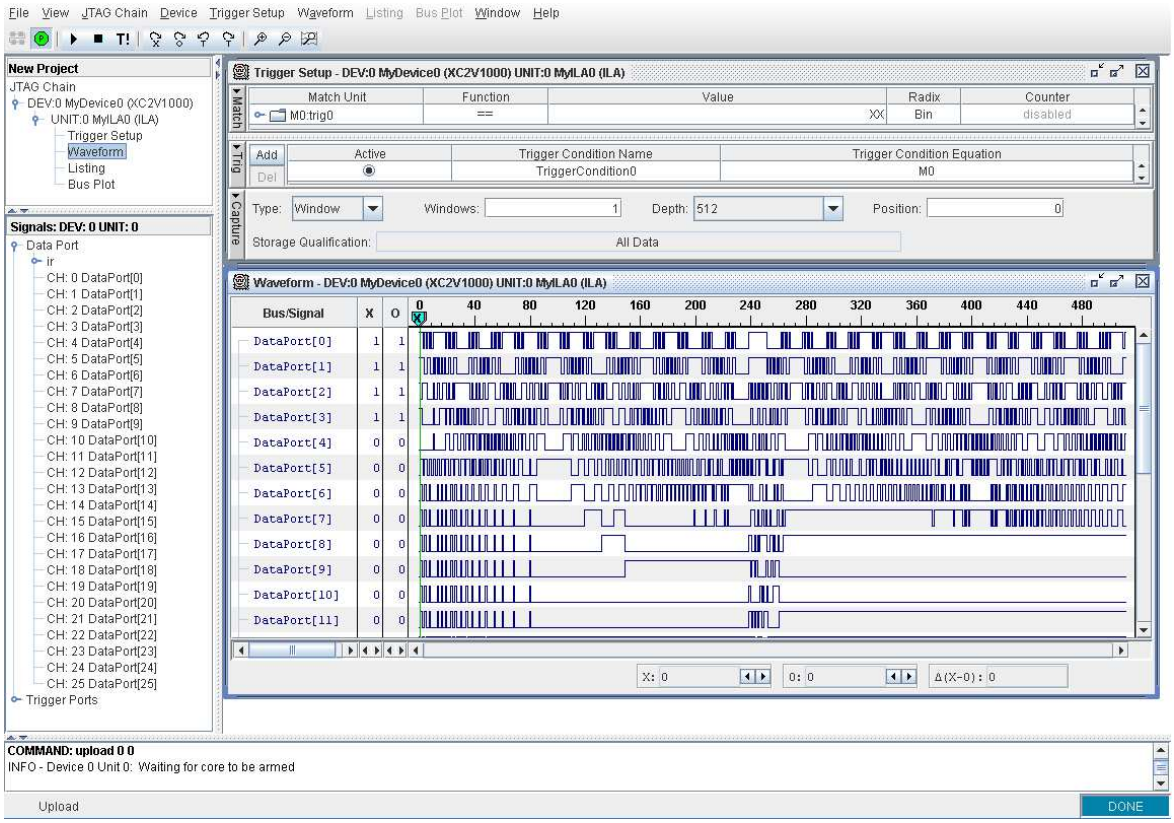


Figure 5.3: ChipScope Pro

Chapter 6

Conclusion & Future Work

Throughout this thesis work we found Matlab Simulink and Xilinx System Generator to be an effective development tool. The graphical nature of modelling made it easy to design efficiently and analyze the results quickly, which greatly reduced the development time. This method eliminates programming errors during the development phase as compared to hand written VHDL. A method is suggested in this report which explains all the necessary steps required for VHDL code generation and implementation using Simulink.

It was realized that the observer size could be reduced by keeping the observer word length to a minimum. It could be achieved by scaling, using lower sampling frequency and zero initial conditions.

The current available hardware was not sufficient enough for the effective implementation of the 4th order Hamiltonian observer.

In future the performance of the current 4th order Hamiltonian observer can be verified by upgrading the current hardware according to the recommendations.

By connecting the current implementation of 3rd order observer to a scaled down model of the power converter, further hardware testing can be performed.

Different upgraded variants of the current observer model with the increased states can be implemented and analyzed.

Better results could be obtained if the simulation is allowed to run for a longer amount of time, allowing the system to stabilize.

Each parameter in the observer model has different initial word length requirements, the model could be further optimised by varying different parameters seperately to get the maximum reduction in word length with minimum increase in error.

Clipping of u_w within the model to keep it below a certain value can be applied to the current model.

Bibliography

[1] M. Spierings and P. van Otterdijk, "Designing vhdl in matlab," tech. rep., Fontys University of Applied Science, Eindhoven, The Netherlands., 2008.

[2] R. W. Erickson and D. Maksimovic, *Fundamentals of Power Electronics*. Klewer Academic Publishers, second ed., 1999.

[3] A. Hultgren and M. Lenells, "Stability of a switched hamiltonian observer applied to a resonant converter," vol. Proceedings IEEE PESC 04, ISBN 07803-8400-8, IEEE, Power Electronics Specialists Society, 2004.

[4] A. Kabodi, "Development and simulation of a hamiltonian observer for an industrial application," Master's thesis, Blekinge Institute of Technology, 2008.

[5] www.wikipedia.org, [online] [cited: July 29, 2010].

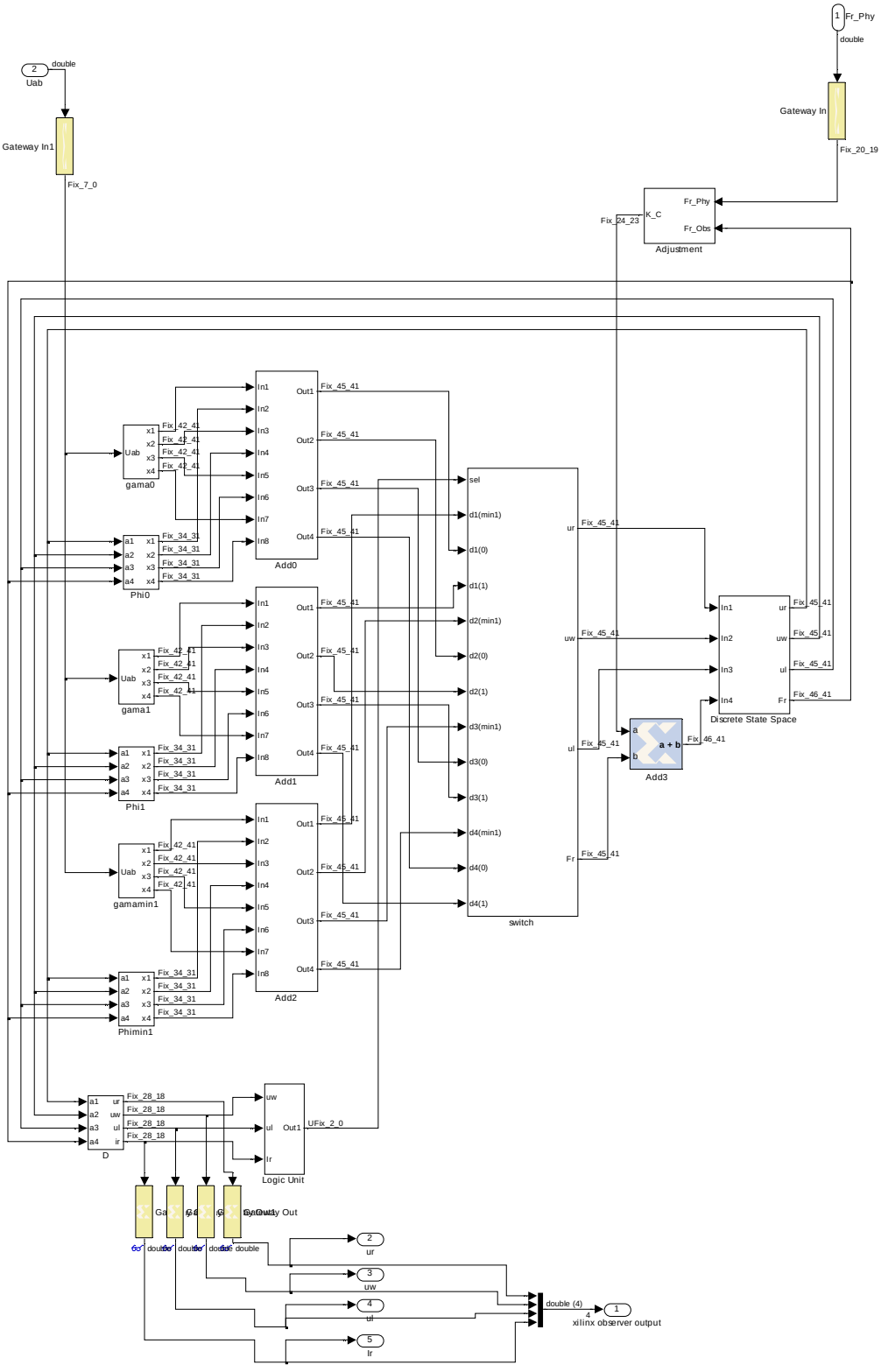
[6] Xilinx, *System Generator for DSP*, 10.1.1 ed., April 2008.

[7] L. Wanhammar and H. Johansson, *Digital Filters*. Department of Electrical Engineering Linköping Universitet, 2002.

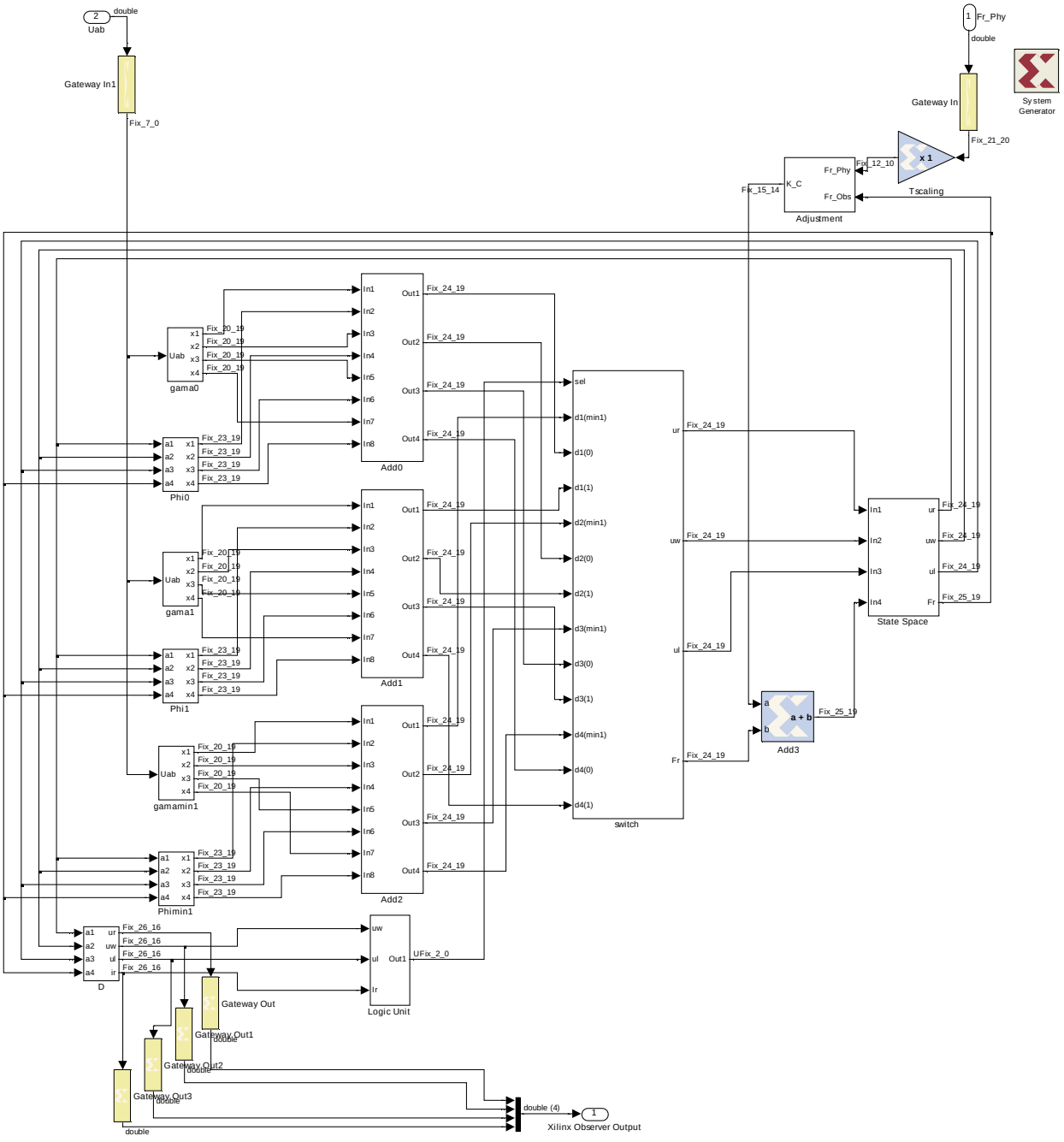
[8] www.xilinx.com/support, [online] [cited: July 30, 2010].

[9] D. D. Laboratory, *Introduction to Xilinx ISE 8.2i*. University of Pennsylvania.

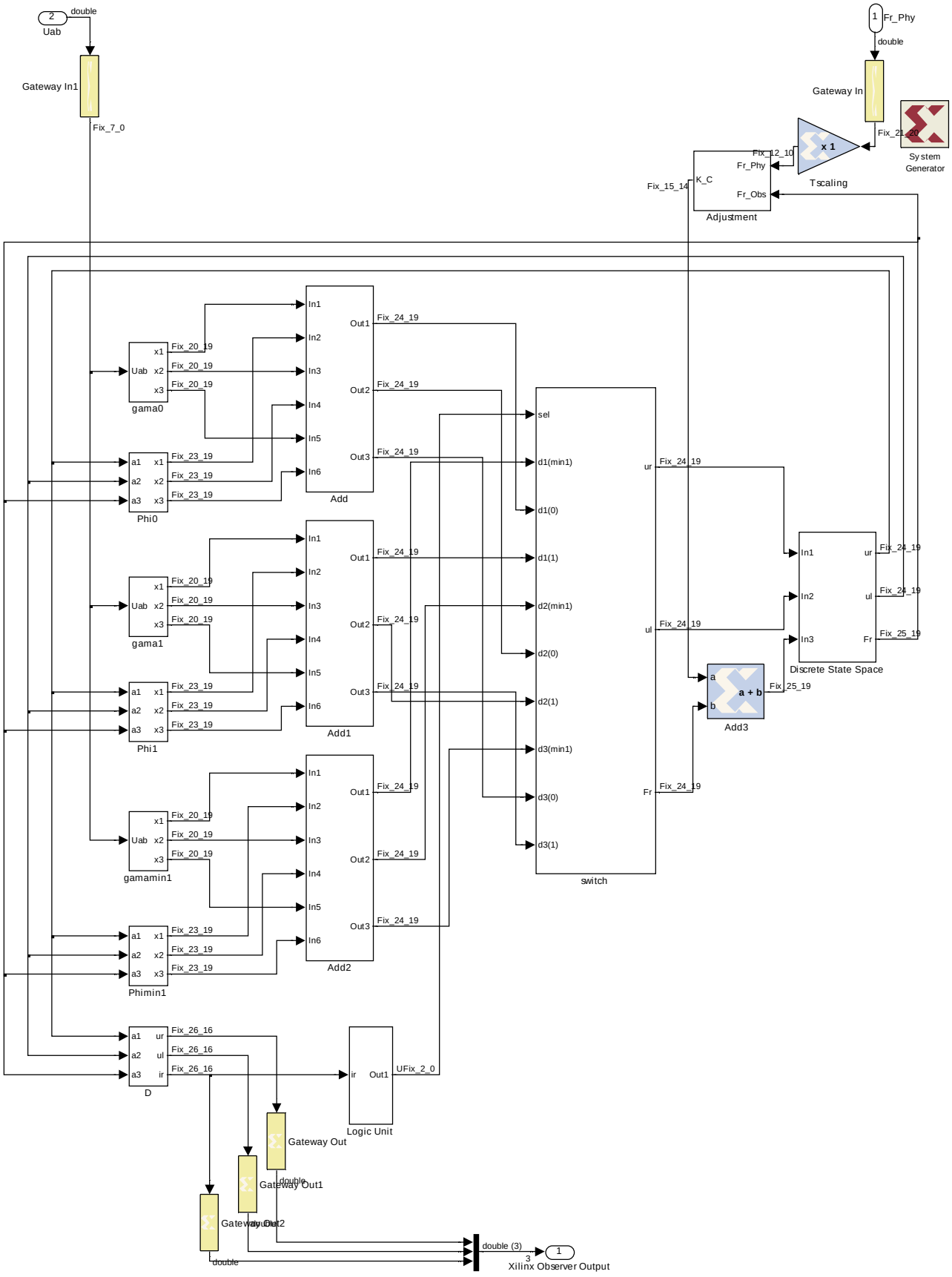
Appendix A1: Xilinx Models



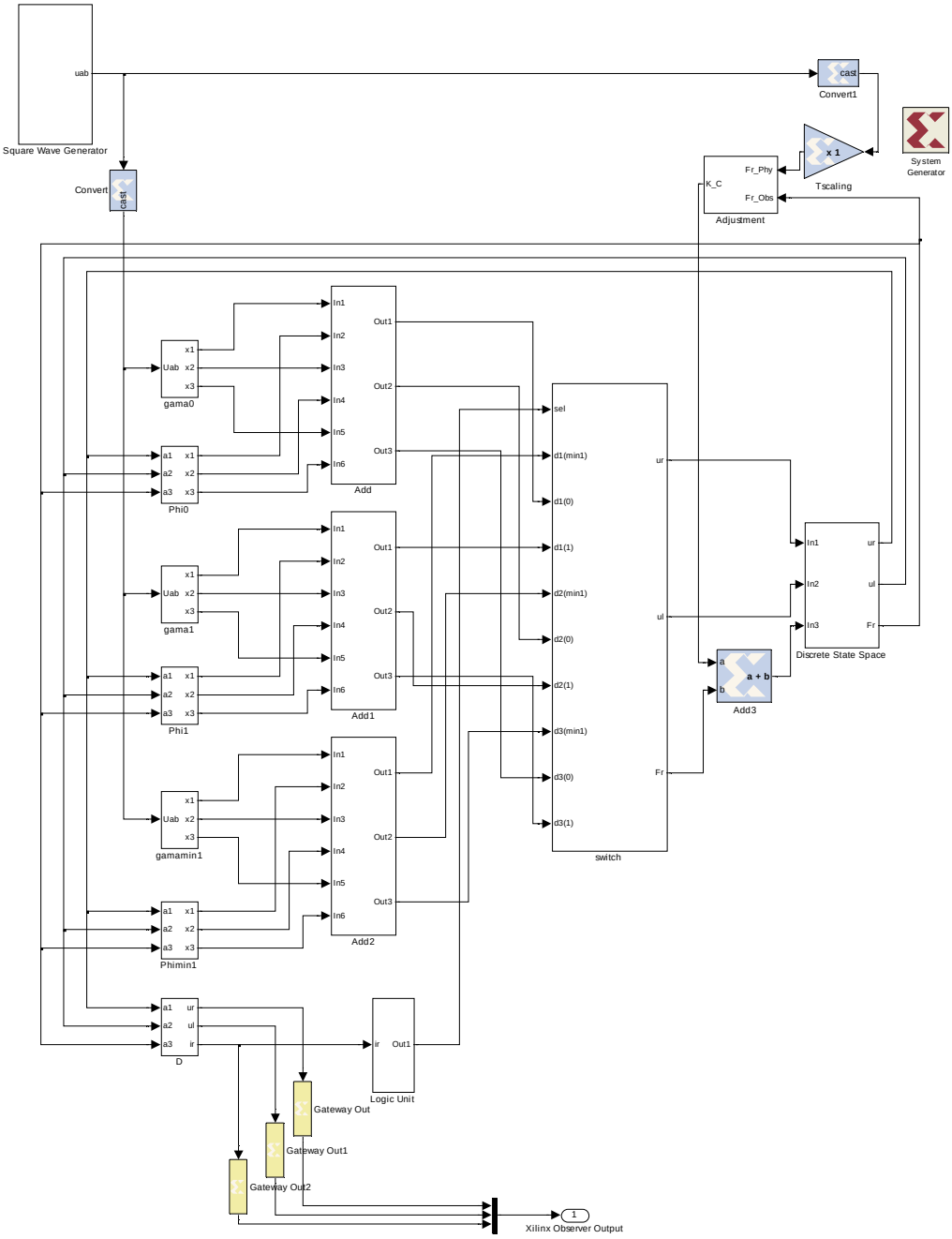
A1.1: 4th Order Observer model with wordlength



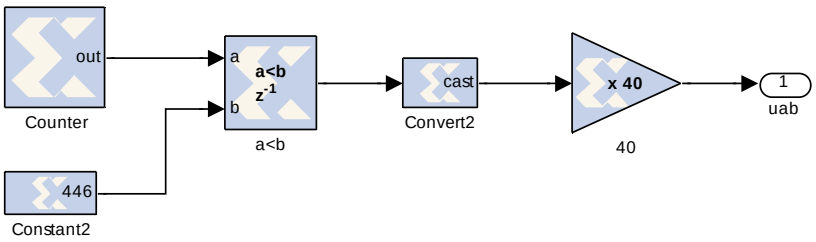
A1.2: 4th Order Scaled Observer model with word length



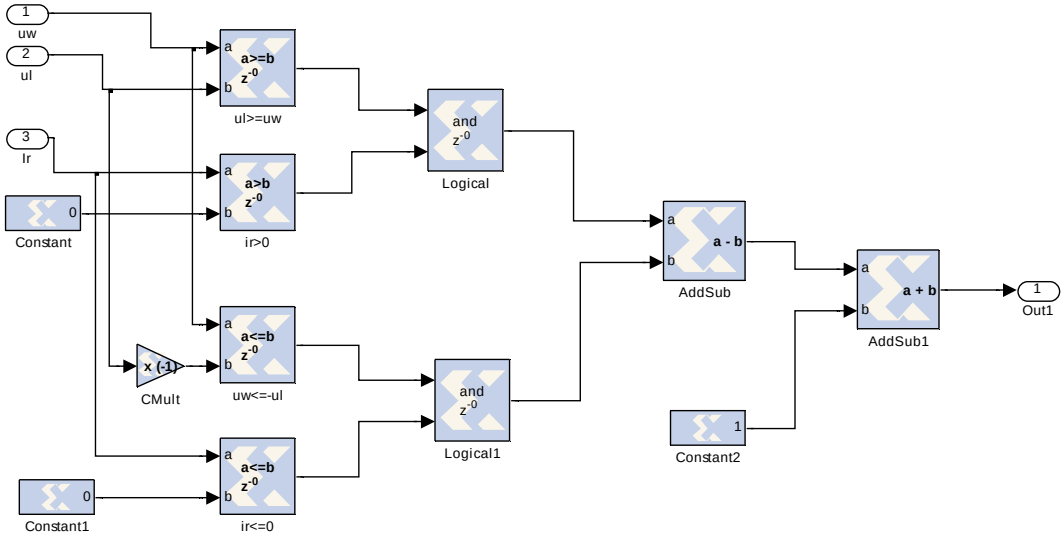
A1.3: 3rd Order Scaled Observer model with word length



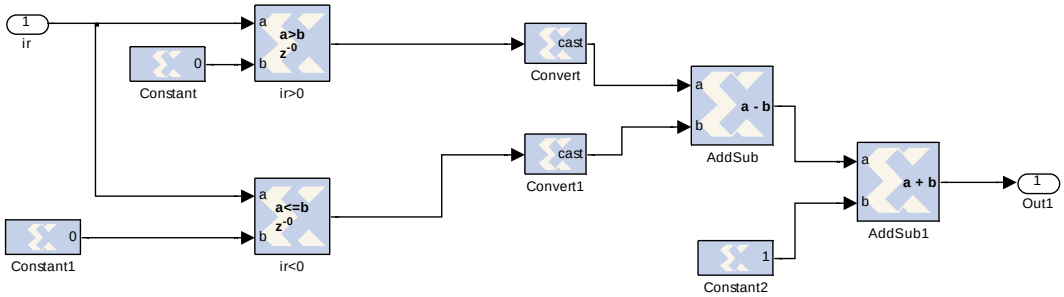
A1.4: 3rd Order Scaled Observer Model with ChipScope and Square Wave Generator



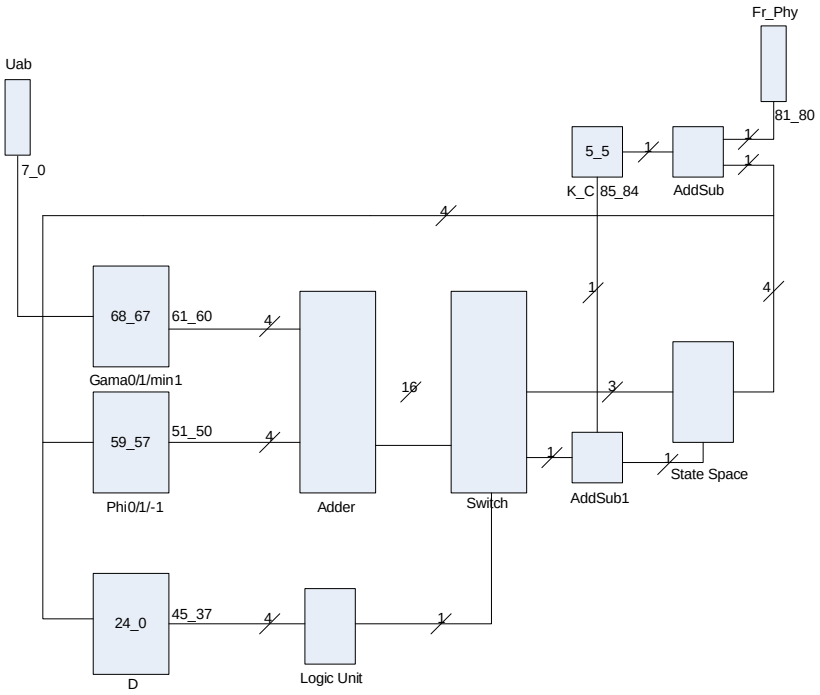
A1.5: Square Wave Generator



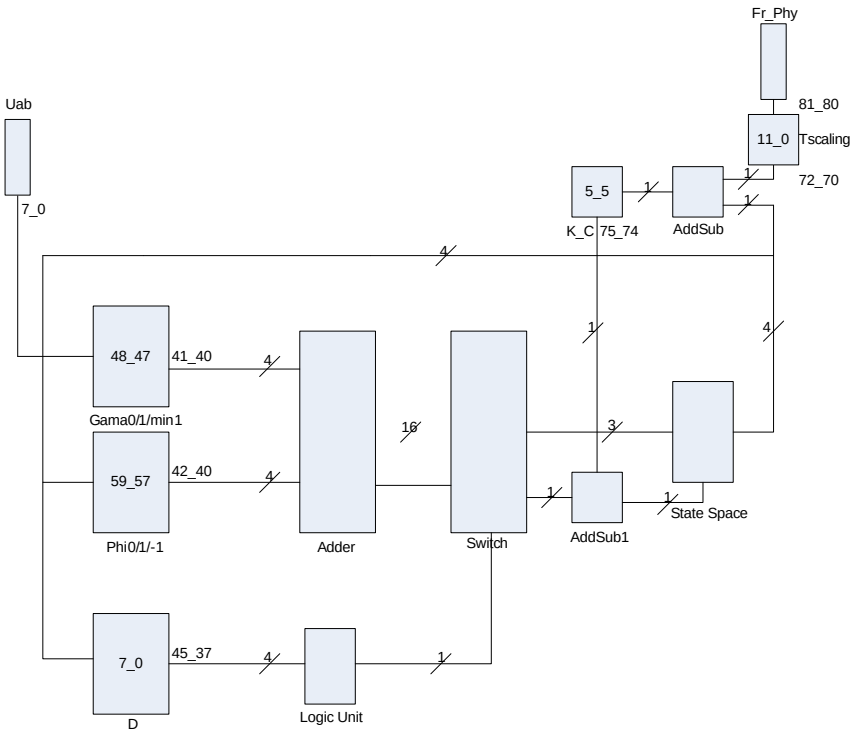
A1.6: 4th Order Observer Logic Unit



A1.7: 3rd Order Observer Logic Unit



A1.7: 4th Order Observer Initial Wordlengths



A1.8: 4th Order Scaled Observer Initial Wordlengths