Spike Removal Technology for Grid-Connected Power Converters

by

Haider Mohomad A R

B.Sc.E., University of New Brunswick, 2009

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science in Engineering

In the Graduate Academic Unit of Electrical and Computer Engineering

Supervisor: Chris Diduch, PhD, ECE
Yevgen Biletskiy, PhD, ECE
Liuchen Chang, PhD, ECE

Examining Board: Mary .E. Kaye, MEng, ECE
Howard Li, PhD, ECE

External Examiner: Zengtao Chen, PhD, ME

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

May, 2012

©Haider Mohomad A R, 2012

Abstract

Power Electronic Converters (PECs) are routinely used to control the energy supplied to the electric grid by distributed generators. The operation of PECs involves the control of voltage and current by rapidly switching power semiconductor devices. The rapid switching of these devices causes a certain class of high frequency noise called spikes to be induced into sensor measurements. In addition to PEC control functions, the sensor measurements are also used to detect over current (OI), over voltage (OV) and under voltage (UV) faults. The presence of noise spikes compromises the performance and leads to unnecessary triggering of fault protection circuitry as a result of false detection of over/under voltage and current faults. The development of a new algorithm to remove short time electrical spikes from PEC sensors is the focus of this thesis. In such a way, the PEC is unaffected and false OI, OV, and UV alarms are eliminated. The methodology uses the discrete wavelet transform (DWT) to generate a feature space where the spike feature is detected and removed, and the inverse DWT to reconstruct spike-free measurements. A Field-Programmable Gate Array (FPGA)-based interface board is designed to test the new algorithm within an actual PEC system. The designed FPGA board along with the new algorithm will be used in the integration of distributed generators into power systems.

Table of Contents

Abstract ii

Table of Contents iii

List of Tables vi

List of Figures vii

Nomenclature xii

1 Introduction 1

1.1 Background and Literature Review 3

1.2 Thesis Contribution 5

1.3 Thesis Structure 6

2 Methodology 7

2.1 Discrete Wavelet Transform 7

2.1.1 Haar Mother Wavelet 9

2.1.2 Daubechies Mother Wavelets 12

2.2 Modified Wavelet De-noising Technique 14

2.3 Proposed DWT Spike Removal Algorithm 15

3 Hardware and Software System Design 18

3.1 FPGA Realization 18
3.2 Xilinx System Generation 20
3.2.1 Gateway In/Out Blocks 21
3.2.2 System Generator Block 22
3.2.3 MCode Block 22
3.2.4 Interface Board Design 22

4 Simulation **26**
4.1 Modified Wavelet De-noising Technique 27
4.1.1 Haar Wavelet Simulation 27
4.1.2 Daubechies Wavelets Simulation 29
4.2 Proposed Spike Removal Algorithm 33
4.2.1 Haar Wavelet Simulation 33
4.2.2 Daubechies Wavelets Simulation 34
4.3 Current Fault Detection Simulation 38
4.3.1 No Fault Condition With Noise Spikes 39
4.3.2 Fault Condition With Noise Spikes 39

5 Implementation **41**
5.1 Proposed Spike Removal Implementation 42
5.2 Fault Detection Implementation 43

6 Discussion **45**

7 Conclusion and Future Work **48**
7.1 Conclusion 48
7.2 Major Contribution 49
7.3 Future Work 49

Bibliography **50**

Appendices **54**

A Software Design **54**

A.1 Simulink Modules 54

A.2 FPGA Input/Output ports 61

A.3 Source Code 63

A.3.1 AD7266 source code 63

A.3.2 ADS7883 source code 66

A.3.3 Serial Peripheral Interface (SPI) Source Code 69

A.3.4 Latched Source Code 74

B Interface Board PCB Design **75**

B.1 Inpot/Output Ports 75

B.2 FPGA 79

B.3 Power Regulator 84

B.4 Analog to Digital Converters 87

B.5 Serial Flash Memory 88

B.6 Anti Aliasing Filters 88

B.7 Relay 91

B.8 Clock circuit 92

B.9 Phase Select 92

B.10 LCD Displayer 93

B.11 Other 93

Vita **95**

List of Tables

2.1 Haar mother wavelet filter coefficients 9

2.2 Daubechies family (db2, db4 and db8) wavelets filter coefficients . . . 12

2.3 Daubechies wavelet cutoff frequencies 13

A.1 Input ports of FPGA chip 61

A.2 Output ports of FPGA chip 62

List of Figures

1.1 Block diagram of power electronics converter in the wind energy system 2

2.1 Dyadic DWT filter bank 8

2.2 Haar filters frequency response 9

2.3 Amplitude spectrum of the sensor measurements 10

2.4 Sensor measurements decomposition using Haar wavelet 11

2.5 Daubechies filters frequency response 13

2.6 Sensor measurements decomposition using Daubechies wavelets . . . 14

2.7 Modified wavelet de-noising technique [1] 15

2.8 Adopted modified wavelet de-noising technique for PEC measurements 16

2.9 Proposed spike removal algorithm 16

3.1 FPGA-based interface board 19

3.2 Example of Xilinx DSP building blocks 20

3.3 Xilinx blockset within the Simulink Library Browser 21

3.4 Simulink model of the interface board 24

3.5 FPGA realization for the proposed spike removal algorithm 25

3.6 Filter structure for the implementation of the DWT 25

4.1 Simulink model used in the simulation 26

4.2 Simulation input signal 27

4.3 Approximation and detail coefficients of the input signal using the Haar wavelet. 28
4.4 Simulation results for the modified wavelet de-noising technique using the Haar wavelet. 28
4.5 Approximation and detail coefficients of the input signal using db2 wavelet. 29
4.6 Simulation results for the modified wavelet de-noising technique using db2 wavelet. 30
4.7 Approximation and detail coefficients of the input signal using the db4 wavelet. 31
4.8 Simulation results for the modified wavelet de-noising technique using the db4 wavelet. 31
4.9 Approximation and detail coefficients of the input signal using the db8 wavelet. 32
4.10 Simulation results for the modified wavelet de-noising technique using the db8 wavelet. 32
4.11 Approximation and detail coefficients of the input signal using the Haar wavelet. 33
4.12 Simulation results for the spike removal algorithm using the Haar wavelet. 34
4.13 Approximation and detail coefficients of the input signal using the db2 wavelet. 35
4.14 Simulation results for the spike removal algorithm using the db2 wavelet. 35
4.15 Approximation and detail coefficients of the input signal using the db4 wavelet. 36
4.16 Simulation results for the spike removal algorithm using the db4 wavelet. 36
4.17 Approximation and detail coefficients of the input signal using the db8 wavelet. 37

4.18 Simulation results for the spike removal algorithm using the db8 wavelet. 37
4.19 Simulink module for simulating the detection of current faults. 38
4.20 Simulation results of the spike removal algorithm: 3-phase current measurements (top), reconstructed measurements with spikes removed (middle) and protection circuits trigger (bottom). 39
4.21 Simulation results of the spike removal algorithm when a fault occurs: 3-phase current measurements (top), reconstructed measurements with spikes removed (middle) and protection circuits trigger (bottom). . . 40

5.1 Implementation structure. 41
5.2 Implementation results for the spike removal using Haar DWT. 43
5.3 Results of the FPGA realization of the spike removal filter: (1-phase current measurements (top), reconstructed measurements with spikes removed (middle) and protection circuits trigger (bottom). 44
5.4 Results of the FPGA realization of the spike removal filter when a fault occurs: (1-phase current measurements (top), reconstructed measurements with spikes removed (middle) and protection circuits trigger (bottom). 44

A.1 Simulink module design of the interface board 55
A.2 Simulink module design of the spike removal algorithm 56
A.3 Simulink module design of the Haar filter 56
A.4 Simulink module design of the latched control hardware interrupt . . 57
A.5 Simulink module design of the OI detection for current signals 58
A.6 Simulink module design of the current comparator 58
A.7 Simulink module design of the OI detection for boost-current signal . 59
A.8 Simulink module design of the OV detection 59
A.9 Simulink module design of the OT detection 59

A.10 Simulink module design of IGBT driving signals 60
A.11 Simulink module design of ADS7883 ADC 60

B.1 Driver signals jumper 75
B.2 Sensor signals jumper 76
B.3 SPI and CS jumpers 76
B.4 Interface board to DSP jumpers 77
B.5 Interface board jumpers 78
B.6 FPGA Bank 1 79
B.7 FPGA Bank 2 79
B.8 FPGA Bank 3 80
B.9 FPGA Bank 4 80
B.10 FPGA control and configuration 81
B.11 FPGA power and ground 82
B.12 FPGA mode select 83
B.13 Power regulator supply 84
B.14 Power supply circuit for 12V 85
B.15 Power supply circuit for 5V 85
B.16 Power supply circuit for 1.8V 85
B.17 Power LED indicator 86
B.18 Reference voltage detection 86
B.19 AD7266 ADC 87
B.20 ADS7883 ADC 87
B.21 16 Mbit serial flash memory 88
B.22 Anti aliasing filter for grid A voltage sensor 88
B.23 Anti aliasing filter for grid B voltage sensor 89
B.24 Anti aliasing filter for grid C voltage sensor 89
B.25 Anti aliasing filter and level shifting for Phase A current sensor . . . 90

B.26 Anti aliasing filter and level shifting for Phase B current sensor 90
B.27 Anti aliasing filter and level shifting for Phase C current sensor 90
B.28 Relay drivers Signals 91
B.29 Relay circuits 91
B.30 Clock circuits 92
B.31 Phase select circuits 92
B.32 LCD displayer circuits 93
B.33 Status LED 93
B.34 Level conversion circuitry 94

Nomenclature

Abbreviation

PEC	Power Electronic Converter
OI	Over Current
OV	Over Voltage
UV	Under Voltage
FPGA	Field Programmable Gate Array
IGBT	Insulated Gate Bipolar Transistor
WT	Wavelet Transform
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
AP	Action Potentials
PCB	Printed Circuit Board
FIR	Finite Impulse Response
dB	DeciBels
DSP	Digital Signal Processing
JTAG	Joint Test Action Group
ADC	Analog to Digital Converter
I/O	Input/Output
ChipScope ILA	ChipScope Pro Integrated Logic Analyzer

IWE	IntuiLink Waveform Editor
db2	Second Order Daubechie Wavelet
db4	Forth Order Daubechie Wavelet
db8	Eighth Order Daubechie Wavelet
SPI	Serial Peripheral Interface

Symbols

$H_o(z)$	Dicomposting High Pass Filter
$G_o(z)$	Dicomposting Low Pass Filter
$H_1(z)$	Reconstructing High Pass Filter
$G_1(z)$	Reconstructing Low Pass Filter
C	Modified Wavelet De-noising Input Signal
d	Modified Wavelet De-noising Detail Coefficients
a	Modified Wavelet De-noising Approximation Coefficients
μ	Input Signal
μ_h	Detail Coefficients
μ_g	Approximation Coefficients
$\hat{\mu}_h$	Adjusted Detail Coefficients
$\hat{\mu}_g$	Adjusted Approximation Coefficients
y	Output Signal
α	Selected Threshold
ω_{max}	Valid Spike Maximum Duration

Chapter 1

Introduction

In the recent years, renewable energy such as wind energy and solar energy has become more popular worldwide [2, 3, 4, 5]. The share in electrical power production of wind energy has been greatly increasing for many reasons, some of them being that wind energy is cost-effective, environmentally friendly, and an alternative to non-renewable energy sources. For utilization of renewable energy, extensive research has been done on grid-connected PECs. Grid-connected PECs belong to a family of electrical apparatuses which transform the variable AC or DC electricity into standard AC power to feed to the utility grid controling rapidly switching power semiconductor devices. In the utilization of wind energy, PECs are routinely used to control frequency, voltages, currents, power and harmonics necessary for connecting to the grid [6]. The rapid switching causes a certain class of high frequency noise called spikes to be induced upon sensor measurements. The sensors which may be part of the PEC or in proximity to it, in addition to converter control may also be used to detect over current (OI), over voltage (OV) and under voltage (UV) faults. The presence of noise spikes in measurements of voltages and currents is problematic to the analysis of data since it compromises real-time detection of abnormal conditions such as over/under voltage and current faults [7]. The presence of large spikes results

in an increase in false detection of faults. The objective of this research focuses on the development of an innovative spike removal technology for grid connected PECs, thus helping to improve the programmability, reliability and communication functions of the PECs. A Field Programmable Gate Array (FPGA)-based interface board is chosen as the development platform for the PECs. The Interface board is a signal processing unit that is used for real-time voltage/current/temperature detection and protection of the PECs. To provide the protection function, the interface board communicates with other components of the PEC as shown in Figure 1.1 including i) a DSP board, used for protection and control , ii) sensors which monitor voltages and currents, and iii) an insulated gate bipolar transistor (IGBT) driver board which is used for controlling the switching devices.

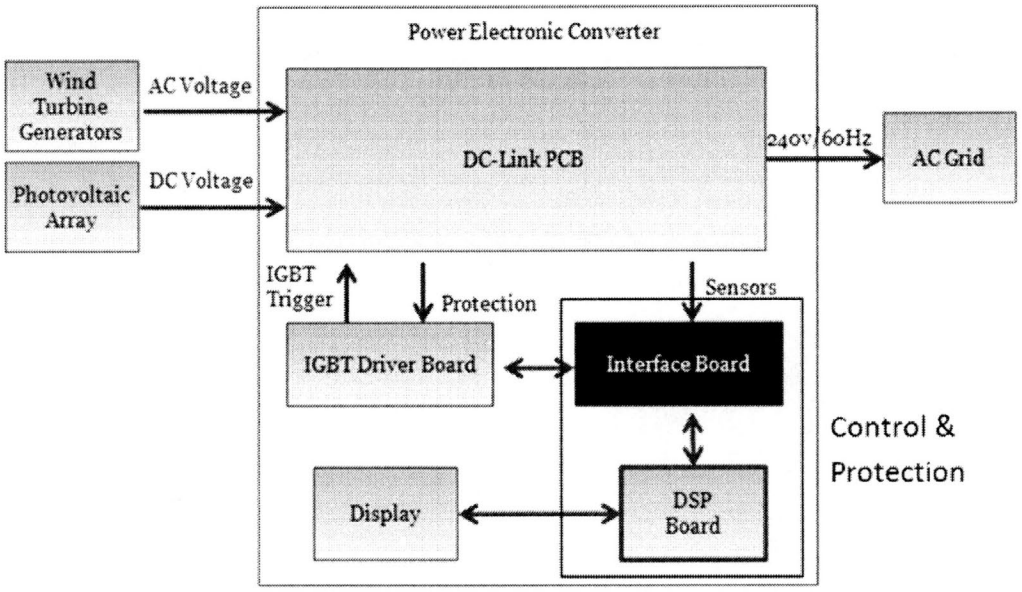

Figure 1.1: Block diagram of power electronics converter in the wind energy system

The interface board consists of a control and protection circuit for the purpose of i) feedback control of the switching action of power semiconductor devices, ii) fault detection and iii) activating protection relays and breakers. Reliable operation depends on valid measurements of voltage, current and temperature. The presence of noise

spikes in sensor measurements leads to: i) false alarms, the result of detecting the presence of a fault when none existed [7] and ii) degraded performance of the control loop of the PEC which may further lead to triggering of protection circuitry.

1.1 Background and Literature Review

The existence of spikes often masks the details of the actual measurements and leads to false identification of the signal of interest [8]. Therefore, it is necessary to remove these spikes from measurements. Most of the traditional methods for noises reduction including Savitzky-Golay (SG) smoothing [9], the adaptive degree polynomial filter [10], and the Butterworth or Chebyshev filter [11] are not directly applicable because they change the signal shape [8].

Automatic detection of short duration spikes is more complicated than manual detection especially when the signal is accompanied by other forms of noise [8]. Outlier detection algorithms, which are based on monitoring higher moments such as skewness and kurtosis [12] were proposed since spikes might be treated as an outlier from the actual signal. However, for real-time detection of fault conditions in PECs, the spike removal algorithm must be fast to compute and easy to implement.

A multi-resolution filtering techniques based on wavelet transform is capable of removing spikes without corrupting the data significantly [7]. The wavelet transform (WT) is an emerging signal processing technique that can be used to represent real-life non-stationary signals with high efficiency [13]. The WT has gained the reputation of being a very effective technique for several practical applications such as transient signal analysis [14], numerical analysis [15], computer vision [16], and image compression [17], among many other audiovisual applications [18]. The WT is a time-frequency analysis tool that expresses a signal as a linear combination of basis functions, similar to the Fourier transform. However, unlike the Fourier transform which has only

frequency resolution, the WT has both frequency and time resolution [19]. In the WT, high frequency components are analyzed through narrow window(s) while low frequency components are analyzed through wide window(s). Narrow windows mean low scales which provide good time resolution and poor frequency resolution. While wide windows mean high scales which provide good frequency resolution and poor time resolution. Due to the multi-resolution analysis of the WT, it is considered a very powerful tool in the study of low frequency signals that have sharp spikes, fast changes, and limited duration impulse components [20].

A set of orthogonal basis functions is generated by scaling and translating a basic function, called mother wavelet. Scaling, also known as dilation, stretches the mother wavelet while translating shifts the mother wavelet along the time axis [21]. The mother wavelet is an oscillatory function that satisfies the conditions of admissibility and regularity. Admissibility implies that the mother wavelet is a wave in the time domain and regularity implies that the mother wavelet possesses smoothness and concentration in both the time and frequency domains [19].

The WT may be categorized into continuous wavelet transform (CWT) and discrete wavelet transform (DWT). The CWT is defined as the inner product of the signal with each of the basis functions. This represents the similarity between the signal and each of the basis functions. The CWT basis functions are generated by continuously scaling and translating the mother wavelet. However, this implies a considerable computing effort for large amounts of data [21]. Therefore, the DWT is often used to overcome the redundancy of the CWT by scaling and translating in discrete steps [19].

The DWT is very important in most computer implementations and digital signal processing applications of the WT. An effective implementation of the DWT is given by the dyadic filter bank in Mallat [22]. A filter bank that consists of a high pass filter in parallel with a low pass filter are usually called quadrature mirror

decomposition filters [23]. The filter outputs are down sampled to halve the bandwidth of the filters' output [20]. The dyadic filter bank decomposes the signal into the so-called detail and approximation coefficients. The detail coefficients are the output of the high pass filter while the approximation coefficients are the output of the low pass filter. On the other hand, to perfectly reconstruct the signal from its detail and approximation coefficients, two up samplers followed by two reconstruction filters are used. The coefficients of the decomposition and the reconstructed filters are determined by the mother wavelet type [24]. It is possible to attenuate certain signal frequencies by manipulation of wavelet coefficients at the different levels. [1] has demonstrated that a modified wavelet de-noising technique is useful for accurate detection of action potentials activity (AP), i.e. spikes. In this process, a nerve signal with additive noise is decomposed using DWT and a threshold is applied to each of the detail coefficient levels. All coefficients with an absolute value greater than the threshold are thought to be part of an action potential, i.e. a spike.

1.2 Thesis Contribution

In the PEC of wind energy systems, the fault detection system may suffer from false fault alarms due to the presence of spikes in PEC measurements. A traditional approach to overcome this problem is to disable the fault protection system over intervals when noise spikes are expected to occur. The goal of this thesis is to develop an innovative algorithm using the DWT to generate a feature space where the spike feature is detected and removed, and using the inverse DWT to reconstruct spike-free measurements. This algorithm is capable of removing spikes while maintaining the fidelity of the sensor measurments in both low and high frequency ranges. This results in i) the elimination of false triggering of protection circuits and ii) improved PEC performance because noise-free measurements are fed back to the PEC controller.

1.3 Thesis Structure

The thesis is organized as follows: Chapter 2 describes the methodology of removing spikes from the measurements. Chapter 3 outlines the details of the hardware and software system design. Chapter 4 shows the simulation results of the spike removal algorithm. Chapter 5 shows the implementation results of the spike removal algorithm. Chapter 6 discusses the simulation and implementation results. Chapter 7 presents conclusions, summarizes the work completed in the thesis and outlines potential future work for the project. Details of the software design and source codes are in Appendix A. Printed Circuit Board (PCB) schematics of the interface board are in Appendix B.

Chapter 2

Methodology

The methodology of removing spikes from the measurements uses the DWT to generate a feature space, where the spike feature can be detected and removed, and then uses the inverse DWT to reconstruct spike-free measurements. The performance depends on the selection of the mother wavelet, the filter structure, the number of levels of decomposition and the correction method.

2.1 Discrete Wavelet Transform

The dyadic filter bank shown in Figure 2.1 is used for implementation of the DWT. The dyadic DWT filter bank decomposes the signal into the so-called detail and approximation coefficients, and it permits signal reconstruction from these coefficients using the inverse DWT. The decomposition stage consists of a high pass filter, $H_0(z)$, and a low pass filter, $G_0(z)$, that are used to generate the detail coefficients and the approximation coefficients. The decomposition stage filters have the property,

$$|H_0(z)|^2 + |G_0(z)|^2 = 1 \tag{2.1}$$

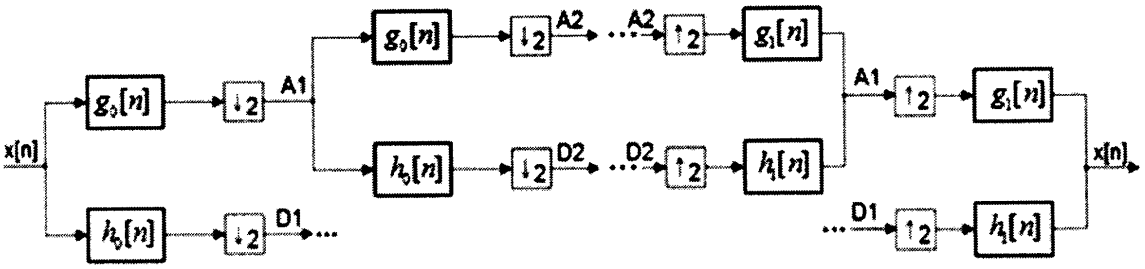

Figure 2.1: Dyadic DWT filter bank

Down sampling is achieved by setting odd samples of the output signal of each filter to zero, halving the bandwidth for the next stage of decomposition. The number of levels of the decomposition can determine the performance of the DWT. Decomposing the signal into a higher level generates a better feature space where the spike feature can be detected and removed, however this increases the input-output time delay. The correction method is applied to the feature space, the detail and approximation coefficients, in order to remove the spikes without corrupting the true measurements.

Another pair of low and high pass filters along with a pair of up samplers are used to reconstruct the spike-free measurements from the detail and approximation coefficients. The up sampler simply inserts zeroes between each sample of the signal. For a perfect reconstruction, the Finite Impulse Response (FIR) filters that are used in the dyadic filter bank must meet the following two conditions [18]: unity transfer function condition and aliasing cancelation condition as expressed in Equation 2.2 and Equation 2.3:

$$G_1(z)G_0(z) + H_1(z)H_0(z) = 2 \tag{2.2}$$

$$G_1(z)G_0(-z) + H_1(z)H_0(-z) = 0 \tag{2.3}$$

The type of the mother wavelet used in the DWT determines the performance of decomposition since it determines filter coefficients and length. In general, greater

filter length leads to better results for low frequency estimation on the account of the complexity computation specially real time [20]. Therefore, selecting the appropriate mother wavelet for spike removal application is critical.

2.1.1 Haar Mother Wavelet

The Haar wavelet is the simplest type of mother wavelet which can be implemented using two-tap FIR digital filters. Table 2.1 shows the filter coefficients corresponding to the Haar wavelet. The Haar DWT decomposes the input signal into two sub-signals where each of the sub-signals has half the length of the original signal. One signal is a running average and the other is a running difference [18].

Table 2.1: Haar mother wavelet filter coefficients

	Low Pass Filter	High Pass filter
Decomposition	[0.707, 0.707]	[0.707, -0.707]
Reconstruction	[0.707, 0.707]	[-0.707, 0.707]

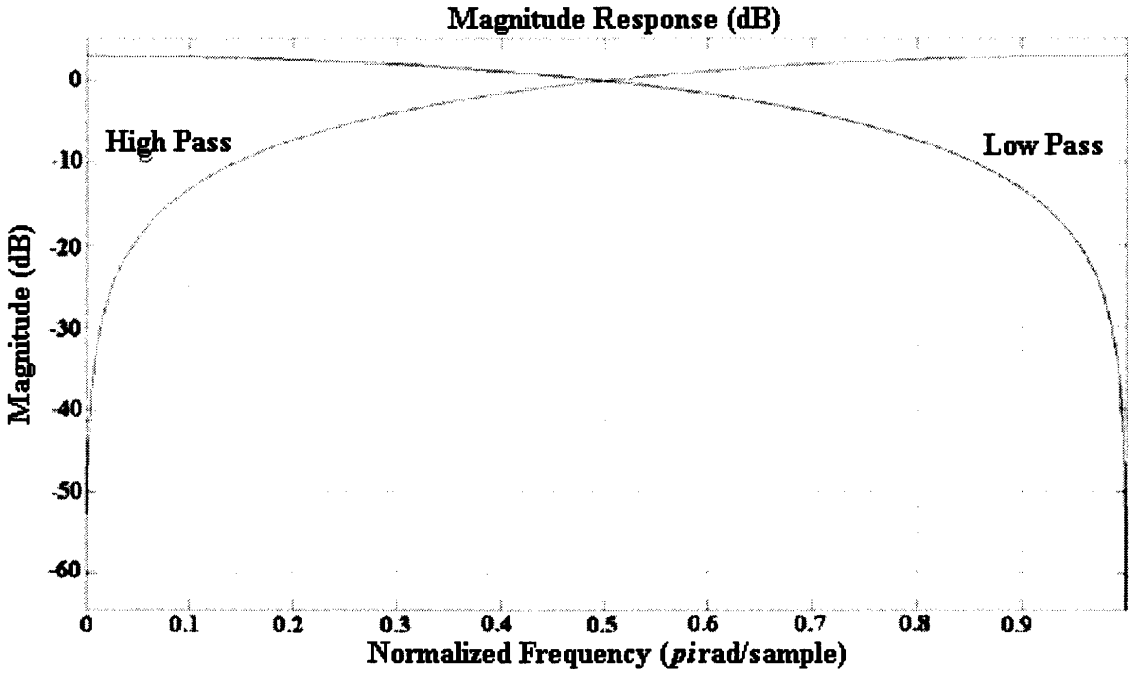

Figure 2.2: Haar filters frequency response

Figure 2.2 shows the frequency response for the Haar low pass and high pass filters. The low pass filter passes frequencies up to 0.6623π rad/sample while the high pass filter passes frequencies above 0.3376π rad/sample. Therefore, the frequencies between 0.3376π rad/sec and 0.6623π rad/sample will appear in the approximation and detail coefficients.

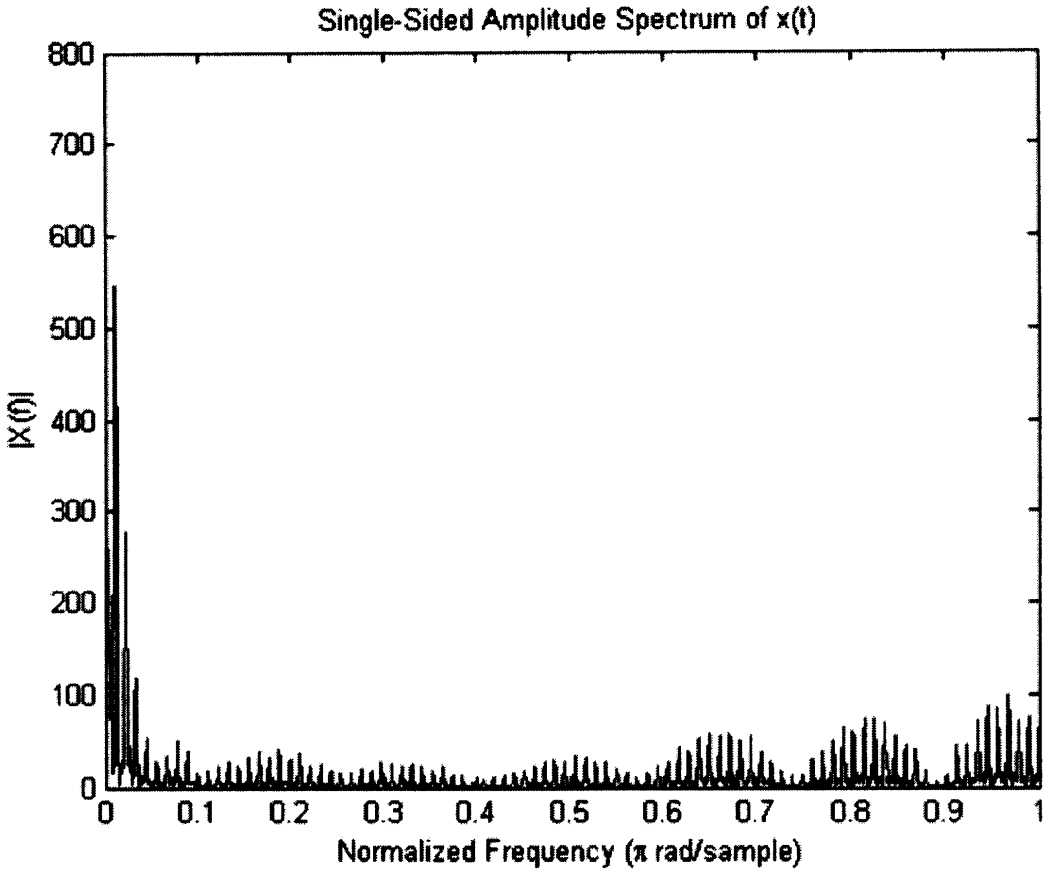

Figure 2.3: Amplitude spectrum of the sensor measurements

Figure 2.3 shows the amplitude spectrum of current sensor measurements. It can be seen that the power of the sensor signal is concentrated below 0.05π rad/sample. However, the noise and spike power is spread over the range of 0.1 to 1 π rad/sample with higher power concentrated at higher frequencies. Therefore, decomposing such a signal using the Haar wavelet, will result in approximation coefficients

that contain components of the signal together with components of spikes, and detail coefficients that contain mostly high frequency spikes and noise. Figure 2.4 illustrates the current sensor measurements, which were recorded from an actual converter, and their decomposition using the Haar wavelet transform. It can be seen that the power associated with the spikes is distributed across the approximation and the detail components.

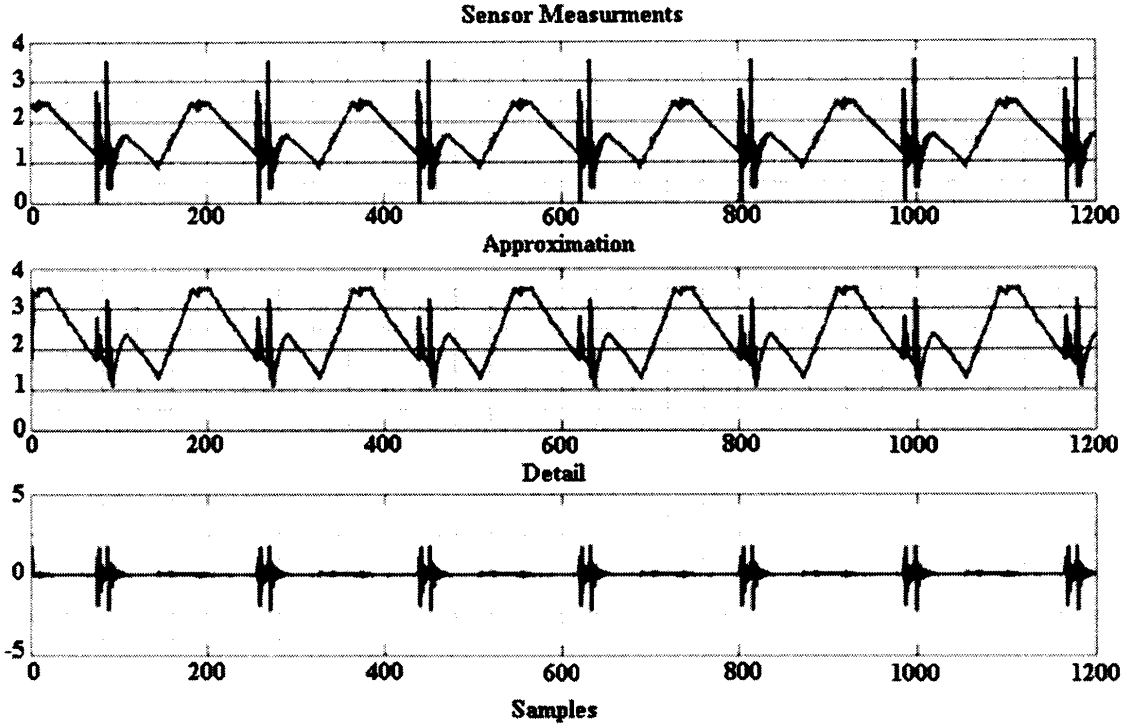

Figure 2.4: Sensor measurements decomposition using Haar wavelet

The Haar wavelet has numerous advantages: it is simple, fast, memory efficient, and exactly reversible without edge effects [25]. However, the Haar wavelet has some limitations [26]. The Haar wavelet uses a sliding window of length two, two-tap FIR, for computing the detail and approximation coefficients. Therefore, if any significant change takes place between an even sample and an odd sample, this change will not be reflected in the detail coefficients due to the down sampler effect.

2.1.2 Daubechies Mother Wavelets

The Daubechies wavelets are a family of orthogonal wavelets [27]. Similar to the Haar wavelet, they compute the running average and difference of the input signal. However, Daubechies wavelets are more complicated and more computationally expensive than the Haar wavelets since they use overlapping windows, therefore, all high frequency changes are reflected in the detail coefficients [18]. The names of the Daubechies wavelets family are expressed as dbN, where db refers to Daubechies wavelet and N refers to the order of the wavelet. It can be implemented using a 2N-tap FIR filter [27]. Table 2.2 shows the filter coefficients corresponding to three orders of the Daubechies wavelets.

Table 2.2: Daubechies family (db2, db4 and db8) wavelets filter coefficients

Decomposition						Reconstruction					
Low Pass			High Pass			Low Pass			High Pass		
db8	db4	db2	db8	db4	db2	db8	db4	db2	db8	db4	db2
0	-0.011	-0.129	-0.054	-0.230	-0.483	0.054	0.230	0.483	0	-0.011	-0.129
0.001	0.033	0.224	0.313	0.715	0.837	0.313	0.715	0.837	-0.001	-0.033	-0.224
0	0.031	0.837	-0.676	-0.631	0.224	0.676	0.631	0.224	0	0.031	0.837
-0.005	-0.187	0.483	0.585	-0.028	-0.129	0.585	-0.028	-0.129	0.005	0.187	-0.483
0.009	-0.028	-	0.016	0.187	-	-0.016	-0.187	-	0.009	-0.028	-
0.014	0.631	-	-0.284	0.031	-	-0.284	0.031	-	-0.014	-0.631	-
-0.044	0.715	-	-0.001	-0.033	-	0.001	0.033	-	-0.044	0.715	-
-0.017	0.230	-	0.129	-0.011	-	0.129	-0.011	-	0.017	-0.230	-
0.129	-	-	0.017	-	-	-0.017	-	-	0.129	-	-
0.001	-	-	-0.044	-	-	-0.044	-	-	-0.001	-	-
-0.284	-	-	-0.014	-	-	0.014	-	-	-0.284	-	-
-0.016	-	-	0.009	-	-	0.009	-	-	0.016	-	-
0.585	-	-	0.005	-	-	-0.005	-	-	0.585	-	-
0.676	-	-	0	-	-	0	-	-	-0.676	-	-
0.313	-	-	-0.001	-	-	0.001	-	-	0.313	-	-
0.054	-	-	0	-	-	0	-	-	-0.054	-	-

Figure 2.5 shows the frequency response for the three orders of the Daubechies low and high pass filters. Table 2.3 shows the -3db frequencies of the decomposing filters of db2, db4, and db8.

Table 2.3 shows that as the order of the Daubechies wavelet increases, the

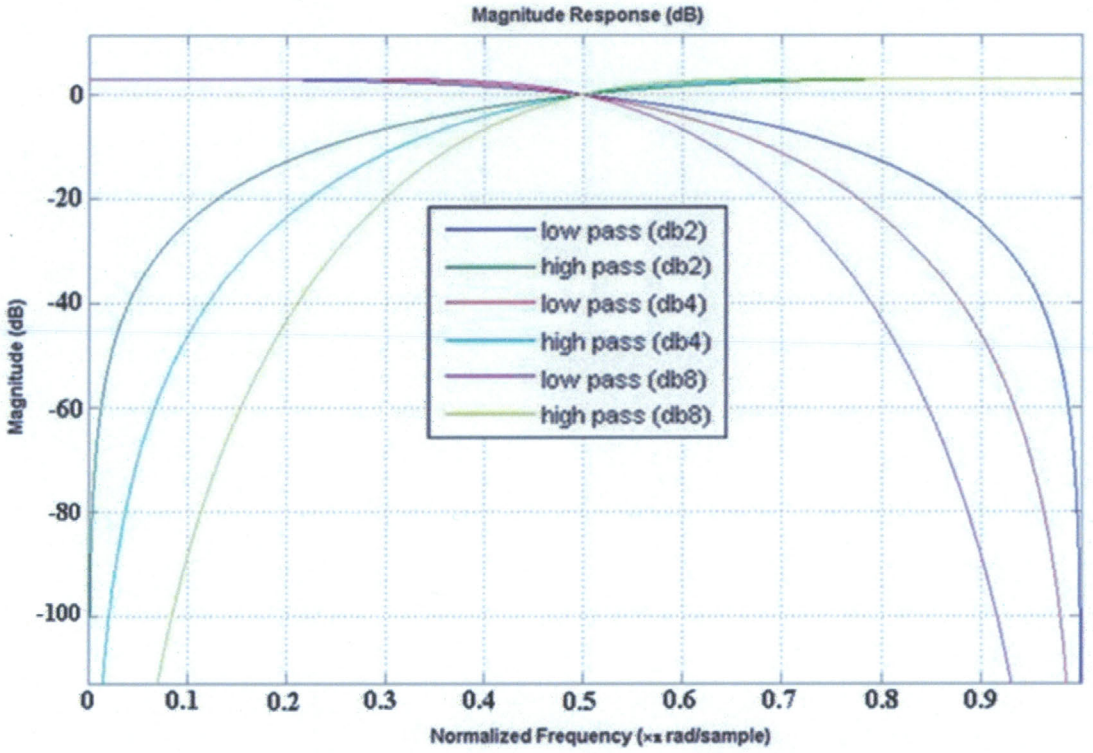

Figure 2.5: Daubechies filters frequency response

Table 2.3: Daubechies wavelet cutoff frequencies

	db2 (πrad/sec)	db4 (πrad/sec)	db8 (πrad/sec)
Low Pass	0.6122	0.5774	0.5535
High Pass	0.3867	0.4227	0.4465

overlapped frequency band between the low and high pass filters decreases. On the other hand, as the filter order increases, the time delay between the input and the output increases. Figure 2.6 shows the decomposition of the current sensor measurements using the Daubechies wavelets. Similar to Figure 2.4, the approximation coefficients reflect low frequency components of the signal and some of the spike energy. The detail coefficients reflect the high frequency components and most of the spike energy.

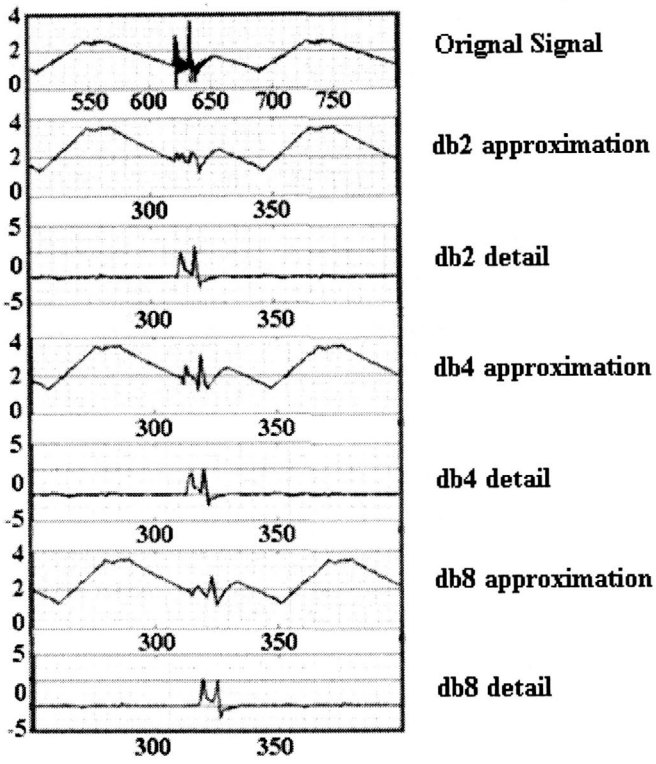

Figure 2.6: Sensor measurements decomposition using Daubechies wavelets

2.2 Modified Wavelet De-noising Technique

[1] has proposed a technique for analyzing the raw neurogram, a nerve signal, using a modified wavelet de-noising technique. The raw neurogram signal, C, with additive noise is decomposed using DWT as shown in Figure 2.7. Approximation coefficients, a, and detail coefficients, d, are the results of the decomposition procedure. Applying thresholding to the detail coefficient prior to reconstruction allows for significant noise reduction. All coefficients with an absolute value greater than the threshold are thought to be part of AP, i.e. spikes, and those below the threshold are presumably derived from noise. The noise coefficients can be set to zero and a noise-free signal can then be reconstructed and used for AP detection. [1] has used the modified wavelet de-noising technique to detect spikes within the raw neurogram signal. Therefore, it is possible to adopt the modified wavelet de-noising technique

to remove spikes from the PEC measurements. Applying a thresholding to the detail coefficients where the spike's coefficients are set to zero, a spike-free signal can then be reconstructed as shown in Figure 2.8. However, as shown in the simulation in Chapter 4, the modified wavelet de-noising technique performance must be enhanced in order to be applied to the PEC measurement. Details of the DWT spike removal algorithm are provided in the next section.

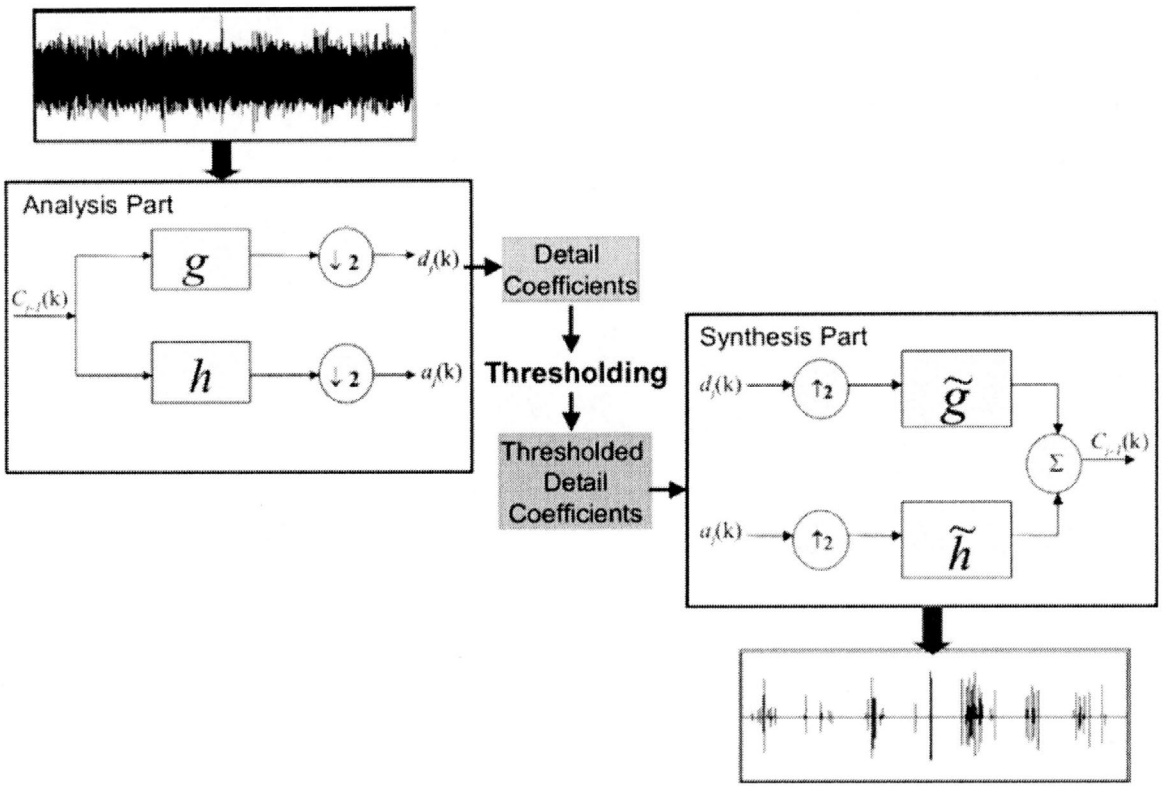

Figure 2.7: Modified wavelet de-noising technique [1]

2.3 Proposed DWT Spike Removal Algorithm

The proposed spike removal algorithm is shown in Figure 2.9. It consists of three stages: i) the decomposing stage where the signal is decomposed into detail coefficients, μ_h, and approximation coefficients, μ_g, using a pair of quadrature mirror

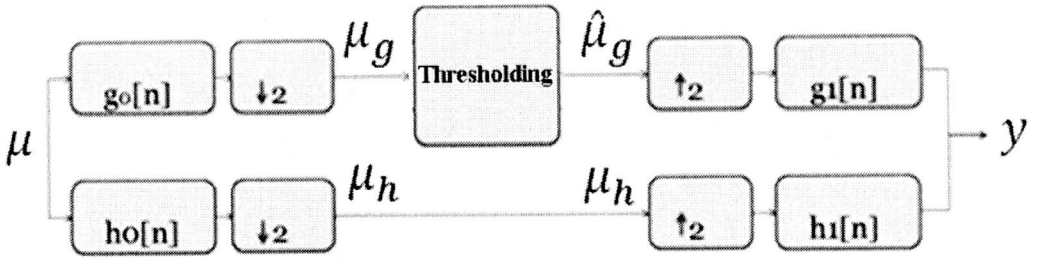

Figure 2.8: Adopted modified wavelet de-noising technique for PEC measurements

filters and two down samplers , ii) the filtering stage where the spike removal occurs, and iii) the signal reconstructing stage consisting of two filters and two up samplers.

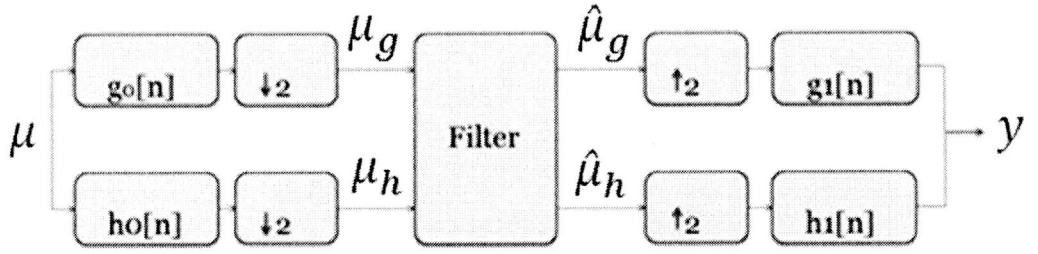

Figure 2.9: Proposed spike removal algorithm

The decomposing stage consists of a high pass filter, $H_0(z)$, and a low pass filter, $G_0(z)$, that are used to generate the detail and approximation coefficients. The filters satisfy Equation 2.1.

In the filtering stage, an algorithm is developed to remove spikes by analyzing the approximation and detail coefficients and changing these if necessary. The conditions that define the presence of a valid spike in the input, μ, are defined independently at each coefficients μ_g and μ_h as follows.

The condition for a spike at μ_g at instant $k = n$ is given by,

$$|\Delta\mu_g(k)| > \alpha, \ for \ k = n, n+1, ..., n+\omega, \ where \ \ \omega \leq \omega_{max} \tag{2.4}$$

The condition for a spike at μ_h at instant $k = n$ is given by,

$$|\mu_h(k)| > \alpha, \ for \ k = n, n+1, ..., n+\omega, \ where \ \omega \leq \omega_{max} \tag{2.5}$$

where α is a selected threshold and ω_{max} is the maximum duration of a spike. If a spike is detected at $\mu_g(k)$ at $k = n$, then

$$\hat{\mu}_g(k) = \mu_g(n-1), \ for \ k = n, n+1, ..., n+\omega \tag{2.6}$$

else

$$\hat{\mu}_g(k) = \mu_g(k) \tag{2.7}$$

and if a spike is detected at $\mu_h(k)$ at $k = n$, then

$$\hat{\mu}_h(k) = 0, \ for \ k = n, n+1, ..., n+\omega \tag{2.8}$$

else

$$\hat{\mu}_h(k) = \mu_h(k) \tag{2.9}$$

The spike-free signal is reconstructed from a pair of quadrature mirror filters and up samplers using the adjusted approximation coefficients, $\hat{\mu}_g(k)$, and adjusted detail coefficients, $\hat{\mu}_h(k)$. The FIR filters in the dyadic filter bank satisfy the conditions Equation 2.2 and Equation 2.3.

Chapter 3

Hardware and Software System Design

3.1 FPGA Realization

An FPGA-based interface board was designed for this research to test the new algorithm within the PEC system. An FPGA realization was considered for the interface board because such a platform facilitates rapid prototyping and system-on-chip design that augments high speed protection schemes and signal processing. Compared with conventional mixed signal systems, the FPGA embedded interface system has the following advantages: i) it enhances the system programmability and flexibility to implement detection algorithms such as phase locked loops (PLLs) and the wavelet transform, ii) it eliminates threshold drifting as the temperature and time changes, iii) it reduces board size, components and numbers of input/output (I/O) ports, and iv) it allows remote control and system upgrading using Ethernet.

The interface board that was designed and developed for this research consists of the following main parts: Xilinx FPGA chip, 16 Mbit Serial Flash Memory, Joint Test Action Group (JTAG) port, Analog to Digital Converters (ADCs), Integrated

Triple Supply Power Regulator, and I/O ports as shown Figure 3.1. The Xilinx FPGA chip which is the main part of the interface board implements real-time protection and detection algorithms. The FPGA chip is a RAM-based device which needs to be configured during powering up, therefore, a 16 Mbit Serial Flash Memory is selected to store the configuration bit file. In order to program the FPGA chip and store the configuration file into the serial flash, a parallel port JTAG is used to communicate between the host PC and the interface board. The ADCs are used to convert the analog sensor measurements to digital data that can be read by the FPGA chip. The Integrated Triple Supply Power Regulator is used to power the FPGA chip, serial flash and the ADC's. The I/O ports are used to communicate with PEC components including the DSP board, driver board and the sensors. The interface board schematic can be found in Appendix B.

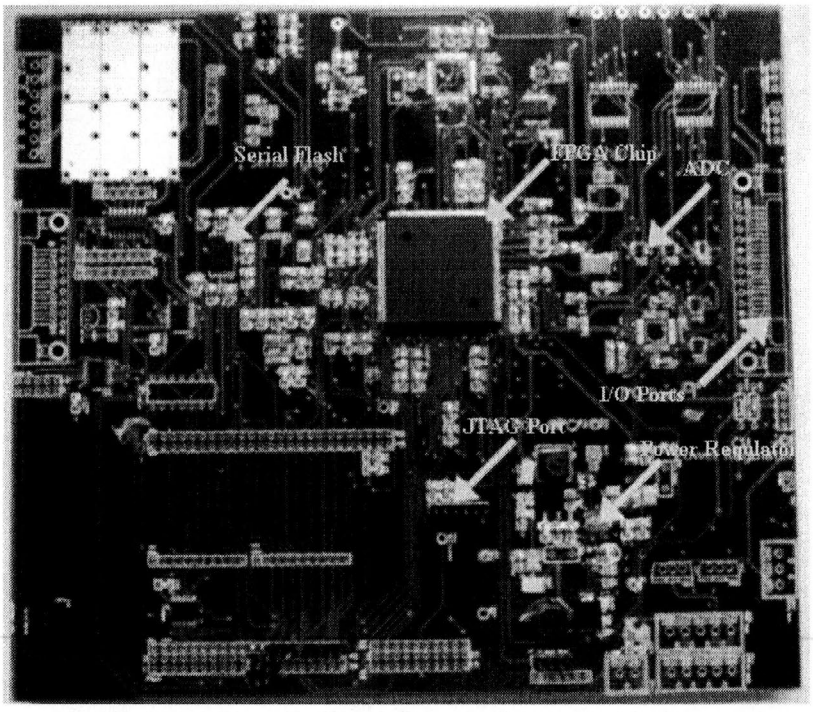

Figure 3.1: FPGA-based interface board

3.2 Xilinx System Generation

System Generator is a high level tool provided by Xilinx to design and implement high performance Digital Signal Processing (DSP) systems using FPGAs [28]. It is a commercial FPGA-based DSP design tool that allows rapid development and prototyping of complex hardware designs. System Generator permits high level system representation using Matlab/Simulink software which is a pervasive design-analysis and simulation tool in the DSP community [29]. The design flow is very useful in closing the gap between the system level design and the physical FPGA implementation. In addition, Xilinx provides specific blocksets and toolboxes that allow the designer to focus on the algorithm implementation rather than the optimization. There are over 90 DSP building blocks including the common building blocks (such as adders, multipliers, multiplexers, registers and other) and complex building blocks (such as filters, memories and FFTs). Figure 3.2 shows some of the Xilinx DSP building blocks.

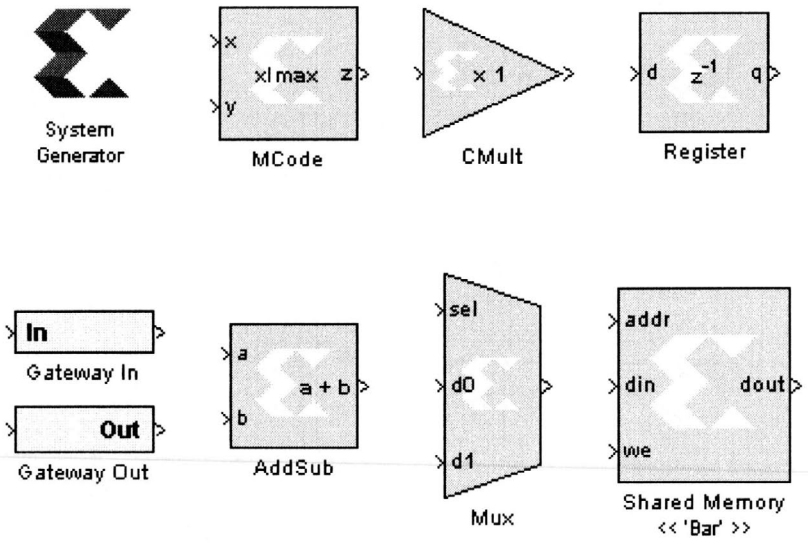

Figure 3.2: Example of Xilinx DSP building blocks

These blocks are accessible via the Simulink Library Browser as shown in Fig-

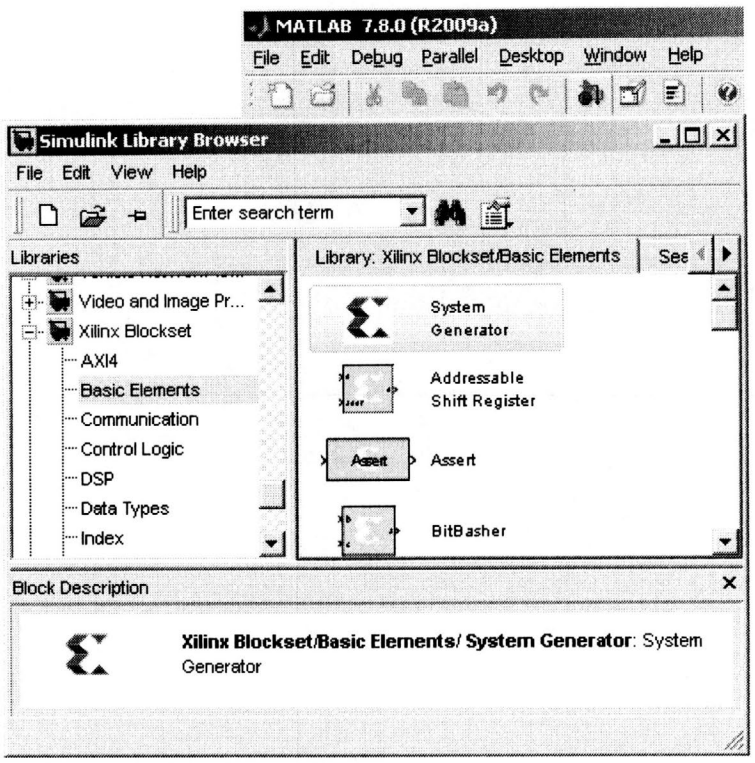

Figure 3.3: Xilinx blockset within the Simulink Library Browser

ure 3.3. The Xilinx blockset is categorized into sub-categories such as Basic Elements, Communications, Control Logic, DSP, Math and others. Double clicking on the block shows its properties window where the block parameters can be specified. The next section contains brief descriptions for some of the blocks that are used in the design and implementation of the spike removal algorithm.

3.2.1 Gateway In/Out Blocks

These two blocks are used to define the FPGA boundary. In the simulation stage, Gateway In converts data type from Simulink integer, double and fixed point into Xilinx fixed point number. While Gateway Out converts a Xilinx fixed point number into an output of type of Simulink integer, double and fixed point. This is useful for input signal generation and output analysis using Matlab. In hardware, they are the top level I/O ports.

3.2.2 System Generator Block

System generator block is required in every system generator design in order to drive the implementation process. In its properties window, the target netlist, the device performance target and the system period can be specified.

3.2.3 MCode Block

This block is capable of executing a user-supplied Matlab function useful in implementing arithmetic functions, finite state machine and control logic. However, MCode block support is limited to a subset of the Matlab language. It has general guidelines [30] such as: i) MCode blocks I/O must be Xilinx fixed-point type, ii) there is at least one output port, and iii) the code for the block and the model that use the block must exist within the same directory.

3.2.4 Interface Board Design

Xilinx system generator is used to design the interface board logic. Figure 3.4 shows the simulink model for the interface board software design. The sensor measurements are sampled using the ADCs at rates of 3 MHz for the measurements of current and 2 MHz for measurements of voltages and temperature . These measurements contain high frequency noises, i.e. spikes, due to the effect of the switching semiconductor devices in the PEC. Therefore, the spike removal filter is included. The spike-free measurements will be used in the implementation of real-time protection and detection algorithms that include over current (OI), over voltage (OV) and under voltage (UV) protection. Details of the design for each module can be found in Appendix A.

Figure 3.5 shows the proposed spike removal algorithm which consists of: i) pair of decomposing filters followed by pair of down samplers (decomposing stage), ii) filtering stage, and iii) pair of up samplers followed by pair of reconstructing filters (signal reconstructing stage).

The MCode block is used to implement the filtering associated with the proposed spike removal algorithm. The source code for this block can be found in the Appendix A. The decomposing and reconstructing filters are designed such that they represent one of the mother wavelets. This can be done by selecting the filter's coefficients from Table 2.1 or Table 2.2. Since the number of coefficients increases with the mother wavelet length, the complexity of the design and the resources required also increase. Figure 3.6 shows the structure of the filters that are used in the decomposing and reconstructing stages. The system generator block is used to generate the bit stream file in order to program the FPGA chip.

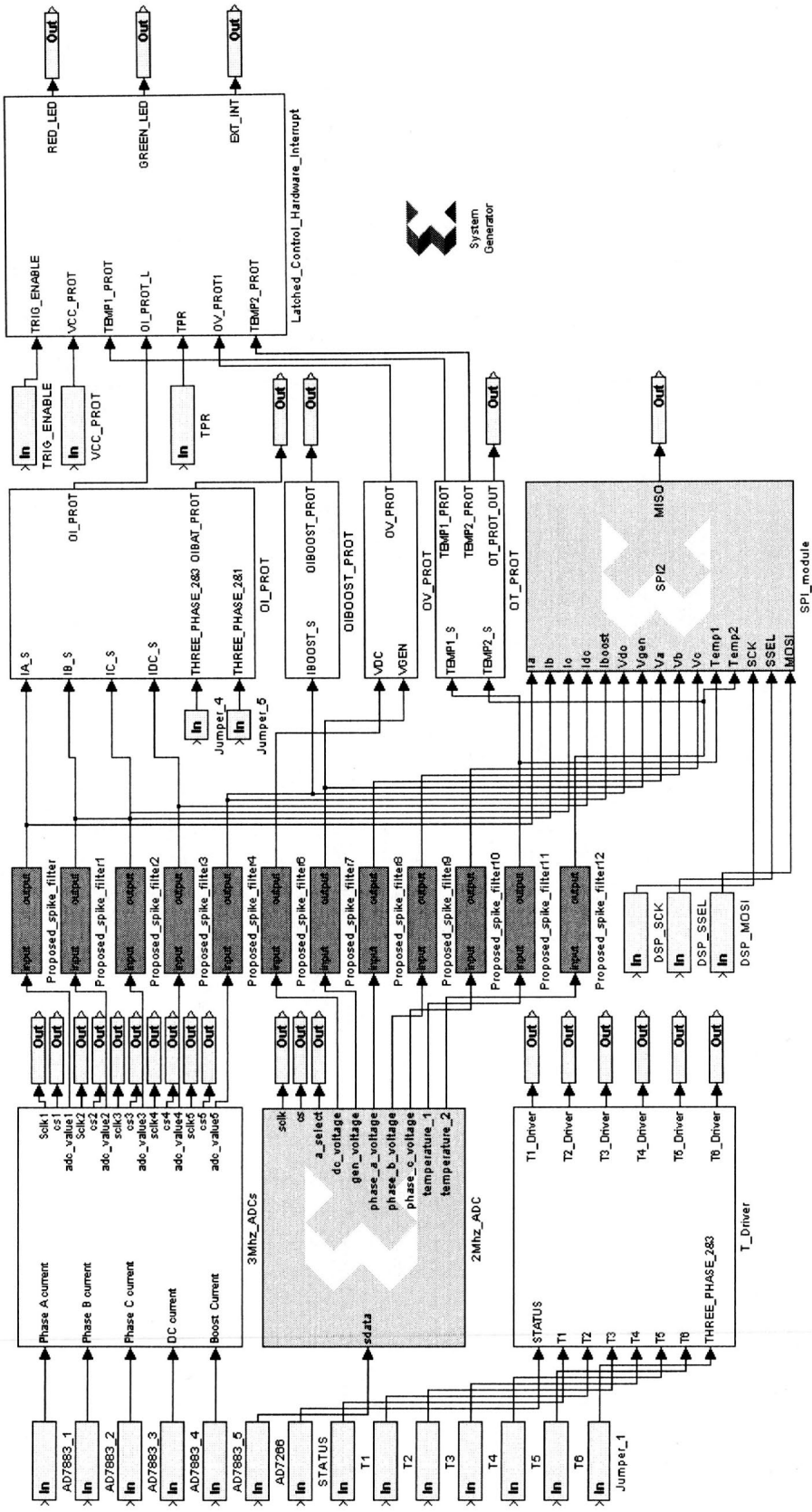

Figure 3.4: Simulink model of the interface board

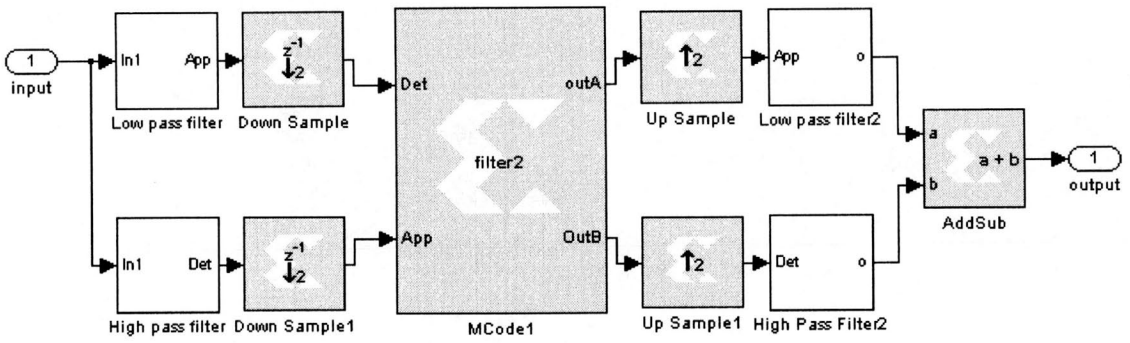

Figure 3.5: FPGA realization for the proposed spike removal algorithm

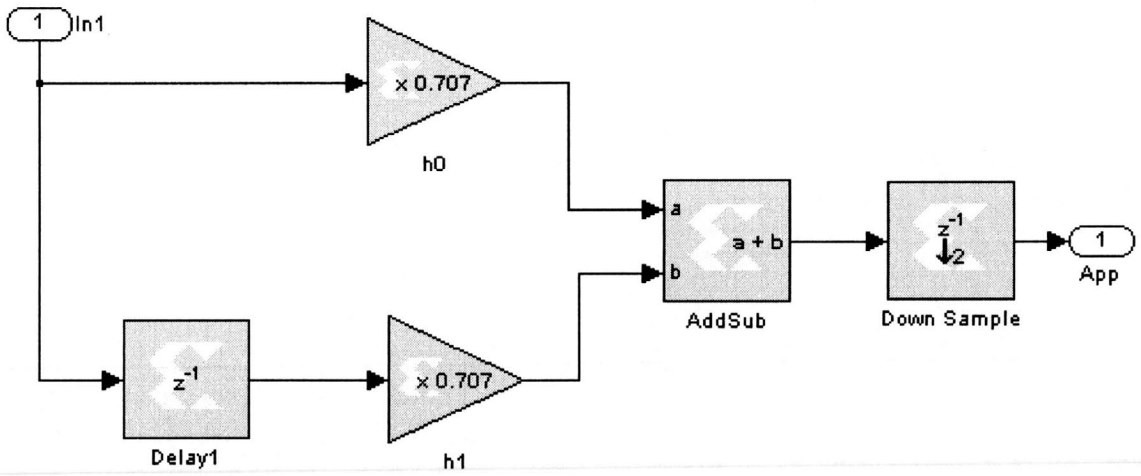

Figure 3.6: Filter structure for the implementation of the DWT

Chapter 4

Simulation

In order to evaluate the performance of the proposed algorithm and the PEC fault protection system, simulations were conducted for the modified wavelet denoising technique and the proposed spike removal algorithm. The Haar and Daubechies mother wavelets were tested. Then, the PEC fault protection system with the proposed spike removal algorithm was simulated.

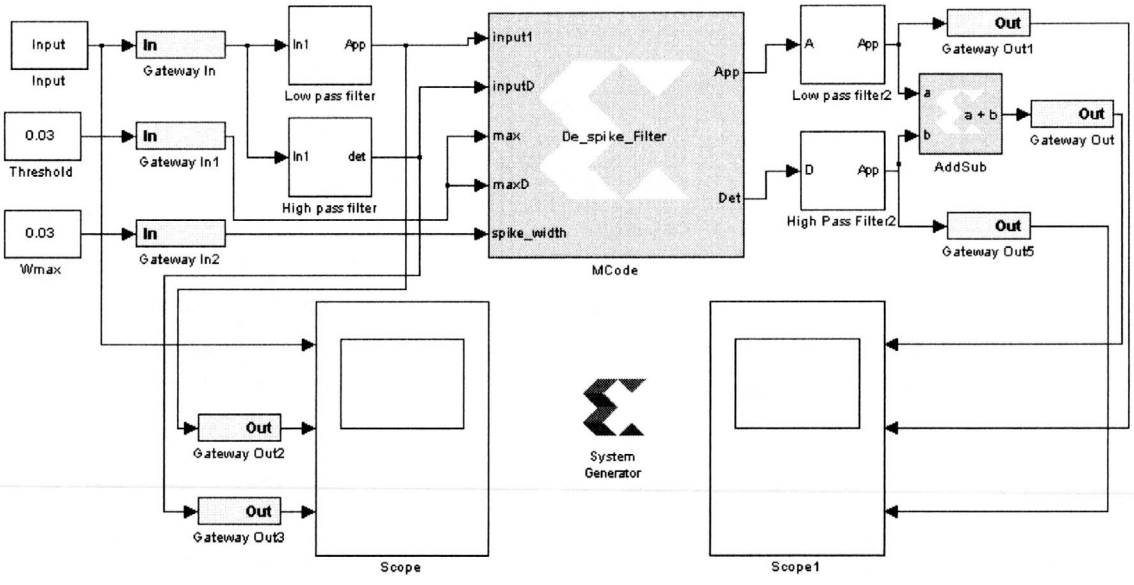

Figure 4.1: Simulink model used in the simulation

Figure 4.1 shows the Simulink model used for this simulation. Note that the

decomposing and the reconstructing filters depend on the type of wavelet used. The input signal that was used in the simulation is shown in Figure 4.2. It consists of two valid spikes that satisfy Equation 2.4 and Equation 2.5 separated by a wider spike that does not satisfy either equations. The parameters of a valid spike were chosen as: $\alpha = 0.03$, $\omega_{max} = 3$.

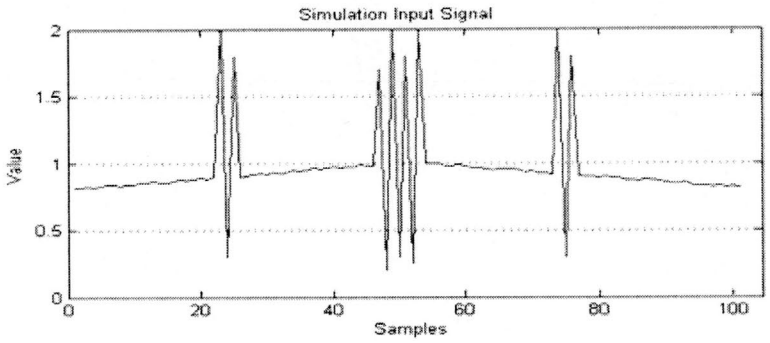

Figure 4.2: Simulation input signal

4.1 Modified Wavelet De-noising Technique

Two types of wavelets were simulated: the Haar wavelet and the Daubechies wavelet.

4.1.1 Haar Wavelet Simulation

In this section, the simulation of the modified wavelet de-noising technique using the Haar mother wavelet is presented. Figure 4.3 shows the approximation and the detail coefficients before and after filtering. Figure 4.4 shows the input signal and the output signal of the modified wavelet de-noising technique using the Haar wavelet.

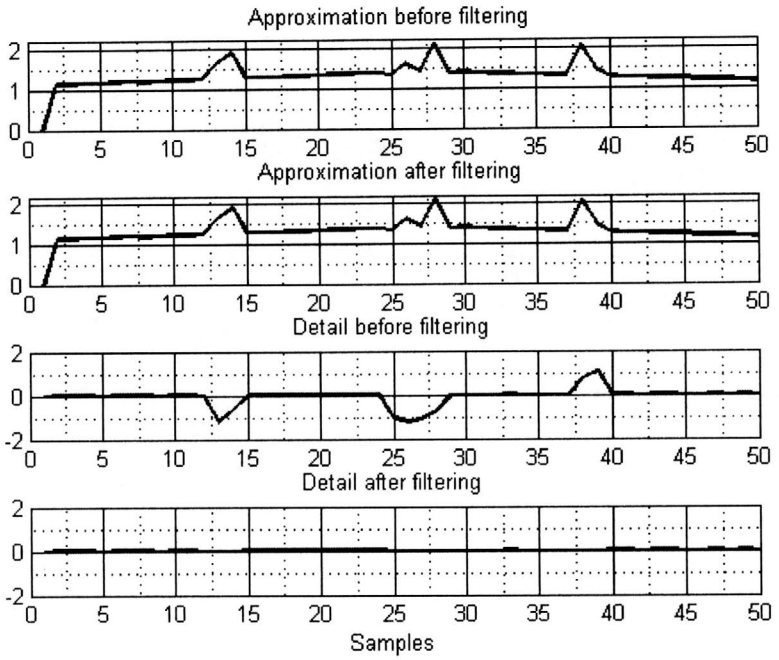

Figure 4.3: Approximation and detail coefficients of the input signal using the Haar wavelet.

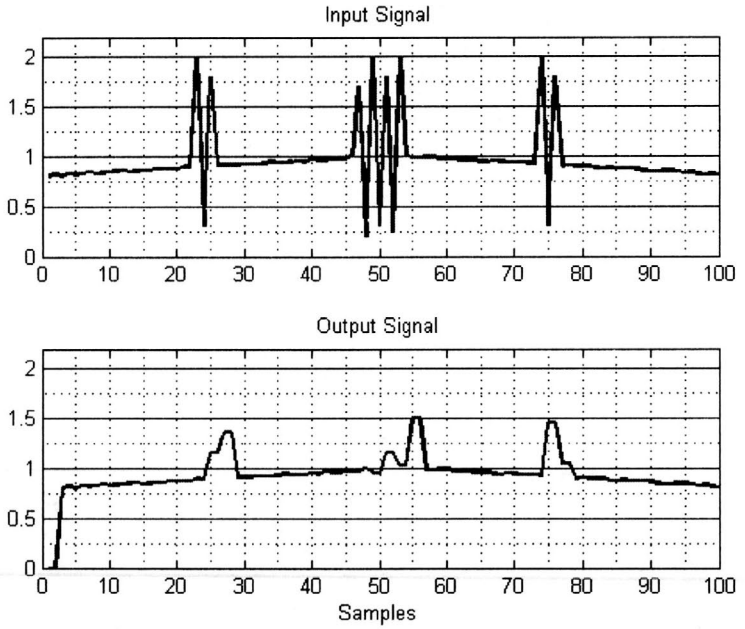

Figure 4.4: Simulation results for the modified wavelet de-noising technique using the Haar wavelet.

4.1.2 Daubechies Wavelets Simulation

In this section, the simulation of the modified wavelet de-noising technique using the Daubechies mother wavelets is presented. Three different orders of the Daubechies mother wavelets were chosen: the second order Daubechie (db2), the fourth order daubechie (db4), and the eighth order daubechie (db8). Figure 4.5 shows the approximation and the detail coefficients before and after filtering and Figure 4.6 shows the input and output signal of the modified wavelet de-noising technique using db2 wavelet. Note that the output signal contains some degree of spikes.

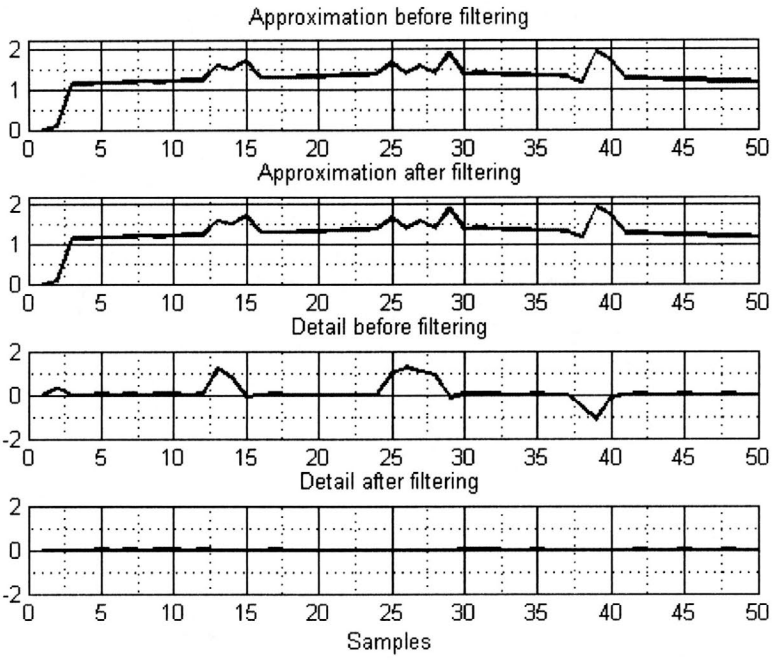

Figure 4.5: Approximation and detail coefficients of the input signal using db2 wavelet.

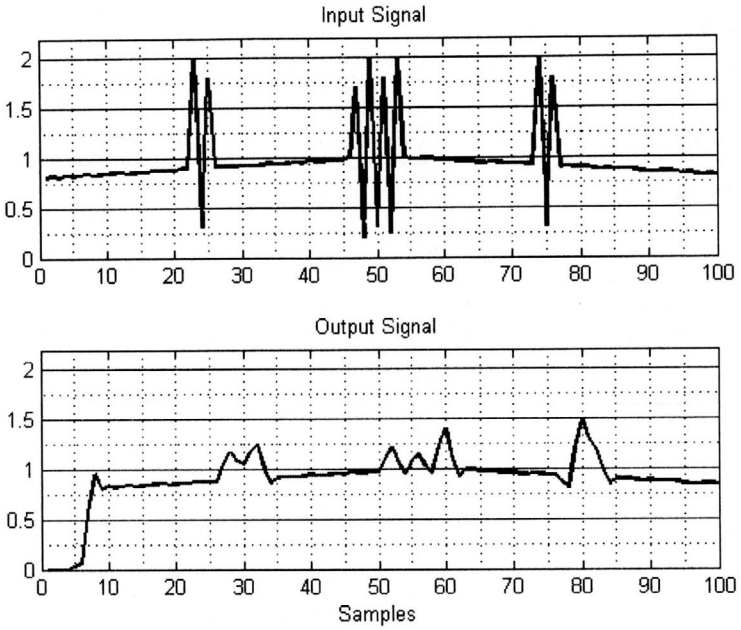

Figure 4.6: Simulation results for the modified wavelet de-noising technique using db2 wavelet.

Figure 4.7 and Figure 4.8 show the simulation results of the modified wavelet de-noising technique using the db4 wavelet. Figure 4.9 and Figure 4.10 show the simulation results of the modified wavelet de-noising technique using the db8 wavelet. Note that the output signal contains some spikes.

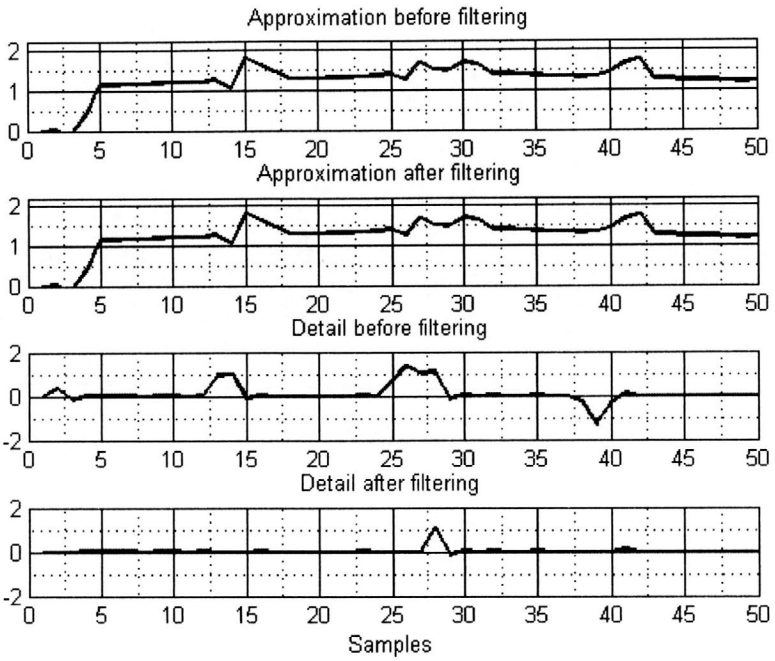

Figure 4.7: Approximation and detail coefficients of the input signal using the db4 wavelet.

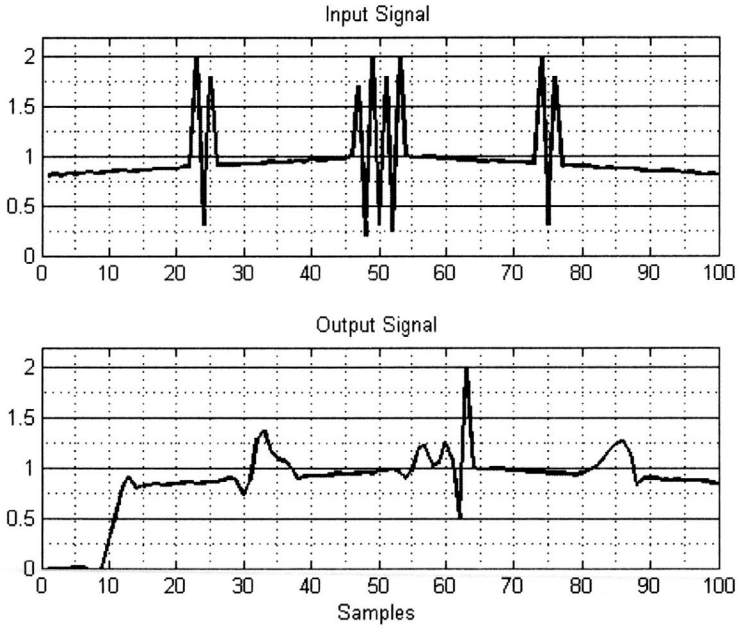

Figure 4.8: Simulation results for the modified wavelet de-noising technique using the db4 wavelet.

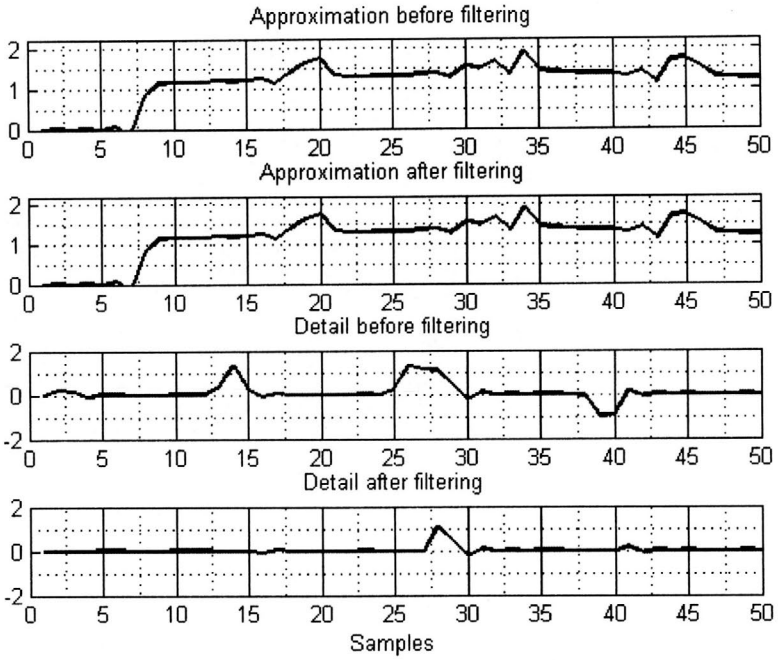

Figure 4.9: Approximation and detail coefficients of the input signal using the db8 wavelet.

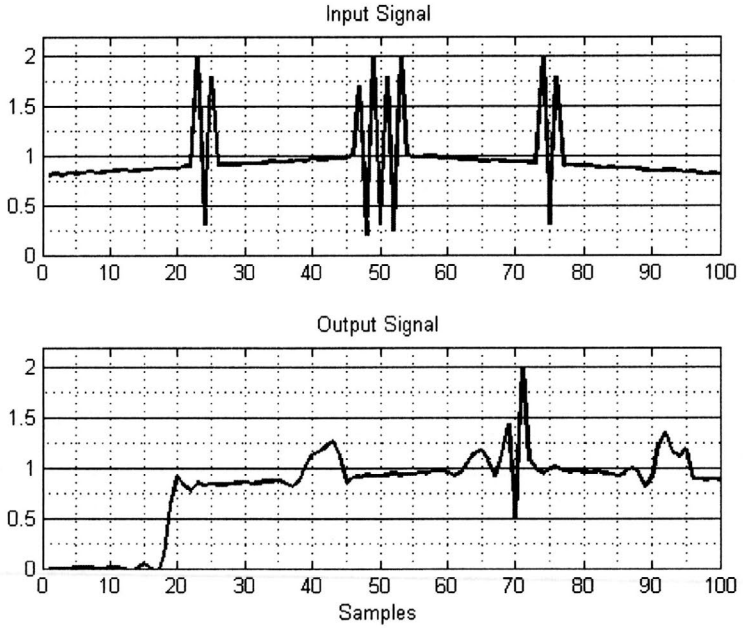

Figure 4.10: Simulation results for the modified wavelet de-noising technique using the db8 wavelet.

4.2 Proposed Spike Removal Algorithm

This section of simulation shows the proposed spike removal algorithm testing. Similar to the modified wavelet de-noising technique simulation, the Haar and the Daubechies wavelets were tested.

4.2.1 Haar Wavelet Simulation

In this section, the simulation of the spike removal algorithm using the Haar mother wavelet is presented. Figure 4.11 shows the approximation and detail coefficients before and after filtering. Figure 4.12 shows the input signal and output signal of the proposed spike removal algorithm using Haar wavelet. Note that the output signal does not contain any valid spikes.

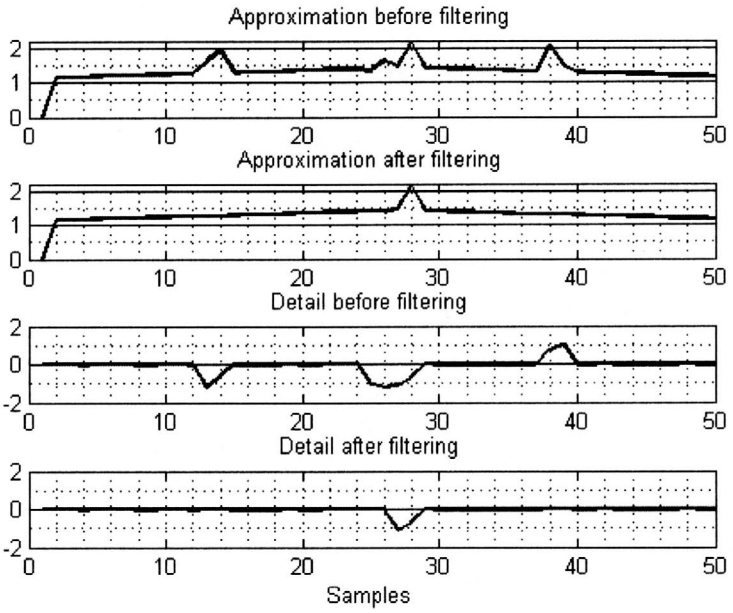

Figure 4.11: Approximation and detail coefficients of the input signal using the Haar wavelet.

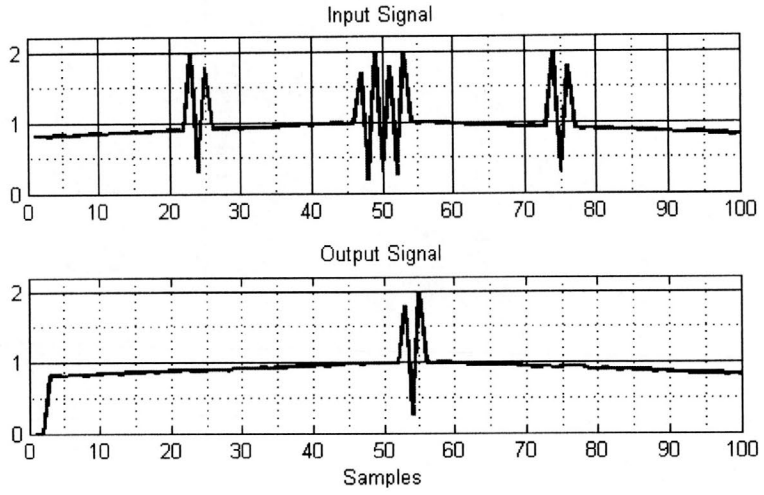

Figure 4.12: Simulation results for the spike removal algorithm using the Haar wavelet.

4.2.2 Daubechies Wavelets Simulation

In this section, the simulation of the spike removal algorithm using the Daubechies mother wavelets is presented. Three different orders of the Daubechies mother wavelets were chosen: db2, db4, and db8. Figure 4.13 shows the approximation and the detail coefficients before and after filtering and Figure 4.14 shows the input and output signals of the proposed spike removal algorithm using db2 wavelet. Figure 4.15 shows the approximation and the detail coefficients before and after filtering and Figure 4.16 shows the input and output signals of the proposed spike removal algorithm using the db4 wavelet. Figure 4.17 and Figure 4.18 show simulation results of the proposed spike removal algorithm using the db8 wavelet.

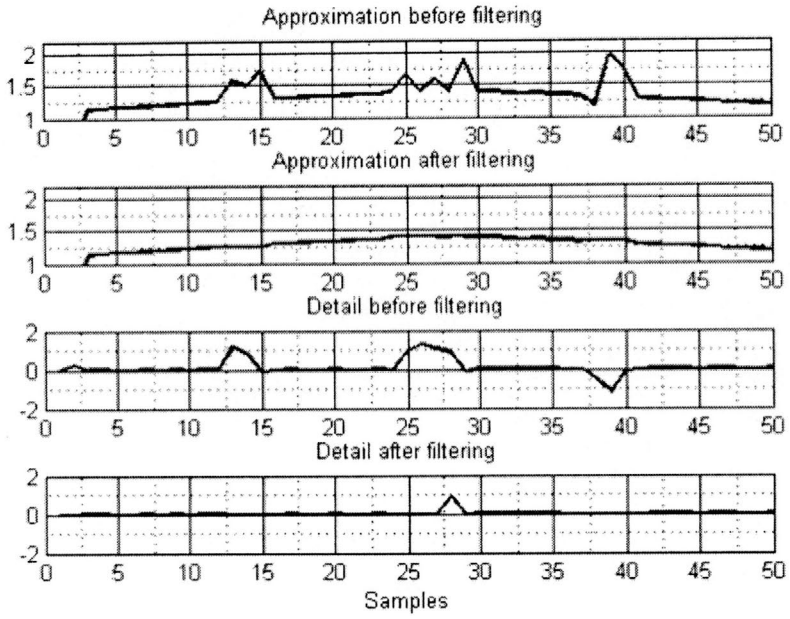

Figure 4.13: Approximation and detail coefficients of the input signal using the db2 wavelet.

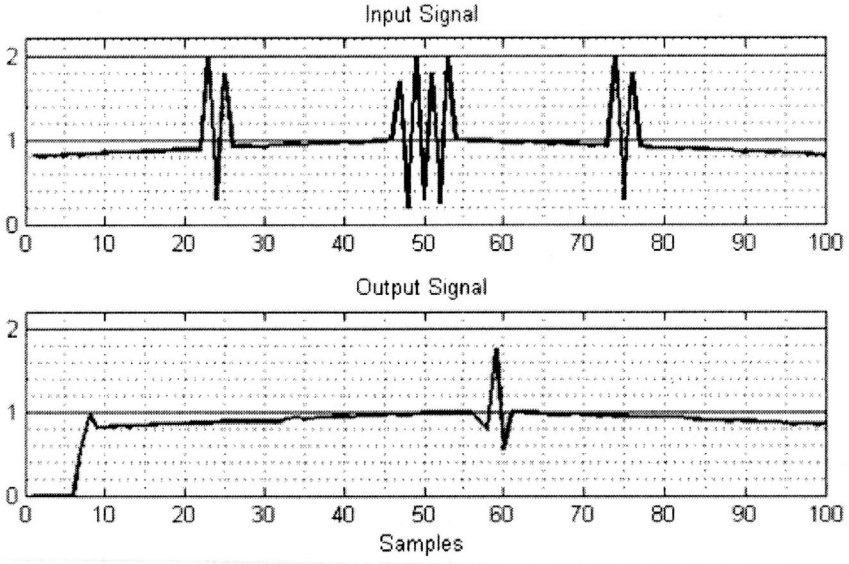

Figure 4.14: Simulation results for the spike removal algorithm using the db2 wavelet.

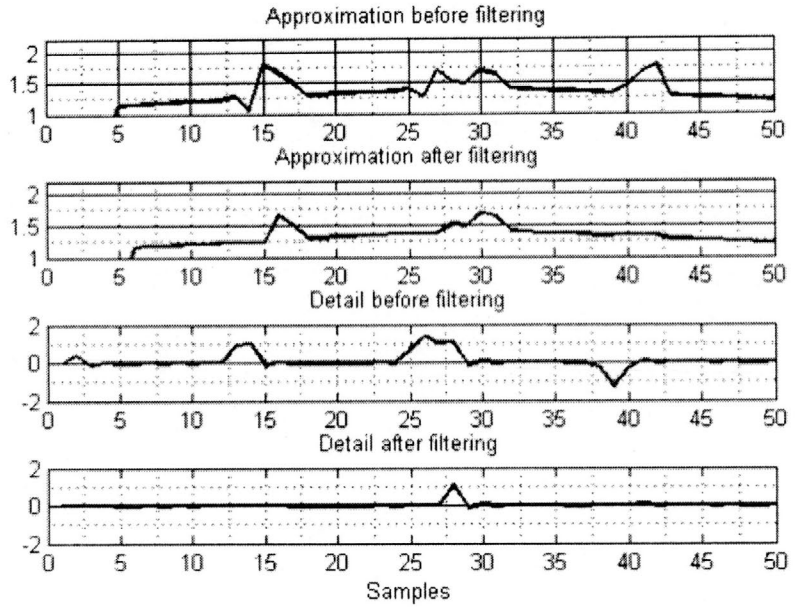

Figure 4.15: Approximation and detail coefficients of the input signal using the db4 wavelet.

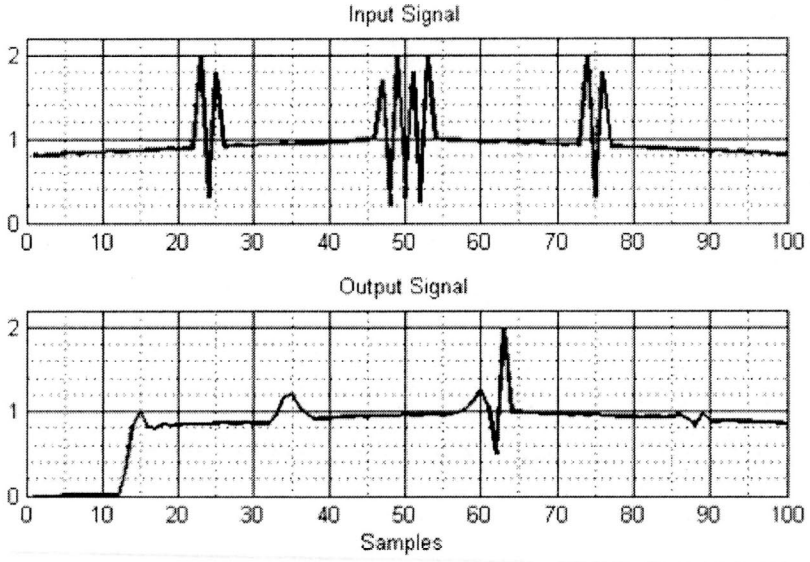

Figure 4.16: Simulation results for the spike removal algorithm using the db4 wavelet.

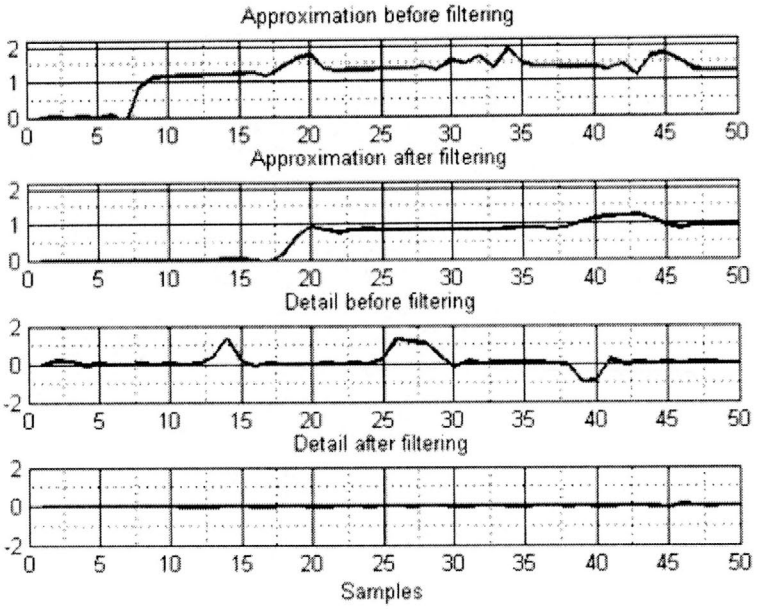

Figure 4.17: Approximation and detail coefficients of the input signal using the db8 wavelet.

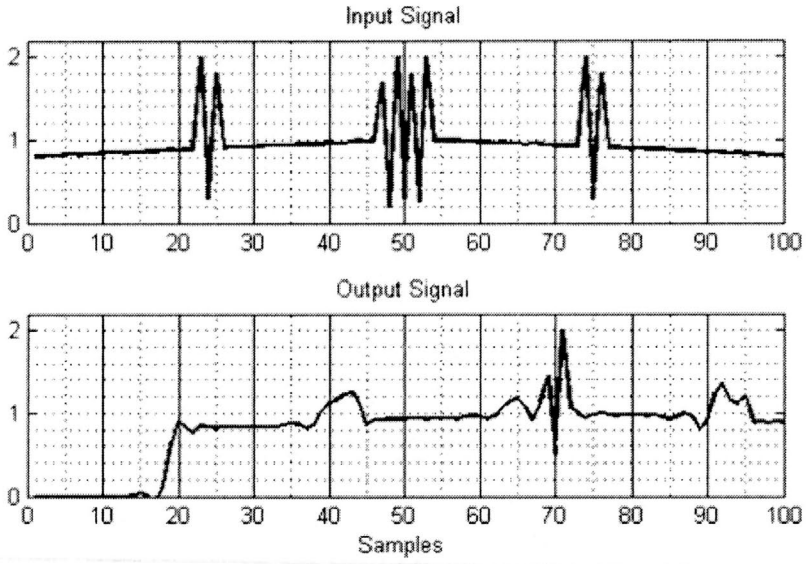

Figure 4.18: Simulation results for the spike removal algorithm using the db8 wavelet.

4.3 Current Fault Detection Simulation

In this section, the detection of a fault in electric current was simulated under two conditions: no fault condition and fault condition. The model that was used for the current fault simulation is shown in Figure 4.19. The fault detection system consists of the spike removal algorithm and IO PROT module. The spike removal algorithm is used to remove the short time spikes from the input signals and the IO PROT is used to detect if the input signals are within normal operating limits, i.e. POS Limit and NEG Limit. In this simulation, the parameters of a valid spike, i.e. spike to be removed, were chosen as: $\alpha = 0.03$, $\omega_{max} = 4$, the POS Limit, which corresponds to the maximum measurement, was set to be 1.2, and the NEG Limit, which corresponds to the minimum measurement, was set to be -1.2. Refer to Appendix A for the details of IO PROT module.

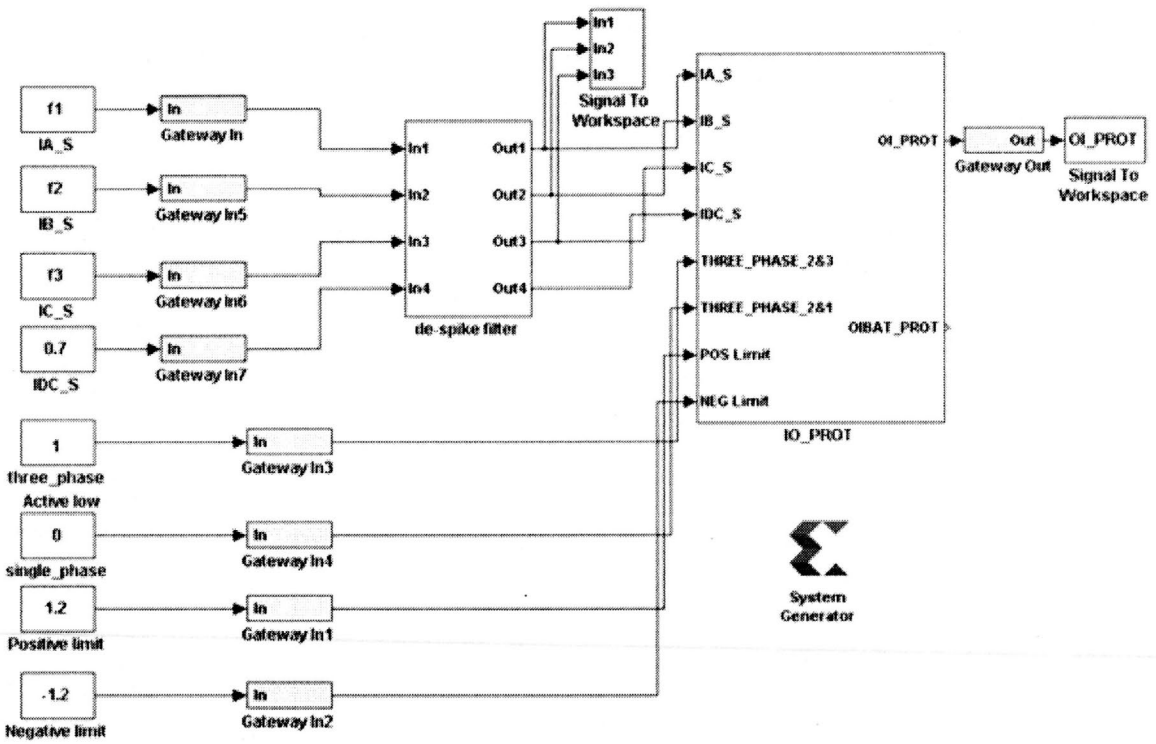

Figure 4.19: Simulink module for simulating the detection of current faults.

4.3.1 No Fault Condition With Noise Spikes

The no fault condition was simulated using the sensor measurements shown in the upper plot of Figure 4.20. Note that the measurements contain valid spikes which represent the spikes that are induced due to the rapidly switching of the power semiconductor devices in the PEC. The middle plot shows the reconstructed signals and the bottom plot shows the output of the detection logic indicating (logic 1 = no fault).

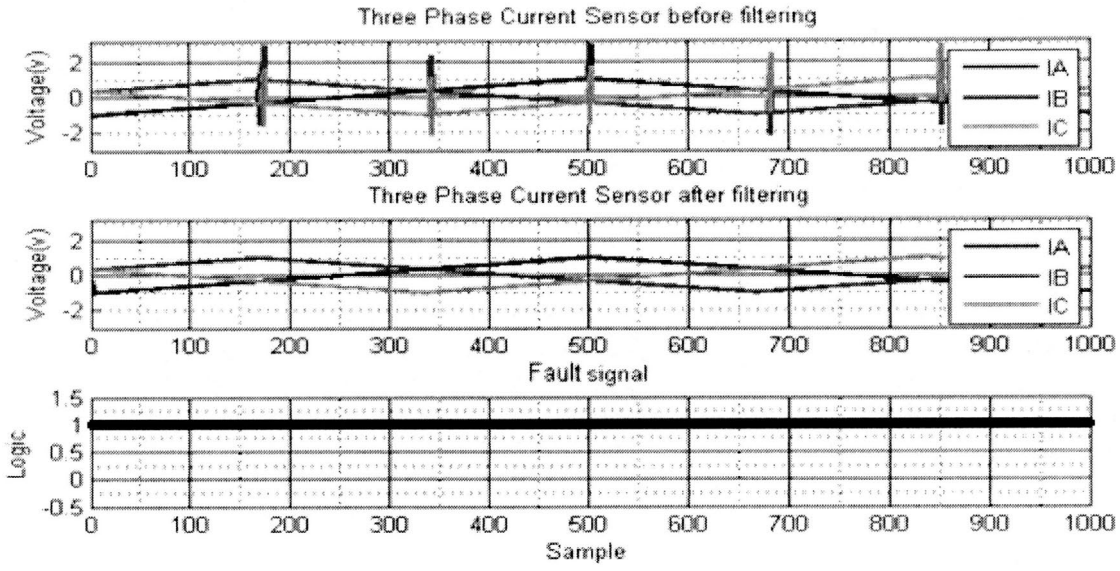

Figure 4.20: Simulation results of the spike removal algorithm: 3-phase current measurements (top), reconstructed measurements with spikes removed (middle) and protection circuits trigger (bottom).

4.3.2 Fault Condition With Noise Spikes

The fault condition was simulated using the sensor measurements shown in the upper plot of Figure 4.21. Note that the measurements not only contain valid spikes but also an actual fault near sample 600 of Phase-A. The middle plot shows the reconstructed signals with the noise spikes removed. The transient resulting from an

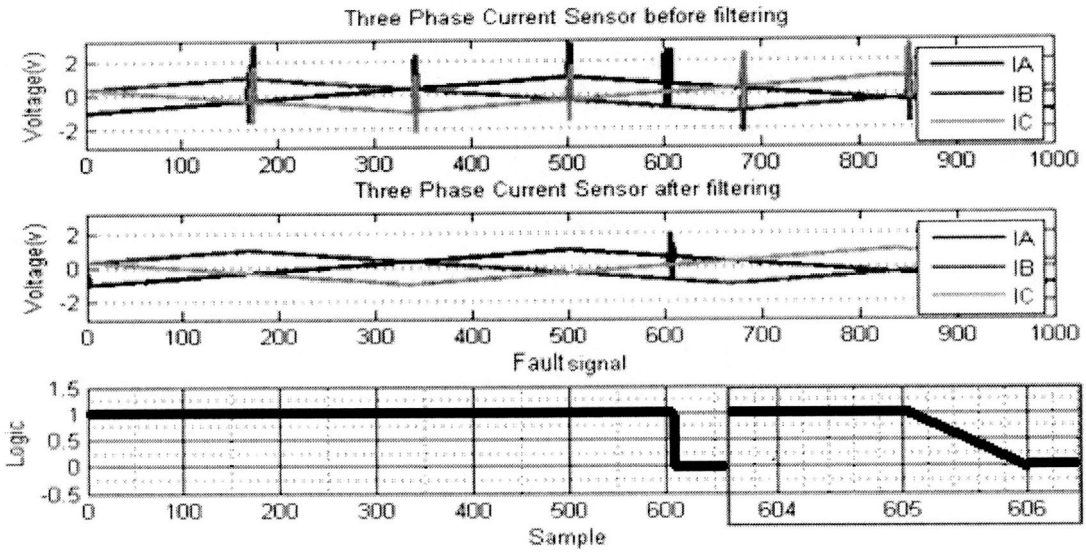

Figure 4.21: Simulation results of the spike removal algorithm when a fault occurs: 3-phase current measurements (top), reconstructed measurements with spikes removed (middle) and protection circuits trigger (bottom).

actual fault is not removed. The bottom plot show that the fault protection circuitry is triggered near sample 600.

Chapter 5

Implementation

The proposed algorithm was implemented using the Xilinx Spartan-3E XC3S500E FPGA and 12-BIT 2-MSPS ADS7883 ADC. Figure 5.1 shows the testing structure for implementing the proposed algorithm.

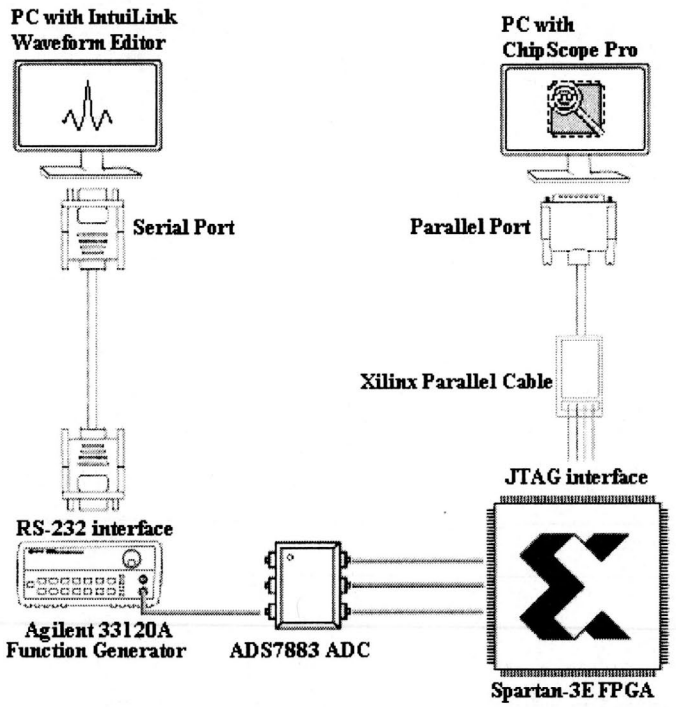

Figure 5.1: Implementation structure.

To test the proposed algorithm and the FPGA behavior within the system,

ChipScope Pro Integrated Logic Analyzer (ILA) was used to perform real time system analysis and assessment of the implementation of the proposed algorithm and protection system. ChipScope Pro ILA is capable of capturing the waveforms within the Xilinx FPGA and analyzing their timing and functionality [31]. The Xilinx Parallel Cable was used to interface the Spartan-3E FPGA to the host Personal Computer (PC) with the ChipScope Pro ILA. The Xilinx Parallel Cable communicates with the Spartan-3E FPGA through the Joint Test Action Group (JTAG) interface and it communicates with the host PC through the Parallel port. The IntuiLink Waveform Editor (IWE) application was used to construct replicas of the real PEC signals to be used in the testing of the hardware implementation. A host PC with the IWE application were used to program an Agilent 33120A function generator with the replica signals using a serial cable. The replica signals are applied as input to the Spartan-3E FPGA for testing and assessment.

5.1 Proposed Spike Removal Implementation

The spikes of the replica signals were chosen; some that satisfy the conditions of Equation 2.4 and Equation 2.5 (i.e. valid spikes) and some that do not satisfy the two conditions and should therefore not be removed. The replica signals were applied as inputs to the FPGA for spike removal. The Haar wavelet was selected to implement the spike removal algorithm as explained in Chapter 6. The implementation results are shown in Figure 5.2. The input signal consists of two valid spikes separated by a wider spike that does not satisfy Equation 2.4 and Equation 2.5. The parameters of a valid spike were chosen as: $\alpha = 0.03$, $\omega_{max} = 4$. The Figure 5.2 shows that the spike removal filter is capable of removing valid spikes from the measurements while keeping a portion of the wider spike.

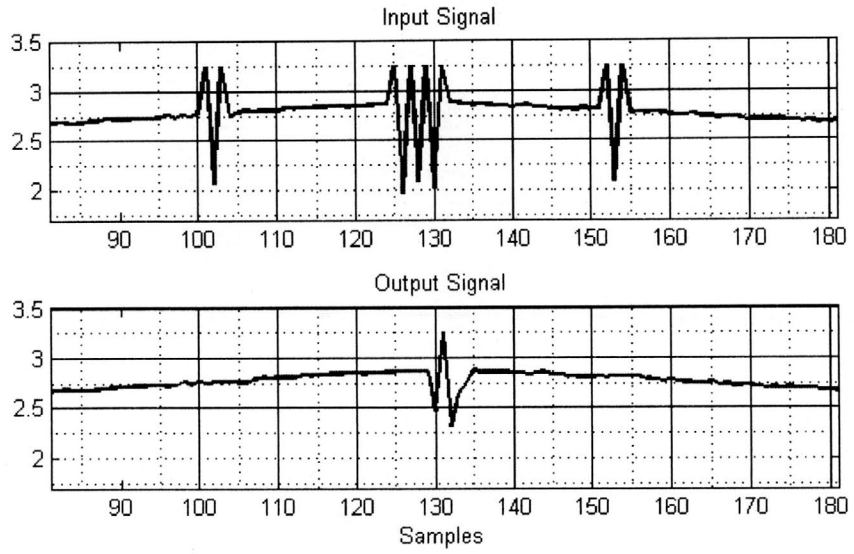

Figure 5.2: Implementation results for the spike removal using Haar DWT.

5.2 Fault Detection Implementation

In this section, the fault detection was implemented under two conditions: no fault condition and fault condition. The algorithm was tested using an FPGA implementation of the spike removal filter when supplied with a replica of the actual measurements of single-phase current as shown in Figure 5.3 and Figure 5.4. The upper plot of Figure 5.3 is a measurement record of the single-phase current that contains noise spikes near samples 32, 65, 98, 132 and 165. The middle plot shows the reconstructed signals and the bottom plot indicates the absence of a fault (logic 1 = no fault). In Figure 5.4, the noise spikes appear near samples 34, 67, 100, 134, and 167. There is an actual fault near sample 120. The middle plot shows the reconstructed signals with the noise spikes removed. The transient resulting from an actual fault is not removed. The bottom plot shows that the fault protection circuitry is triggered near sample 120.

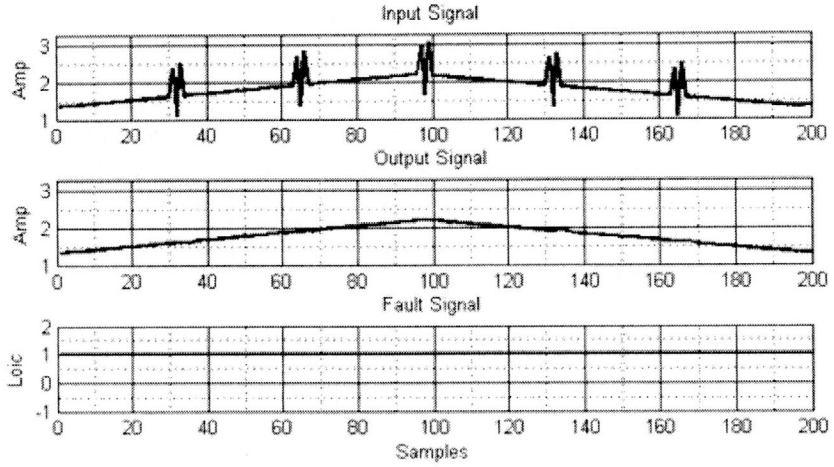

Figure 5.3: Results of the FPGA realization of the spike removal filter: (1-phase current measurements (top), reconstructed measurements with spikes removed (middle) and protection circuits trigger (bottom).

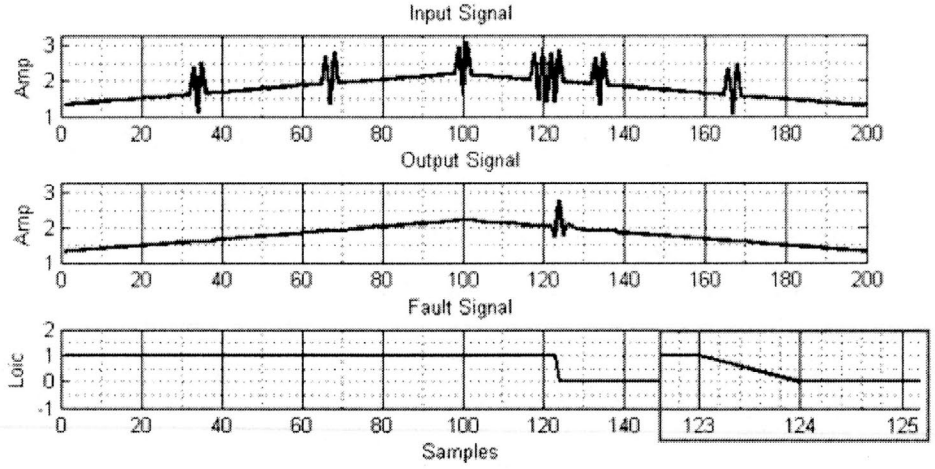

Figure 5.4: Results of the FPGA realization of the spike removal filter when a fault occurs: (1-phase current measurements (top), reconstructed measurements with spikes removed (middle) and protection circuits trigger (bottom).

Chapter 6

Discussion

The spike removal algorithm is simulated to test its performance and compare it with the modified wavelet de-noising technique. Varieties of test spikes were chosen: some that satisfy the conditions of Equation 2.4 and Equation 2.5 and must be removed from the measurements, i.e. valid spikes, and others that do not satisfy the two conditions and should therefore not be removed. The test inputs were applied to simulink models of the modified wavelet de-noising technique and the spike removal filter for two cases: i) Haar DWT and ii) Daubechies DWT. The simulation results are shown in Figures 4.3 to 4.18. The results show that the modified wavelet de-noising technique is not capable of removing spikes completely as shown in Figure 4.4, Figure 4.6, Figure 4.8, and Figure 4.10. This is because the technique uses thresholding on the detail coefficients without considering the approximation coefficients. As shown in Figure 4.3, Figure 4.5, Figure 4.7, and Figure 4.9, the approximation coefficients contain part of the spikes. Therefore, it is not enough to threshold the detail coefficients to completely remove the spikes, specially when one level of DWT is used. It is possible to enhance the performance of the modified wavelet de-noising technique by using multiple levels of DWT. However, multiple levels of DWT is costly in term of the resources needed and time delay.

The spike removal algorithm is proposed where the signal is decomposed using one level of DWT and spikes are detected and removed from both detail and approximation coefficients. Simulation results in Figure 4.12 and Figure 4.14 show that the spike removal algorithm using Haar or db2 wavelets is capable of removing valid spikes from the measurements while keeping a portion of the wider spike, i.e. actual fault. However, Figure 4.16 and Figure 4.18 show that the spike removal algorithm performance decreases as the order of the wavelet used increases. As the mother wavelet order increases, the approximation coefficients become smoother as shown in Figure 4.11, Figure 4.13, Figure 4.15, and Figure 4.17. Therefore, spikes within the approximation coefficients will not be detected. The spike removal algorithm using the Haar wavelet is chosen to simulate and implement the protection algorithm of the PEC system because the Haar wavelet has better performance when compared to the db4, and db8 wavelets and it has less time delay between the input and output than the db2 wavelet. The time delay in the Haar wavelet is two samples while it is four samples in the db2 wavelet. The PEC protection algorithm simulation results are shown in Figure 4.20 and Figure 4.21. Figure 4.20 shows the no fault condition where the measurements in the upper plot contain noise spikes near samples 170, 340, 500, 680 and 850. The middle plot shows the reconstructed signals and the bottom plot indicates the absence of a fault (logic 1 = no fault). Figure 4.21 shows the fault condition where the measurements contain an actual fault near sample 600 along with noise spikes. The middle plot shows the reconstructed signals with the noise spikes removed. The transient resulting from an actual fault is not removed. The bottom plot show that the fault protection circuitry is triggered near sample 600.

Figure 5.1 shows the implementation testing structure of the proposed spike removal algorithm. Figure 5.2 shows that the spike removal algorithm using the Haar wavelet is capable of removing the valid spikes from the measurements while keeping a portion of the wider spike, i.e. actual fault. Figure 5.3 and Figure 5.4 show the

results of the PEC protection algorithm implementation. The implementation shows that the algorithm is capable of removing the measurements noise and responding to an actual fault within six samples of its occurrence. Using 3 MHz ADC, the time delay in the PEC response to an actual fault is about 2.0 μs. The desired time delay of current fault detection is less than 10 μs in order to protect the PEC system.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The thesis provides an innovation and effective algorithm for the detection and removal of noise spikes in the sensor measurements of PECs. This algorithm will replace the common heuristic approach where the fault protection triggers are disabled over intervals when noise spikes are expected to occur. The algorithm is using DWT to generate a feature space where the spike feature is detected and removed, and the inverse DWT to reconstruct spike-free measurements. DWT is commonly used in biomedical applications such as spike detection of nerves activates and Electromyography (EMG) signals processing. The significance of the proposed algorithm is adopting the DWT de-noising technique for spike detection to remove spikes in the sensor measurements of PECs. The presence of noise spikes compromises the performance and leads to unnecessary triggering of fault protection circuitry as a result of false detection of over/under voltage and current faults. The proposed approach removes spikes from sensor measurements resulting in i) the elimination of false triggering of protection circuits and ii) improved PEC performance because noise free measurements are fed back to the PEC controller. This thesis shows a number of

simulation and experimental test cases that verify the spike removal algorithm performance and compares it with the DWT de-noising technique proposed by [1] to detect spikes within nerve signals. In addition to that, the spike removal algorithm is implemented and tested using the FPGA interface board that was developed for this research. The implementation shows the capability of the spike removal algorithm in removing spikes while maintaining the quality of the PEC's sensor measurements in both low frequency and high frequency ranges.

7.2 Major Contribution

The spike removal algorithm was successfully tested and verified with the FPGA interface board. This algorithm is capable of removing spikes while maintaining the fidelity of the PECs sensor measurements in both low and high frequency ranges. Integrating the spike removal algorithm in the PEC system results in i) the elimination of false triggering of protection circuits and ii) improved PEC performance because noise-free measurements are fed back to the PEC controller. The digital system design for the interface board was developed and the bit stream was generated using Xilinx system generator. The first version of the interface board was designed and manufactured. However, during testing in the lab, several bugs were discovered. Therefore, some modifications were done to the original design. The second version of the interface board is currently under manufacturing.

7.3 Future Work

The second version of the interface board needs to be tested in the lab to verify the hardware design. Then, the FPGA has to be programmed with the bit stream file that was generated by Xilinx system generator. Then, the new interface board can be integrated into the PEC system.

Bibliography

[1] A. Diedrich, W. Charoensuk, R. Brychta, A. Ertl, and R. Shiavi, "Analysis of raw microneurographic recordings based on wavelet de-noising technique and classification algorithm: wavelet analysis in microneurography," *Biomedical Engineering, IEEE Transactions on*, vol. 50, pp. 41 –50, jan. 2003.

[2] H. Ye, *The control technology of wind energy conversion systems.* Beijing: mechanical Industries Publishing Company, 2002.

[3] Z. an Wang, *Power electronic technology.* Beijing: Mechanical Industry Press, 2000.

[4] M. B. Farriz, A. N. Azmi, N. A. M. Said, A. Ahmad, and K. A. Baharin, "A study on the wind as a potential of renewable energy sources in Malaysia," in *Proc. Int Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON) Conf*, pp. 651–655, 2010.

[5] Z. Chen and J. M. Guerrero, "Editorial Special Issue on Power Electronics for Wind Energy Conversion," *Power Electronics, IEEE Transactions on*, vol. 23, pp. 1038 –1040, may 2008.

[6] R. Shukla, R. Tripathi, and S. Gupta, "Power electronics applications in wind energy conversion system: A review," in *Power, Control and Embedded Systems (ICPCES), 2010 International Conference on*, pp. 1 –6, 29 2010-dec. 1 2010.

[7] E. O. Sheybani, O. J. Mengshoel, and S. Poll, "Removing spikes while preserving data and noise using wavelet filter banks," in *Proc. IEEE Aerospace Conf*, pp. 1–10, 2010.

[8] D. Feuerstein, K. H. Parker, and M. G. Boutelle, "Practical Methods for Noise Removal: Applications to Spikes, Nonstationary Quasi-Periodic Noise, and Baseline Drift," *Analytical Chemistry*, vol. 81, no. 12, pp. 4987–4994, 2009. PMID: 19449858.

[9] H. H. Madden, "Comments on the Savitzky-Golay convolution method for least-squares-fit smoothing and differentiation of digital data," *Analytical Chemistry*, vol. 50, no. 9, pp. 1383–1386, 1978.

[10] P. Barak, "Smoothing and Differentiation by an Adaptive-Degree Polynomial Filter," *Analytical Chemistry*, vol. 67, no. 17, pp. 2758–2762, 1995.

[11] S. Darlington, "Filters with Chebyshev stopbands, flat passbands, and impulse responses of finite duration," *Circuits and Systems, IEEE Transactions on*, vol. 25, pp. 966 – 975, dec 1978.

[12] P. Huber, *The behavior of maximum likelihood estimates under nonstandard conditions*, vol. 1, pp. 221–233. Prentice-Hall: New York, 1967.

[13] A. M. Al-haj, "Fast discrete wavelet transformation using FPGAs and distributed arithmetic," *International Journal of AppliedScienceandEngineering*, vol. 1, pp. 160–171, 2003.

[14] O. Rioul and M. Vetterli, "Wavelets and signal processing," *Signal Processing Magazine, IEEE*, vol. 8, pp. 14 –38, oct. 1991.

[15] V. R. G. Beylkin, R. Coifman, *Wavelets in Numerical Analysis.* NewYork: Jones and Bartlett, 1992.

[16] D. J. Field, "Wavelets, vision and the statistics of natural scenes," *Philosophical Transactions of the Royal Society of London. Series A:Mathematical, Physical and Engineering Sciences*, vol. 357, no. 1760, pp. 2527–2542, 1999.

[17] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *Image Processing, IEEE Transactions on*, vol. 1, pp. 205 –220, apr 1992.

[18] M. I. Mahmoud, M. I. M. Dessouky, S. Deyab, and F. H. Elfouly, "Comparison between Haar and Daubechies Wavelet Transformions on FPGA Technology," 2008.

[19] X. Zhang, Y. Wang, and R. P. S. Han, "Wavelet transform theory and its application in EMG signal processing," in *Proc. Seventh Int Fuzzy Systems and Knowledge Discovery (FSKD) Conf*, vol. 5, pp. 2234–2238, 2010.

[20] H. Darwish, M. Hesham, A.-M. Taalab, and N. Mansour, "Close Accord on DWT Performance and Real-Time Implementation for Protection Applications," *Power Delivery, IEEE Transactions on*, vol. 25, pp. 2174 –2183, oct. 2010.

[21] E. D. Ubeyli, G. Ilbay, D. Sahin, and N. Ates, "Discrete wavelet transform for analysis of spike-wave discharges in rats," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pp. 4680 –4683, aug. 2008.

[22] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, pp. 674 –693, jul 1989.

[23] R. Brychta, S. Tuntrakool, M. Appalsamy, N. Keller, D. Robertson, R. Shiavi, and A. Diedrich, "Wavelet Methods for Spike Detection in Mouse Renal Sympa-

thetic Nerve Activity," *Biomedical Engineering, IEEE Transactions on*, vol. 54, pp. 82 –93, jan. 2007.

[24] A. Keuer, R. Schrott, J. Taube, D. Schmuck, H. Beikirch, and W. Baumann, "FPGA based time detection of spikes within neural signals," in *Sensor Array and Multichannel Signal Processing Workshop Proceedings, 2004*, pp. 186 – 190, july 2004.

[25] J. S. Walke, *A Primer on Wavelets and Their Scientific Applications.* Chapman and Hall/CRC, 2009.

[26] I. Kaplan, "Applying the Haar Wavelet Transform to Time Series Information," 2 2004.

[27] S. Chowdhury and D. Chakrabarti, "Daubechies wavelet decomposition based baseline wander correction of trans-abdominal maternal ECG," in *Electrical and Computer Engineering (ICECE), 2010 International Conference on*, pp. 630 – 633, dec. 2010.

[28] Xilinx, *System Generator for DSP: Getting Started Guide*, 10.1 ed., 3 2008.

[29] U. Meyer-Baese, "Discrete wavelet transform FPGA design using MatLab/Simulink," *Proceedings of SPIE*, pp. 624703–624703–10, 2006.

[30] M. Knezevic, "An Introduction to Xilinx System Generator."

[31] "Xilinx Software Solutions."

Appendix A

Software Design

A.1 Simulink Modules

This section provides the Xilinx system generation modules that is used to generate the bit stream file to program the FPGA chip.

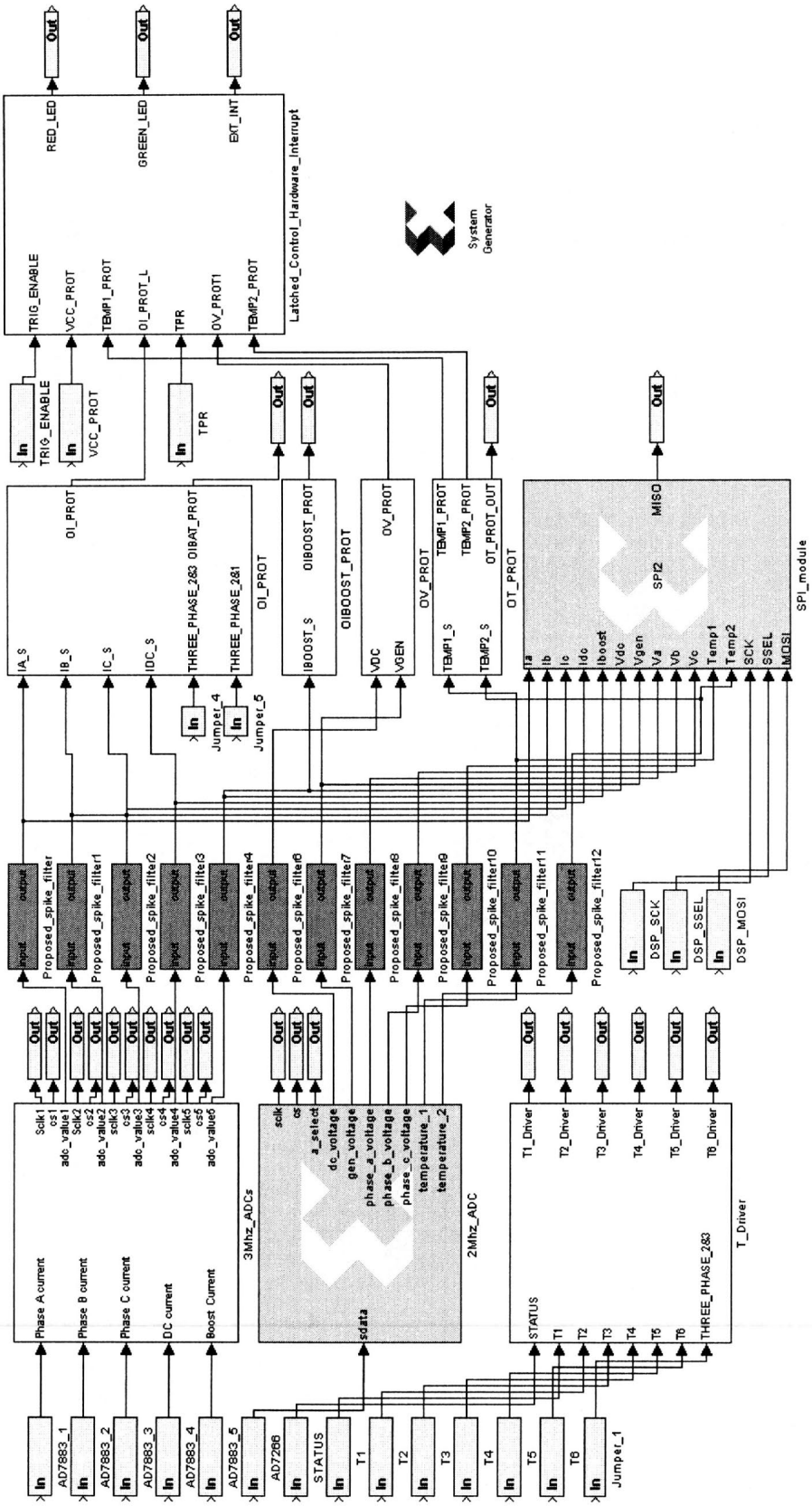

Figure A.1: Simulink module design of the interface board

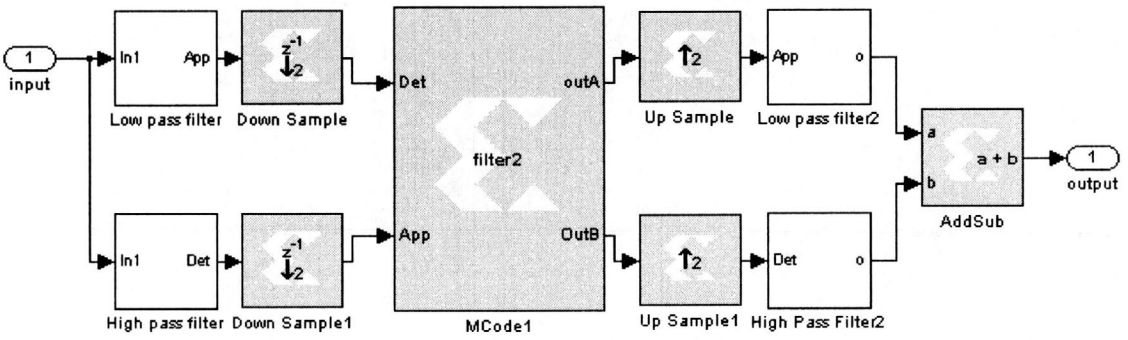

Figure A.2: Simulink module design of the spike removal algorithm

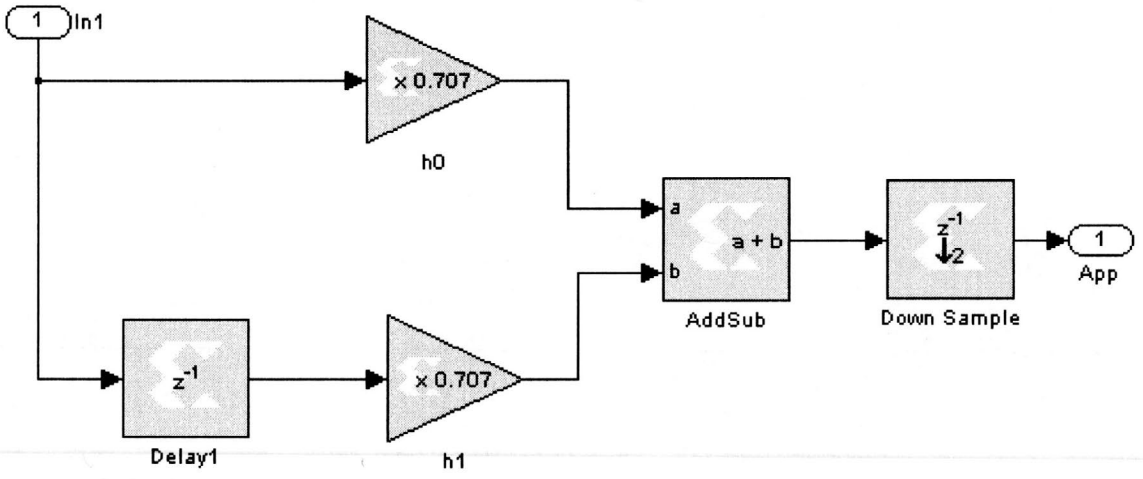

Figure A.3: Simulink module design of the Haar filter

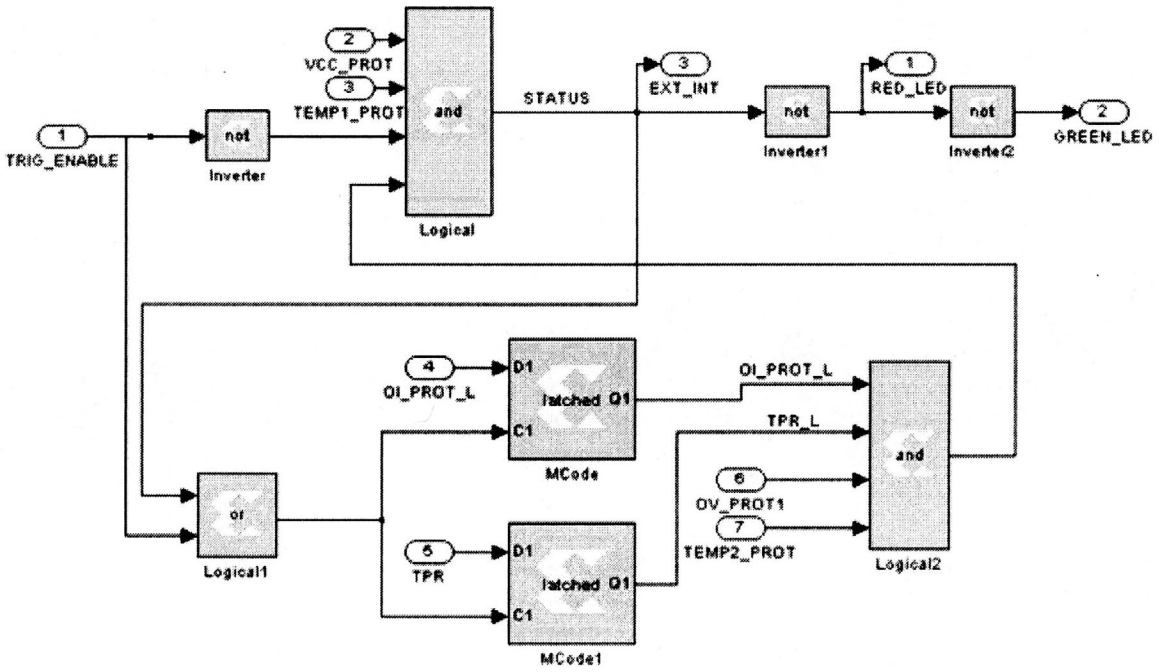

Figure A.4: Simulink module design of the latched control hardware interrupt

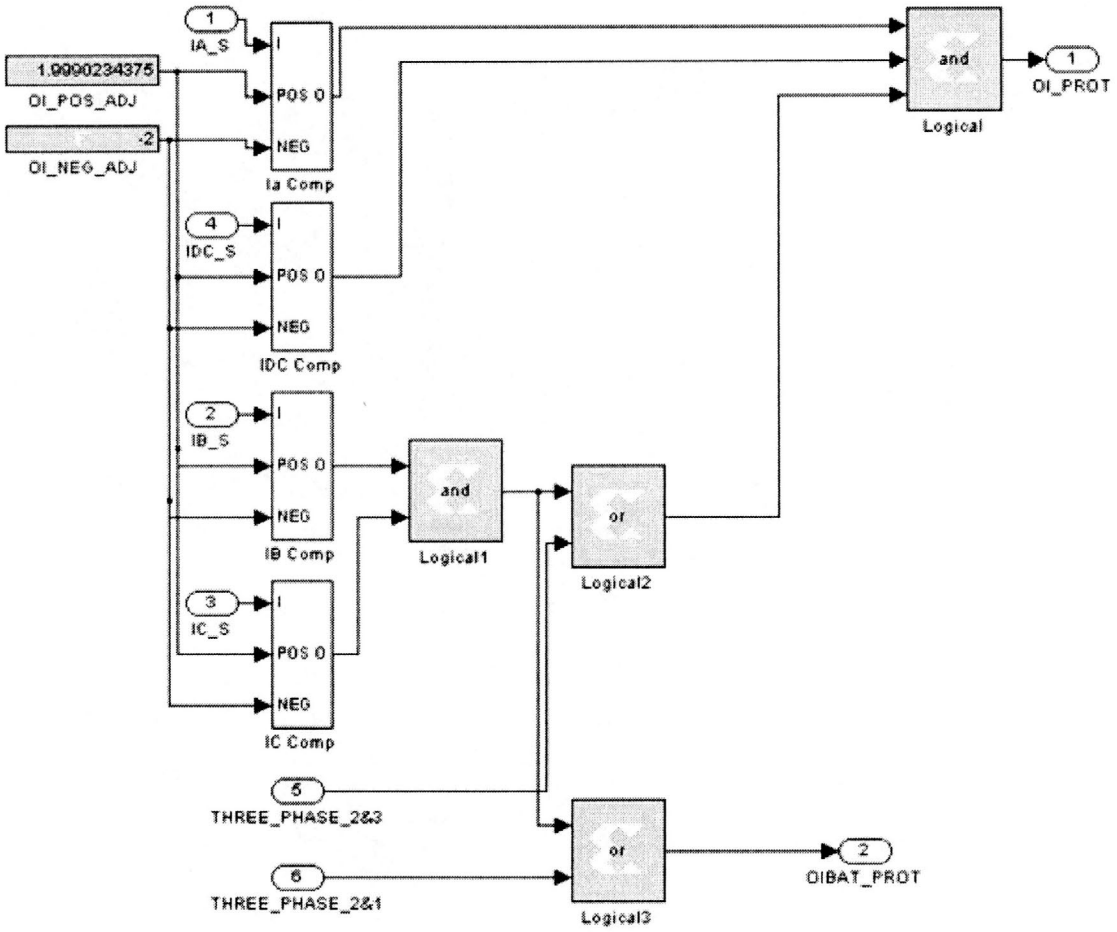

Figure A.5: Simulink module design of the OI detection for current signals

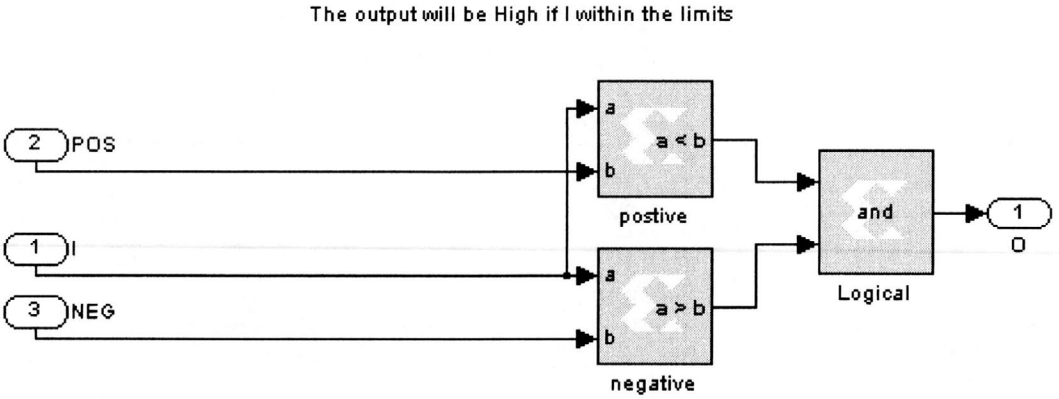

Figure A.6: Simulink module design of the current comparator

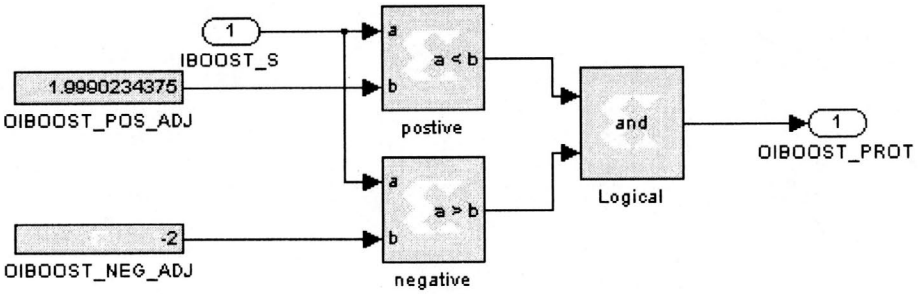

Figure A.7: Simulink module design of the OI detection for boost-current signal

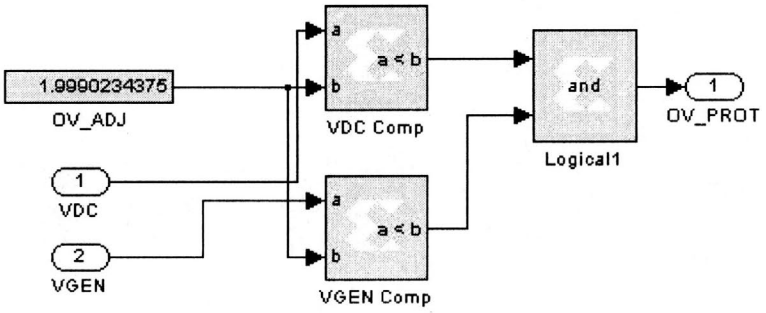

Figure A.8: Simulink module design of the OV detection

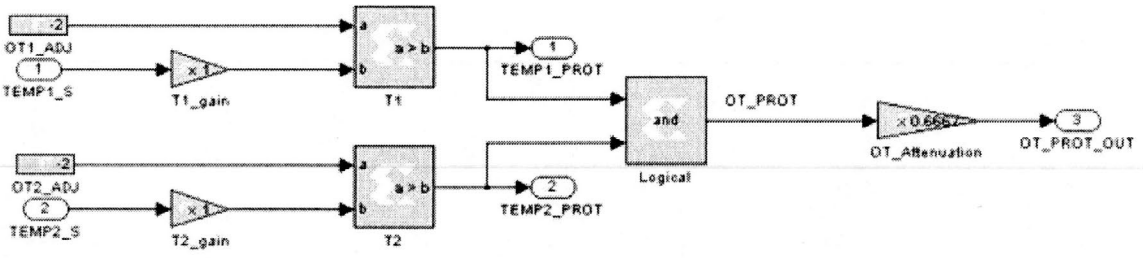

Figure A.9: Simulink module design of the OT detection

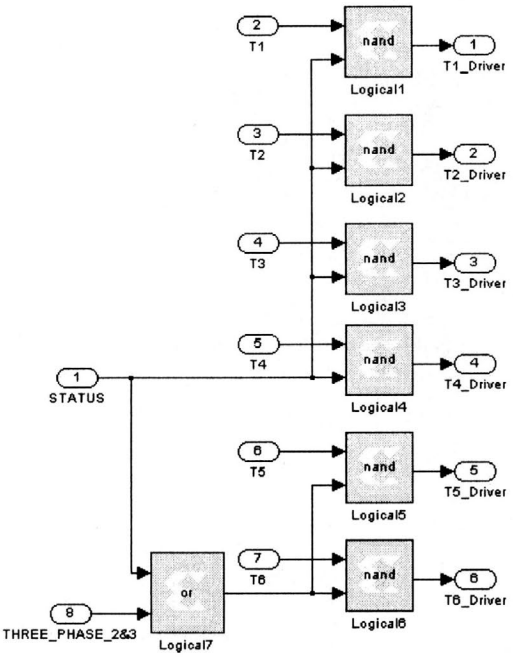

Figure A.10: Simulink module design of IGBT driving signals

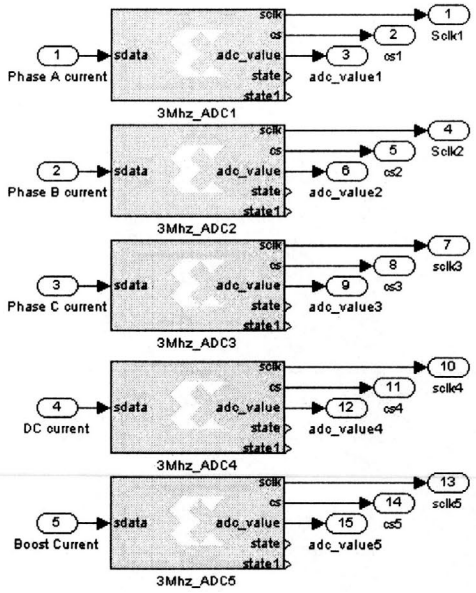

Figure A.11: Simulink module design of ADS7883 ADC

A.2 FPGA Input/Output ports

Table A.1: Input ports of FPGA chip

Port Name	FPGA Pin	Discription
AD7883 1	P159	Serial data for current A sensor
AD7883 2	P195	Serial data for current B sensor
AD7883 3	P204	Serial data for current C sensor
AD7883 4	P174	Serial data for DC current sensor
AD7883 5	P175	Serial data for Boost current sensor
DOUT A	P177	Serial data A of AD7266
DOUT B	P178	Serial data B of AD7266
T1 IN	P110	DSP signal to control the first IGBT
T2 IN	P118	DSP signal to control the second IGBT
T3 IN	P124	DSP signal to control the third IGBT
T4 IN	P130	DSP signal to control the fourth IGBT
T5 IN	P142	DSP signal to control the fifth IGBT
T6 IN	P148	DSP signal to control the sixth IGBT
TB IN	P154	DSP signal to control the Boost Chopper IGBT
1ph/3ph	P26	Input to select 1-phase or 3-phase
SPICLK	P139	SPI communication clock
SPISTE	P120	SPI communication slave select
SPISIMO	P119	SPI communication input data
TRIG ENABLE3	P135	Trigger enable signal from DSP
T56 ENABLE3	P134	Trigger enable signal from DSP
GCLK10	P185	FPGA clock signal
RC1	P126	DSP control signal to Relay 1
RC2	P127	DSP control signal to Relay 2
RC3	P128	DSP control signal to Relay 3
RC4	P129	DSP control signal to Relay 4
RC5	P132	DSP control signal to Relay 5
RC6	P133	DSP control signal to Relay 6
START IN	P6	DSP Start signal
STOP IN	P14	DSp Stop signal

Table A.2: Output ports of FPGA chip

Port Name	FPGA Pin	Discription
AD SCLK	P205	Serial clock signal to AD7883 ADCs
ADV SCLK	P199	Serial clock signal to AD7266 ADC
AD CS	P160	Chip select signal to AD7883 ADCs
ADV CS	P167	Chip select signal to AD7266 ADC
AD A0	P161	Bit1 mux select signal to AD7266
AD A1	P162	Bit2 mux select signal to AD7266
AD A2	P163	Bit3 mux select signal to AD7266
AD SGL	P162	Diff. or single ended signal to AD7266
AD RANGE	P189	Input Range Selection signal to AD7266
T1 OUT	P93	Output to drive the first IGBT
T2 OUT	P94	Output to drive the second IGBT
T3 OUT	P96	Output to drive the third IGBT
T4 OUT	P97	Output to drive the fourth IGBT
T5 OUT	P99	Output to drive the fifth IGBT
T6 OUT	P100	Output to drive the sixth IGBT
TB OUT	P102	Output to drive the Boost Chopper IGBT
SPISOMI	P122	SPI communication output data
RC1S	P168	Relay 1 drive signal
RC2S	P112	Relay 2 drive signal
RC3S	P98	Relay 3 drive signal
RC4S	P35	Relay 4 drive signal
RC5S	P133	Relay 5 drive signal
RC6S	P144	Relay 6 drive signal
OI PROT	P106	Over current protection signal
OV PROT	P107	Over voltage protection signal
OT PROT	P107	Over temperature protection signal
OIBAT PROT	P109	Over current protection signal
TPR PROT	P20	TPR protection signal
EXIT INT	P123	External interrupt signal
START	P137	Start signal
STOP	P138	Stop signal
RED LED	P89	Red led control signal
GREEN LED	P90	Green led control signal

A.3 Source Code

This section provides the source codes for the MCode and Blackbox blocks that are used in the Xilinx system generation modules.

A.3.1 AD7266 source code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Haider Mohomad AR
//
// Create Date:    12:17:47 07/19/2011
// Design Name:
// Module Name:    ad7266
// Project Name: Interface board design
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module ad7266(clk,clkin,ce,sclk,cs,sdata,adc_value,adc_value2,adc_value3,adc_value4,ad_sgl,a_select,ad_range,state)
input clk;
input clkin;
input ce;
output sclk;
output [4:0] state;
output cs;
output [11:0]adc_value;
output [11:0]adc_value2;
output [11:0]adc_value3;
output [11:0]adc_value4;
input sdata;
output ad_sgl;
```

```
output ad_range;
output [2:0] a_select;
reg cs = 1'b1;
reg [11:0]adc_value = 0;
reg [11:0]adc_value2 = 0;
reg [11:0]adc_value3 = 0;
reg [11:0]adc_value4 = 0;
reg [11:0]adc_channel= 0; //channel adc output
reg ad_sgl = 1'b1; // high = single ended, low = true diff
reg ad_range = 1'b1; // high = 0-2Vref, low = 0-Vref
reg [2:0] a_select = 3'b011;
reg [1:0] state1 = 0; // to control the selection A2 A1 A0
reg [4:0]state = 5'b00000; //to count 14 clk
assign sclk = clk;


always @(posedge clk)
begin
state <= state + 5'b00001;
case (state)
5'b00011:
// state=3, begin to convert
begin
cs <= 1'b0;
end

endcase
                5'b10001:
                        begin
     case(a_select)
3'b010: begin adc_value[11:0] <= adc_channel[11:0]; a_select<=3'b011;end
3'b011: begin adc_value2[11:0] <= adc_channel[11:0];a_select<=3'b100;end
3'b100: begin adc_value3[11:0] <= adc_channel[11:0];a_select<=3'b101;end
3'b101: begin adc_value4[11:0] <= adc_channel[11:0];a_select<=3'b010;end
default: a_select<=3'b010;

endcase
    cs <= 1'b1;
  end
5'b10010:
// state=18, shift data to FPGA
```

```
begin

//
//
state <= 5'b00011;

end

endcase
end
always @ (negedge clk)
begin
if ((state >= 5'b00100) && (state <= 5'b10001))
begin
adc_channel[11:1] <= adc_channel[10:0]; // transfer serial data to parallel
adc_channel[0] <= sdata;
end
end


endmodule
```

A.3.2 ADS7883 source code

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: UNB
// Engineer:  Roger and Haider
//
// Create Date:    15:02:51 05/19/2011
// Design Name:    ADC codes for ADS7883 (2MHZ)
// Module Name:    adc_7883
// Project Name:   FPGA embedded Interface Board
// Target Devices:   Spartan 3E
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments: Provide 5V supply and change the DCM to 48MHZ, the adc will sample data at 3MHZ.
//                      connect a serial 300 ohm to the FPGA to transfer SDO
//
//////////////////////////////////////////////////////////////////////////////////
module adc_7833(clk,ce,sclk,cs,adc_value,sdata,state,state1,wr_en,re_en,test);
input clk;
input ce;
output sclk;
output cs;
output [11:0]adc_value;
input sdata;
output [4:0]state;
output [11:0]state1;
output wr_en,re_en;
output test;



reg cs = 1'b1;
reg [11:0]adc_value = 0;
reg [11:0]adc_channel= 0;  //channel adc output
reg [11:0] state1 = 0; // to generate lower clock to observe the output waveform
reg [4:0]state = 5'b00000; //state
```

```
reg wr_en = 0;
reg re_en = 0;



assign sclk = clk;
assign test = cs;

always@(posedge clk)
   begin
     state1 <= state1 + 12'd1;

  if (state1 == 12'b111111111111)
   begin
     state1 <= 12'd0;
   end


end

always @(posedge clk)
   begin
  state <= state + 5'b00001;
  case (state)
  5'b00011:   // state=3, begin to convert
       begin cs <= 1'b0;
 end
  5'b10001:   // state=17, shift data to FPGA
    begin
 adc_value[11:0] <= adc_channel[11:0];
 cs <= 1'b1; // assert high to stop conversion
 //test <= 0;
 end
  5'b10010:  // go back to state=3 and ready for next conversion
    begin

  state <= 5'b00011;
       end

  endcase
end
```

```
always @(posedge clk)
    begin

      if (state == 5'b10001)
   wr_en <= 1;
else wr_en <=0;

      if (state == 5'b10010)
          re_en <= 1;
   else re_en <= 0;

    end

always @ (negedge clk)
begin
     if ((state >= 5'b00100) && (state <= 5'b10001))
  begin
       adc_channel[11:1] <= adc_channel[10:0];  // transfer serial data to parallel
 adc_channel[0] <= sdata;

  end

end


endmodule
```

A.3.3 Serial Peripheral Interface (SPI) Source Code

```
function [MISO] = SPI2(Ia,Ib,Ic,Idc,Iboost,Vdc,Vgen,Va,Vb,Vc,Temp1,Temp2,Select, SCK, SSEL, MOSI )
proto = {xlSigned, nbits, binpt};
persistent mem, mem = xl_state(zeros(1, depth), proto);
%persistent rom, rom = xl_state(zeros(1, depth), proto);
persistent dout_temp, dout_temp = xl_state(0, proto);
persistent state, state = xl_state(0, {xlUnsigned, 4, 0});

persistent v1r, v1r = xl_state(0, proto);
persistent v2r, v2r = xl_state(0, proto);
persistent v3r, v3r = xl_state(0, proto);
persistent v4r, v4r = xl_state(0, proto);



persistent r, r = xl_state(zeros(1, 3), SCK);
persistent SSELr, SSELr = xl_state(ones(1, 3), SSEL);
persistent MOSIr, MOSIr = xl_state(zeros(1, 2), MOSI);

persistent bitcnt, bitcnt = xl_state(0, {xlUnsigned, 3, 0});
persistent byte_recevived, byte_recevived = xl_state(0, {xlBoolean});
persistent byte_data_recevived, byte_data_recevived = xl_state(zeros(1, 8), MOSI);

persistent byte_data_send, byte_data_send = xl_state(0, {xlUnsigned, 8, 0});

persistent bitcnt_send, bitcnt_send = xl_state(0, {xlUnsigned, 3, 0});
persistent byte_sending, byte_sending = xl_state(0, {xlBoolean});
persistent bit_data_send, bit_data_send = xl_state(0, {xlBoolean});
persistent byte_send_completed, byte_send_completed = xl_state(0, {xlBoolean});

persistent complet_send, complet_send = xl_state(0, {xlBoolean});
persistent state_send, state_send = xl_state(0, {xlUnsigned, 4, 0});
persistent MISOr, MISOr = xl_state(0, {xlUnsigned, 1, 0});



 if we
    v1r = v1;
    v2r = v2;
    v3r = v3;
    v4r = v4;
 end
```

```
r(2)=r(1);
r(1)=r(0);
r(0)=SCK;

SSELr(2)=SSELr(1);
SSELr(1)=SSELr(0);
SSELr(0)=SSEL;

MOSIr(1)=MOSIr(0);
MOSIr(0)=MOSI;

sck_risingEdge = ( r(2) == false & r(1)==true);
sck_fallingEdge = ( r(2) == true & r(1)==false);
SSEL_active = ~SSELr(1);
SSEL_Start =( SSELr(2) == true & SSELr(1)==false);
SSEL_End   =( SSELr(2) == false & SSELr(1)==true);
MOSI_Data = MOSIr(1);


if(~SSEL_active)        % Reset controlling signals
    bitcnt = 0;
    bitcnt_send=0;
    complet_send=false;
    state_send=0;
    MISOr=0;
    MISO=0;

else                    % SSEL Activate
    if(sck_risingEdge)
        if(~byte_sending)                   % Data Receving
        bitcnt = bitcnt + 1;
        byte_data_recevived(7)=byte_data_recevived(6);
        byte_data_recevived(6)=byte_data_recevived(5);
        byte_data_recevived(5)=byte_data_recevived(4);
        byte_data_recevived(4)=byte_data_recevived(3);
        byte_data_recevived(3)=byte_data_recevived(2);
        byte_data_recevived(2)=byte_data_recevived(1);
        byte_data_recevived(1)=byte_data_recevived(0);
```

```
        byte_data_recevived(0)=MOSI_Data;
        MISOr=0;


        else                                % Sending Data

                    if(~complet_send )
                        switch double(state_send)
                          case 0
                              MISOr= xl_slice(byte_data_send, 7, 7); state_send=1;
                          case 1
                              MISOr= xl_slice(byte_data_send, 6, 6); state_send=2;
                          case 2
                              MISOr= xl_slice(byte_data_send, 5, 5); state_send=3;
                          case 3
                              MISOr= xl_slice(byte_data_send, 4, 4); state_send=4;
                          case 4
                              MISOr= xl_slice(byte_data_send, 3, 3); state_send=5;
                          case 5
                              MISOr= xl_slice(byte_data_send, 2, 2); state_send=6;
                          case 6
                              MISOr= xl_slice(byte_data_send, 1, 1); state_send=7;
                          case 7
                               MISOr= xl_slice(byte_data_send, 0, 0); state_send=0; complet_send=true;
                            otherwise
                                MISOr=0;
                                state_send=0;
                        end
                    else


                    end
        end
    else

    end
end

MISO=MISOr;
byte_data_recevived_address = xl_concat(byte_data_recevived(7), byte_data_recevived(6), ...
    (byte_data_recevived(5)), (byte_data_recevived(4)), ...
    (byte_data_recevived(3)), (byte_data_recevived(2)), ...
```

```
    (byte_data_recevived(1)), (byte_data_recevived(0)));

byte_recevived = SSEL_active & sck_risingEdge & (bitcnt == 8);

disp(['SSEL_active = ', num2str(SSEL_active)]);
disp(['sck_risingEdge = ', num2str(sck_risingEdge)]);
disp(mem);
disp(['bitcnt = ', num2str(bitcnt)]);
disp(['byte_data_recevived = ', num2str(byte_data_recevived(7)), num2str(byte_data_recevived(6)), ...
    num2str(byte_data_recevived(5)), num2str(byte_data_recevived(4)), ...
    num2str(byte_data_recevived(3)), num2str(byte_data_recevived(2)), ...
    num2str(byte_data_recevived(1)), num2str(byte_data_recevived(0))]);
disp(['byte_recevived = ', num2str(byte_recevived)]);
disp(['byte_data_recevived_address = ', num2str(byte_data_recevived_address)]);

disp(['byte_data_send = ', num2str(byte_data_send)]);
disp(byte_data_send);
disp(['byte_sending = ', num2str(byte_sending)]);
disp(['state_send = ', num2str(state_send)]);
disp(['MISO = ', num2str(MISO)]);
disp('complet = ');
disp(complet_send);
disp([' ']);
disp([' ']);


if(~byte_recevived )
    switch double(state)
      case 0
          mem(state)=v1r; state = 1;
      case 1
          mem(state)=v2r; state = 2;
      case 2
          mem(state)=v3r; state = 3;
      case 3
          mem(state)=v4r; state = 0;
        otherwise
            state=0;
    end
else
    if(byte_data_recevived_address <depth)
```

```
        byte_data_send = mem(byte_data_recevived_address);
        byte_sending=true;
    else
       error('byte_data_recevived_address out of memory boundary');
    end
end

% bit_data_send = xl_slice(byte_data_send, bitcnt_send, bitcnt_send);
% MISO=bit_data_send;



% disp(['byte_data_send = ', byte_data_send]);
% disp(['bit_data_send = ', bit_data_send]);

% MISO=sck_risingEdge;

% disp(['r(2) = ', num2str(r(2)), ', ', ...
% 'r(1) = ', num2str(r(1)), ', ', ...
% 'r(0) = ', num2str(r(0))]);
% disp(['sck_risingEdge = ', num2str(sck_risingEdge), ', ','sck_fallingEdge = ', num2str(sck_fallingEdge)]);
% disp(['SSELr(2) = ', num2str(SSELr(2)) , ', SSELr(1) = ', num2str(SSELr(1)), ',  SSELr(0) = ', num2str(SSELr(0))]
% disp(['SSEL_active = ', num2str(SSEL_active),', SSEL_Start = ', num2str(SSEL_Start), ', ', 'SSEL_End = ', num2str
% disp(['MOSIr(1) = ', num2str(MOSIr(1)), ',  MOSIr(0) = ', num2str(MOSIr(0))]);
%  disp(['MOSI_Data = ', num2str(MOSI_Data)]);


end
```

A.3.4 Latched Source Code

```
function [Q1] = latched(D1,C1)
persistent r, r = xl_state(0,{xlBoolean});
if (C1)
    r=D1;
end
Q1=r;
disp(['C1 = ', num2str(C1),', D1 = ', num2str(D1),', r = ', num2str(r),', Q1 = ', num2str(Q1)]);
end
```

Appendix B

Interface Board PCB Design

This section provides the schematics for the interface board. The interface board PCB design was a mutual collaboration with Roger Luo, a UNB MScE Candidate.

B.1 Inpot/Output Ports

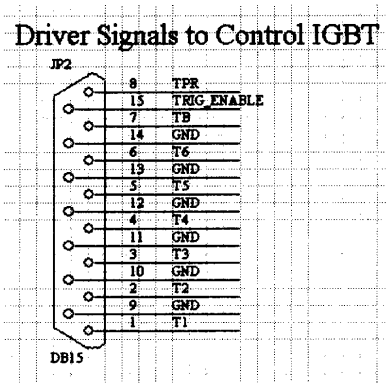

Figure B.1: Driver signals jumper

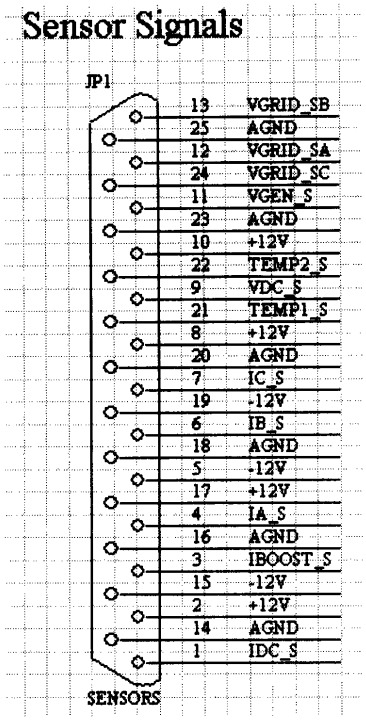

Figure B.2: Sensor signals jumper

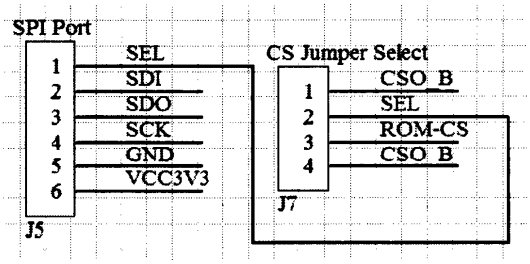

Figure B.3: SPI and CS jumpers

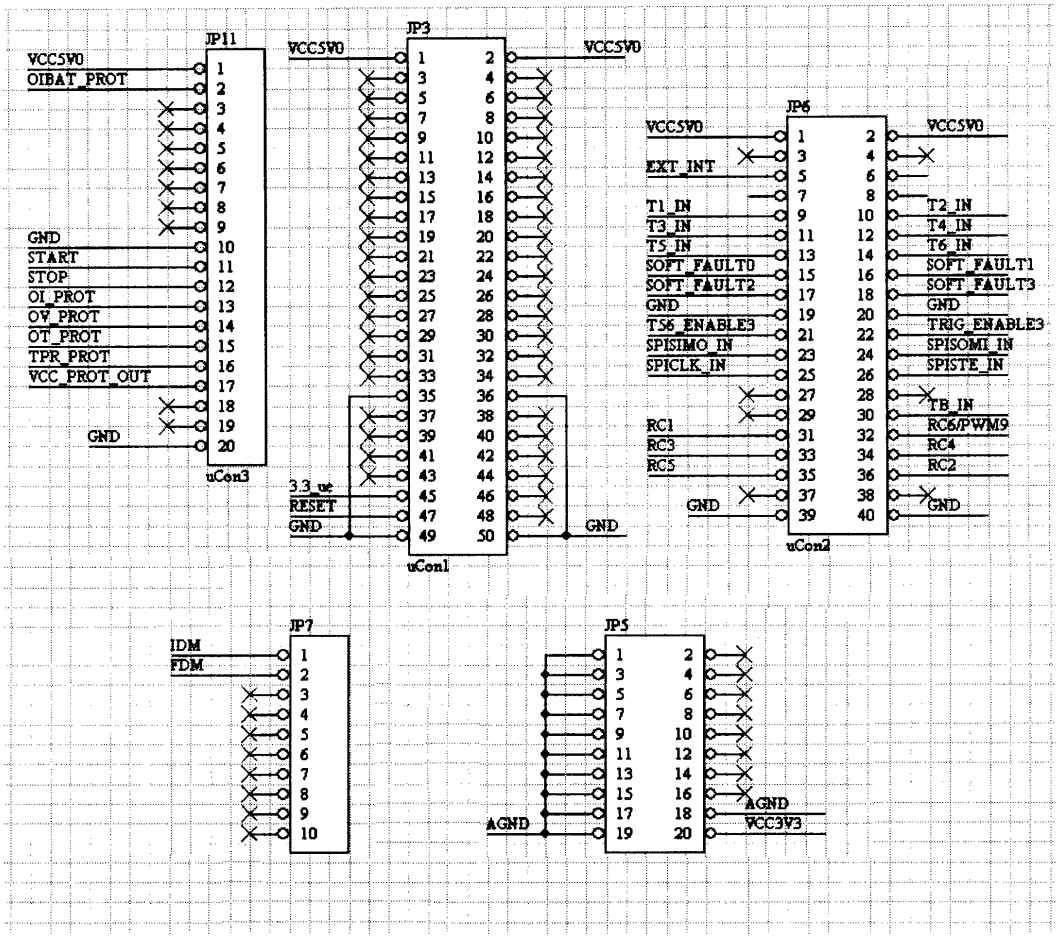

Figure B.4: Interface board to DSP jumpers

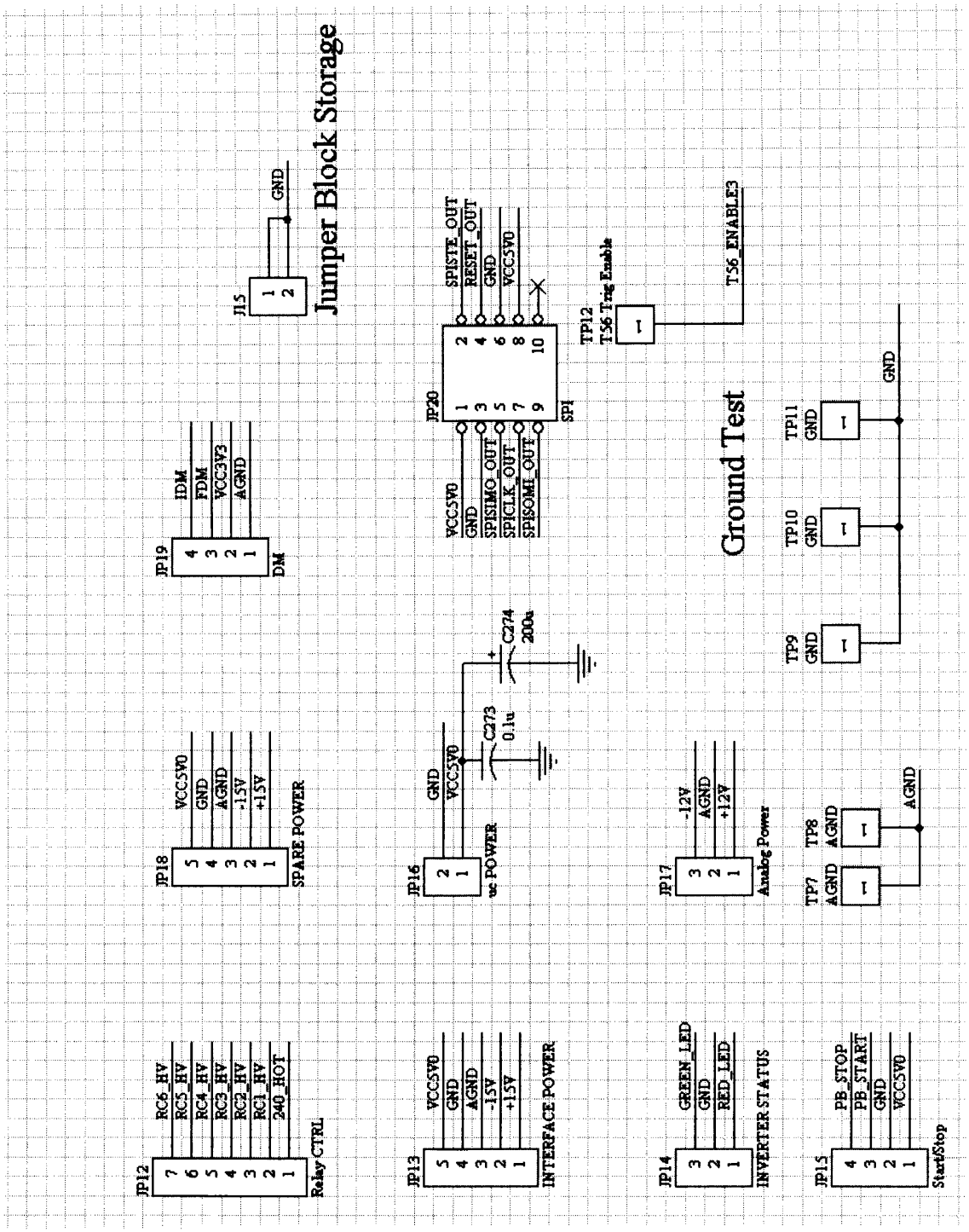

Figure B.5: Interface board jumpers

B.2 FPGA

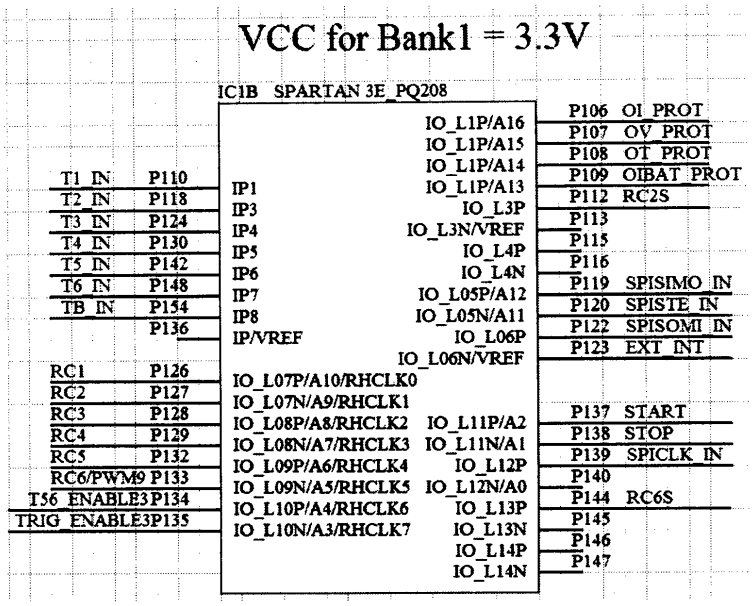

Figure B.6: FPGA Bank 1

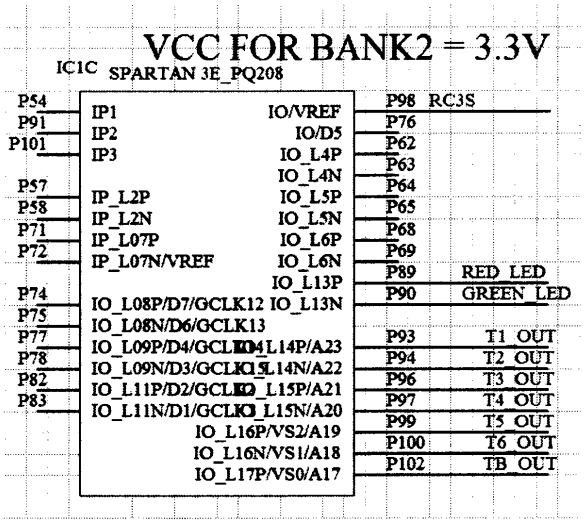

Figure B.7: FPGA Bank 2

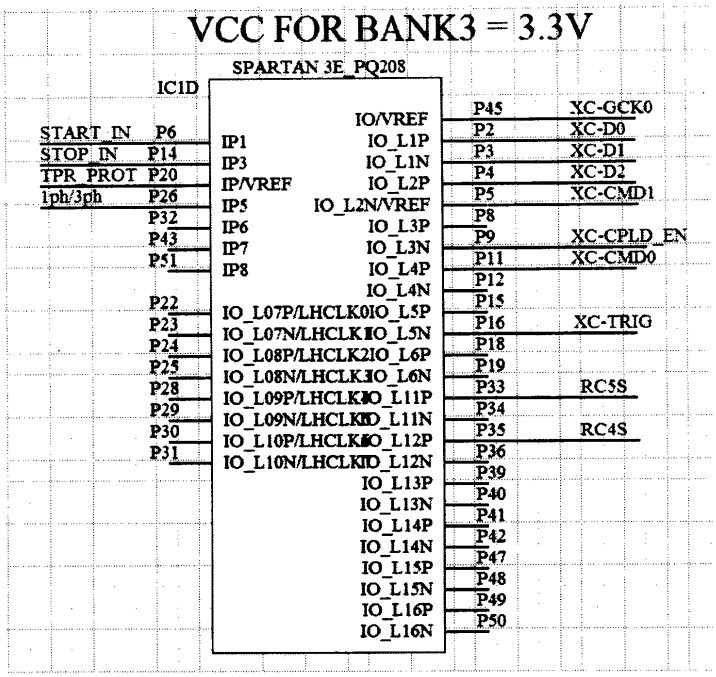

Figure B.8: FPGA Bank 3

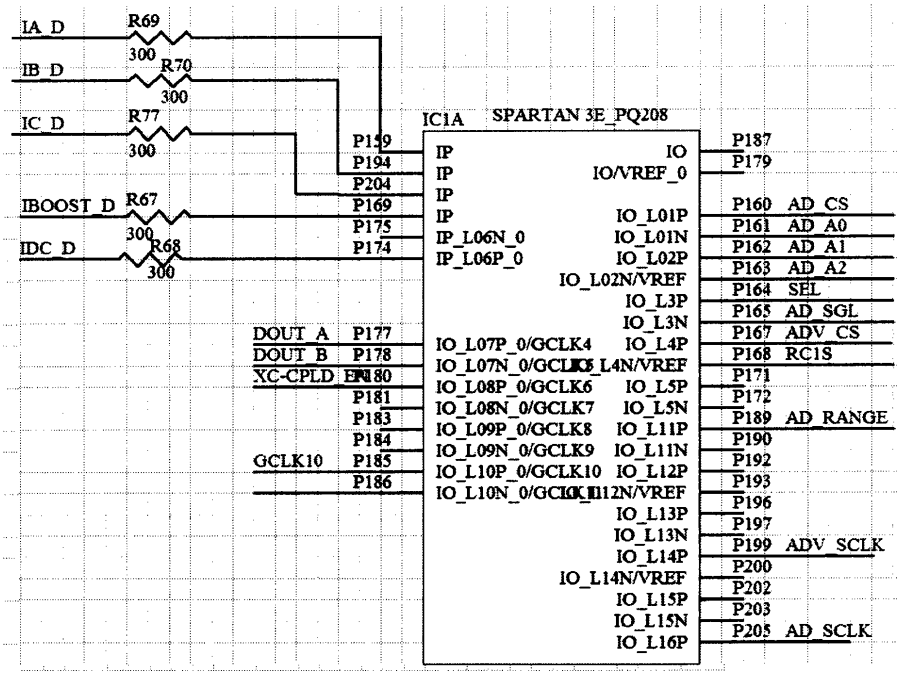

Figure B.9: FPGA Bank 4

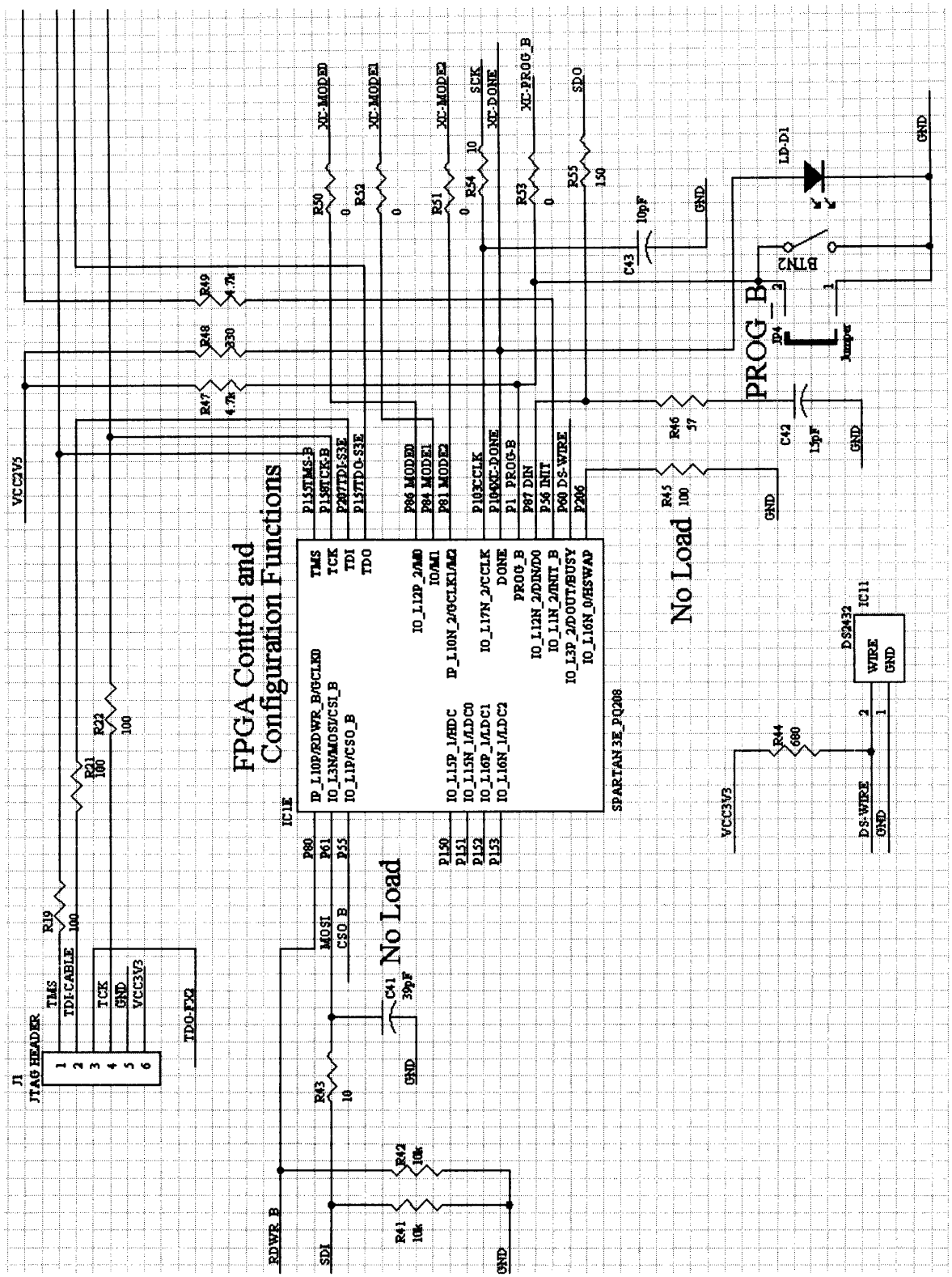

Figure B.10: FPGA control and configuration

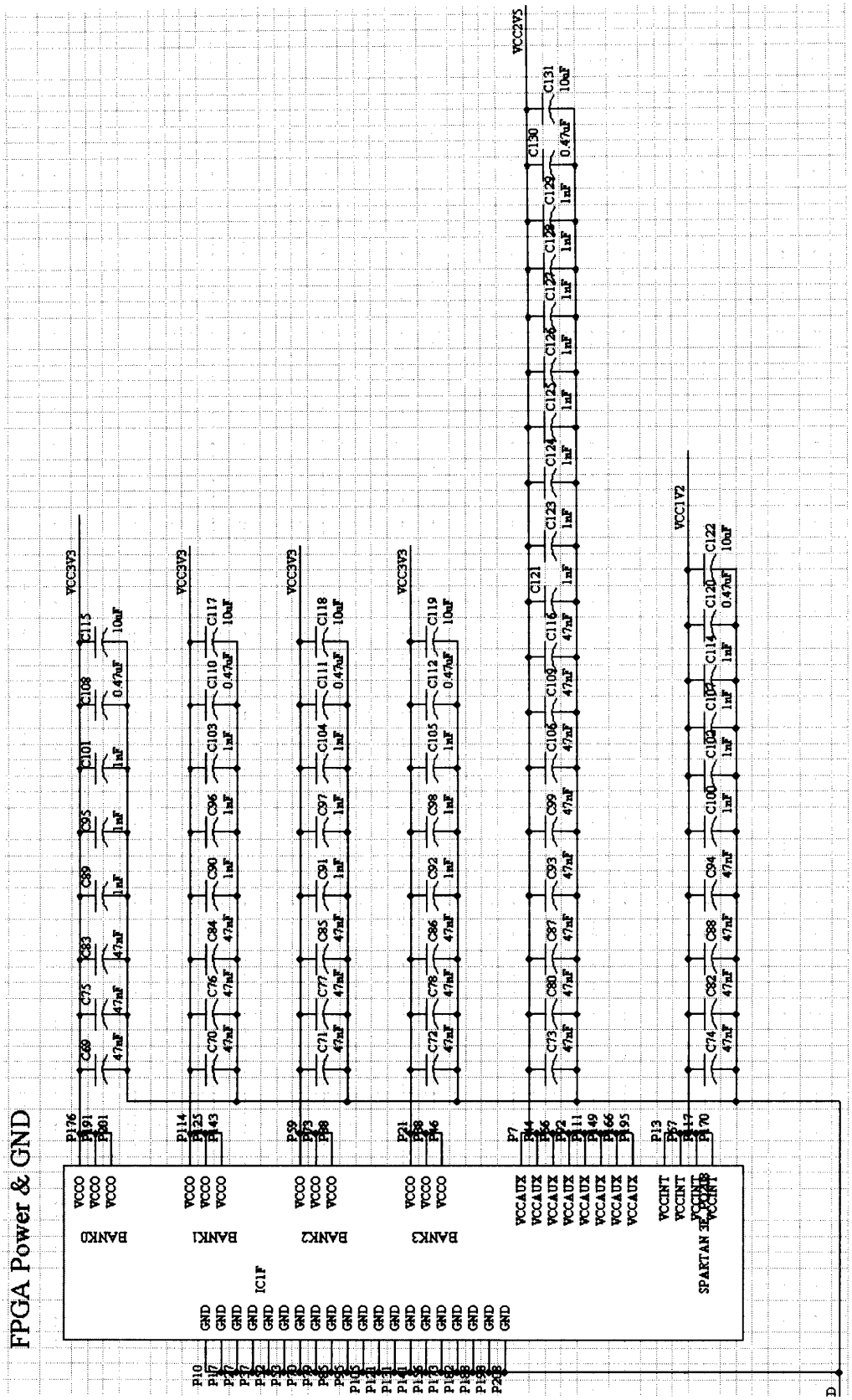

Figure B.11: FPGA power and ground

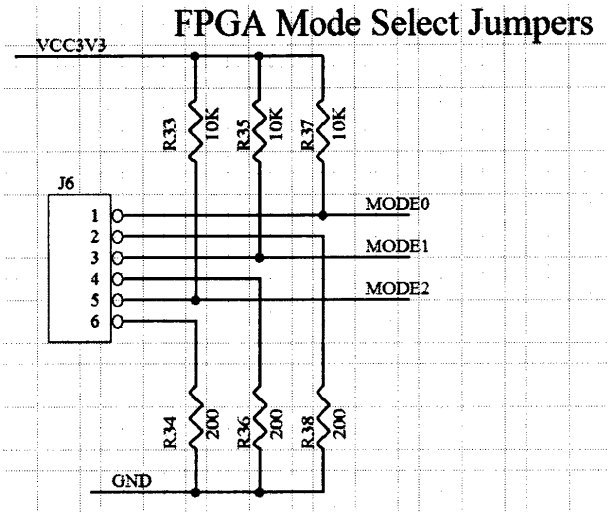

Figure B.12: FPGA mode select

B.3 Power Regulator

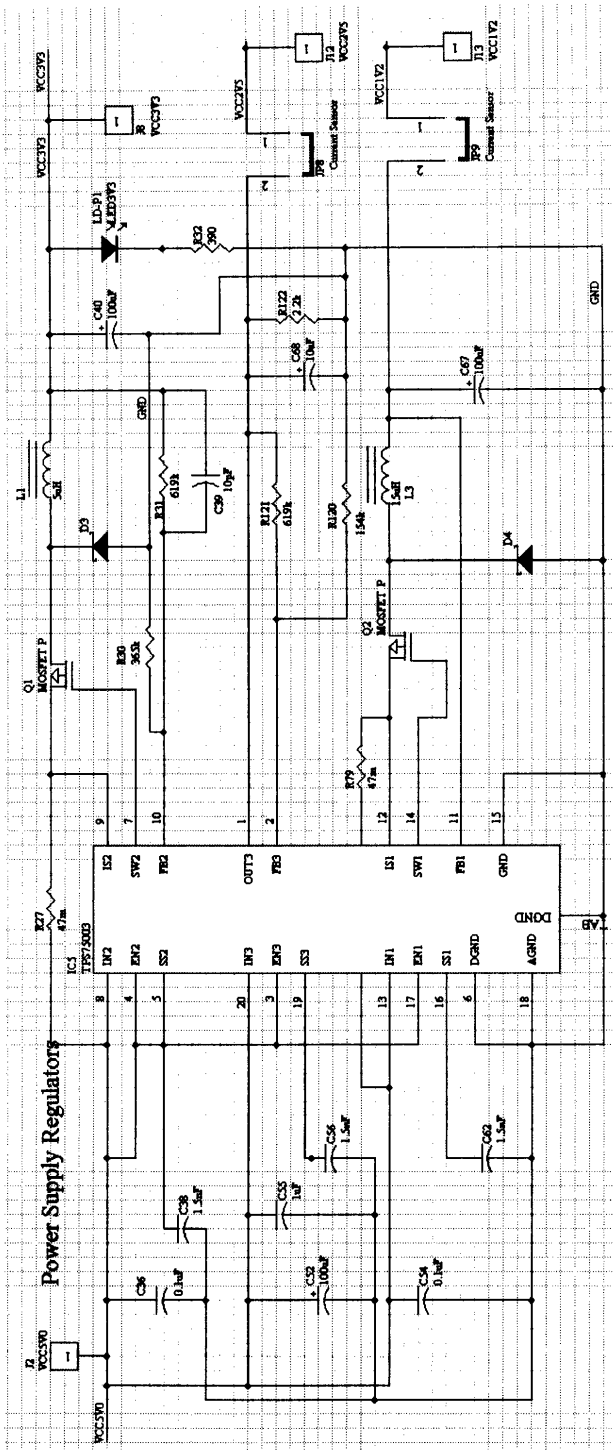

Figure B.13: Power regulator supply

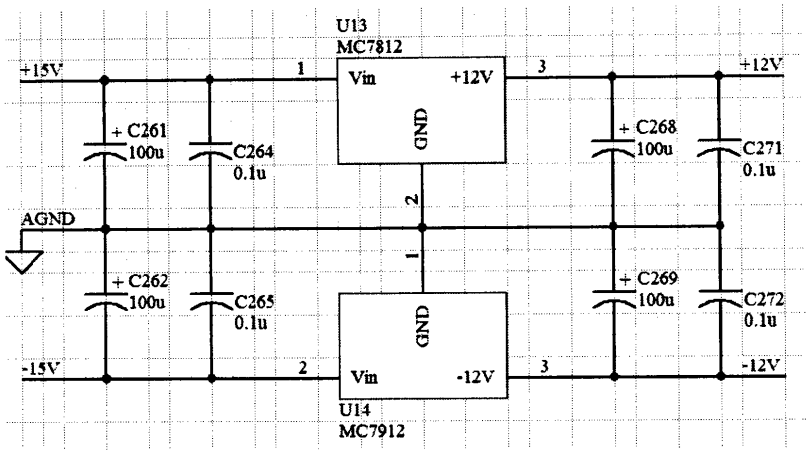

Figure B.14: Power supply circuit for 12V

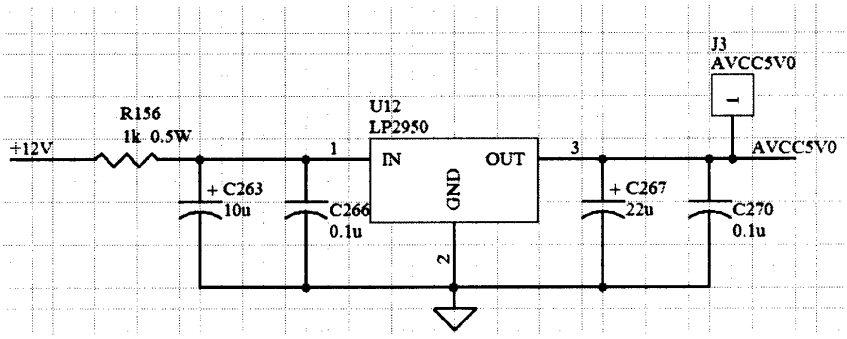

Figure B.15: Power supply circuit for 5V

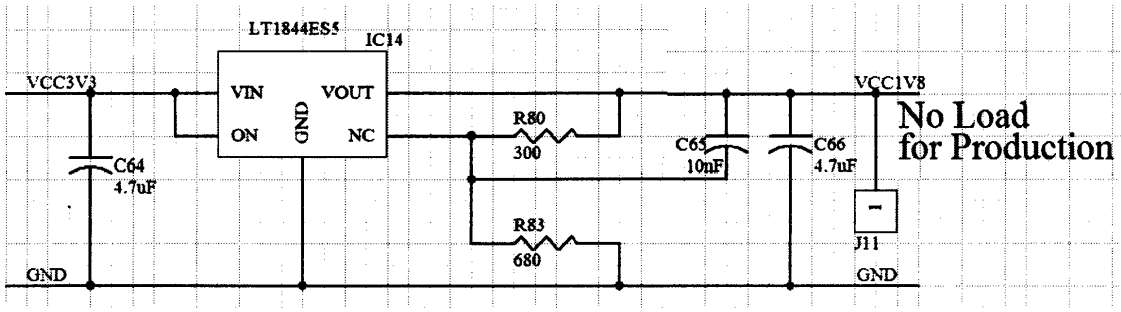

Figure B.16: Power supply circuit for 1.8V

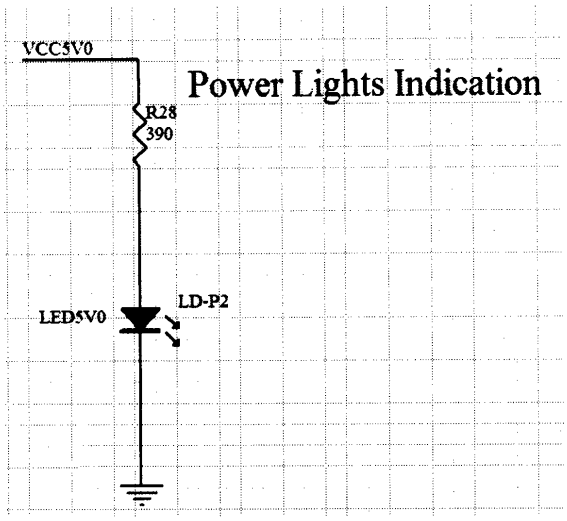

Figure B.17: Power LED indicator

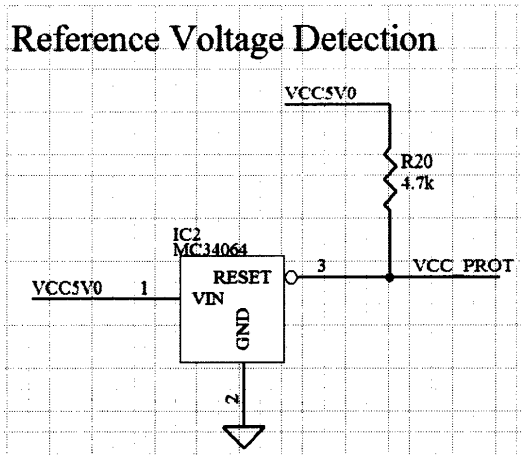

Figure B.18: Reference voltage detection

B.4 Analog to Digital Converters

The AD7266 is 2 MHz ADC that is used to convert the analog signals of the voltage and temperature sensors.

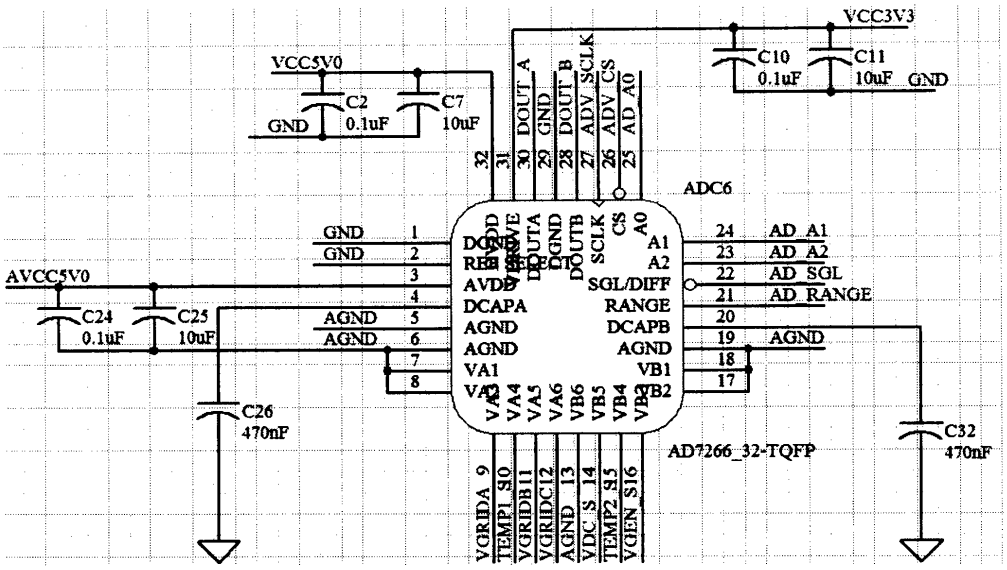

Figure B.19: AD7266 ADC

The ADS7883 is 3 MHz ADC that is used to convert the analog signals of the current sensors. There are five ADS7883 ADCs in the interface board.

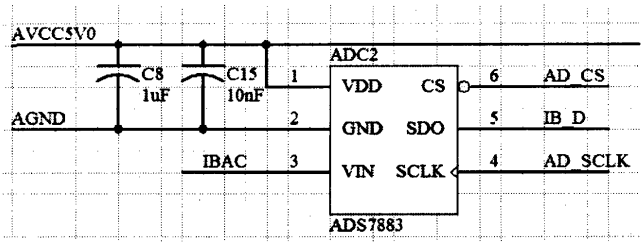

Figure B.20: ADS7883 ADC

B.5 Serial Flash Memory

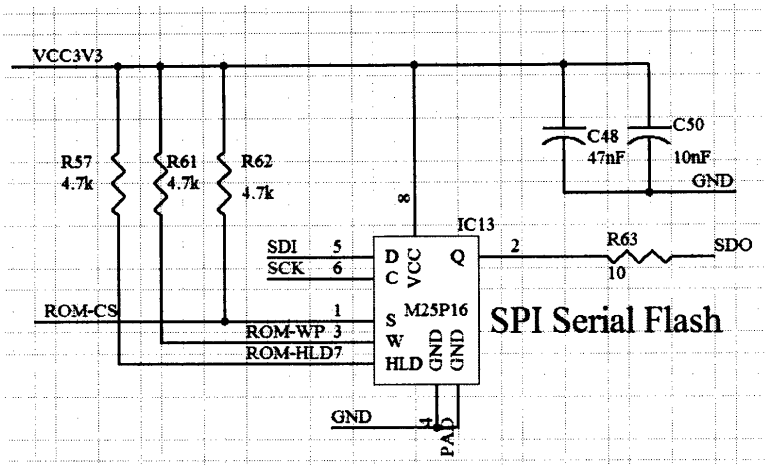

Figure B.21: 16 Mbit serial flash memory

B.6 Anti Aliasing Filters

The anti aliasing filters are used to limit the bandwidth of the input signals of the ADCs.

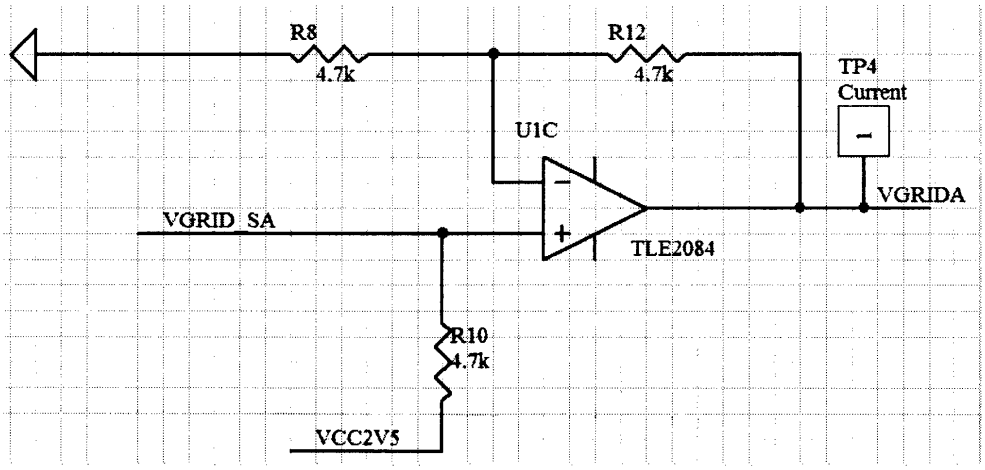

Figure B.22: Anti aliasing filter for grid A voltage sensor

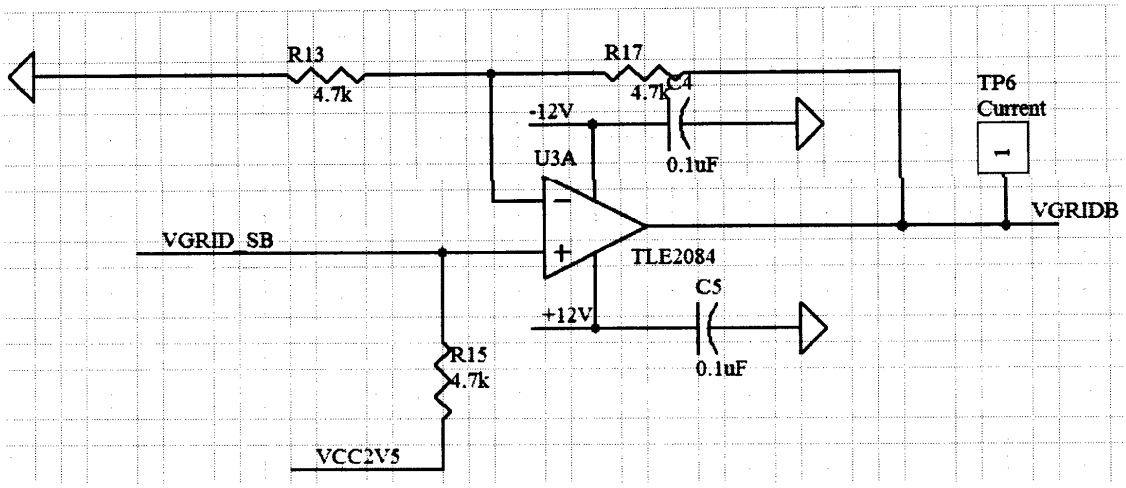

Figure B.23: Anti aliasing filter for grid B voltage sensor

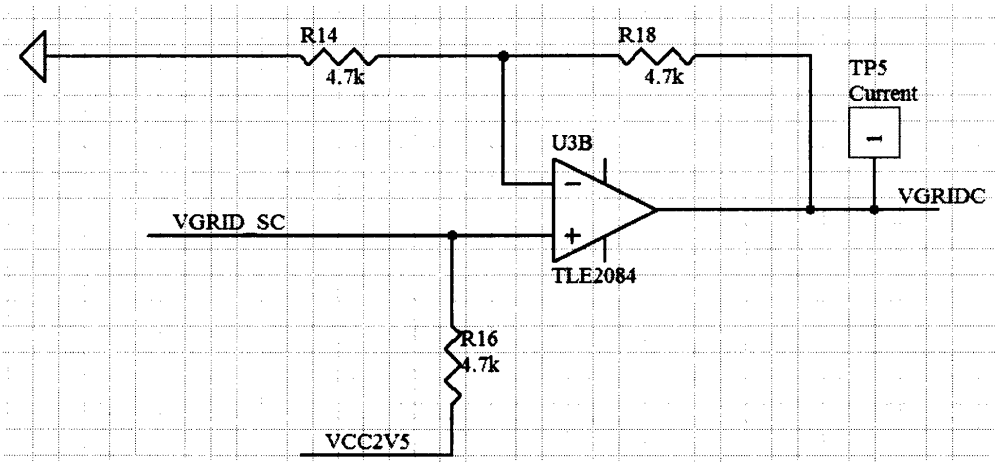

Figure B.24: Anti aliasing filter for grid C voltage sensor

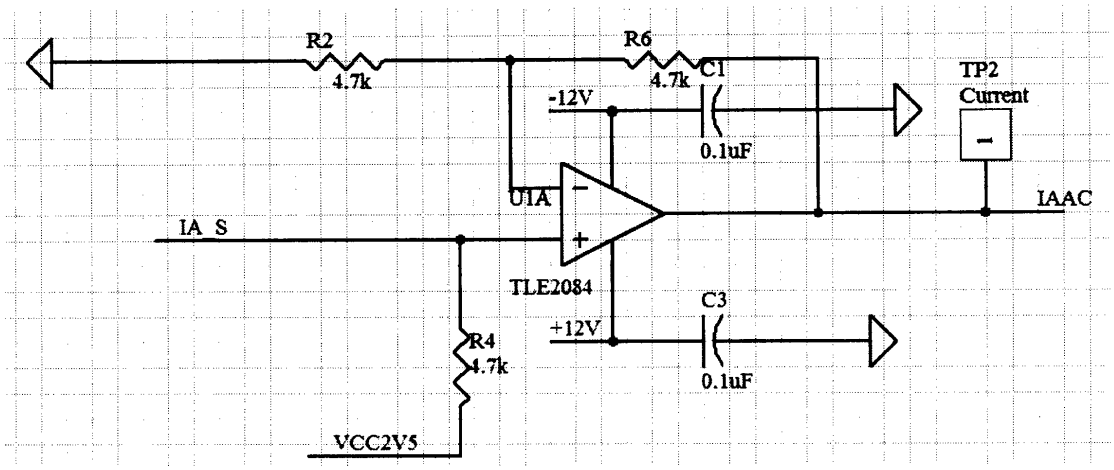

Figure B.25: Anti aliasing filter and level shifting for Phase A current sensor

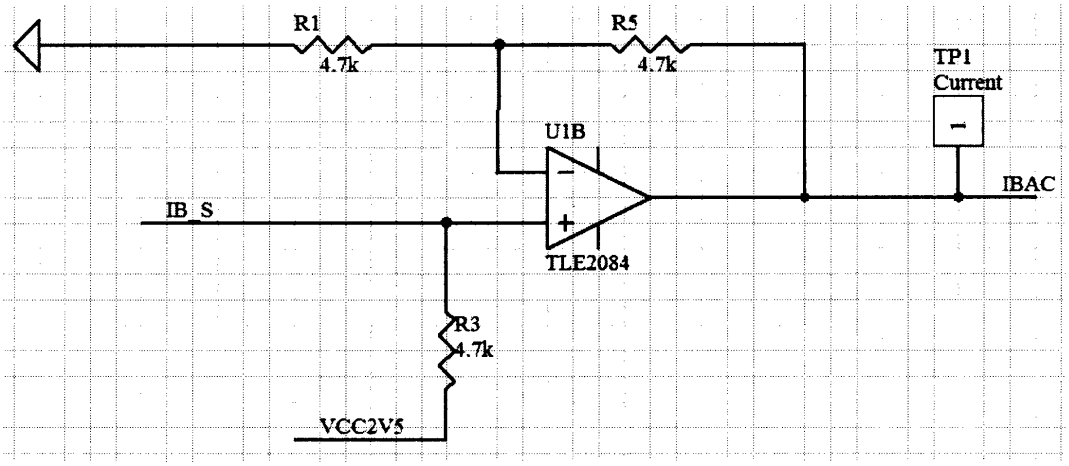

Figure B.26: Anti aliasing filter and level shifting for Phase B current sensor

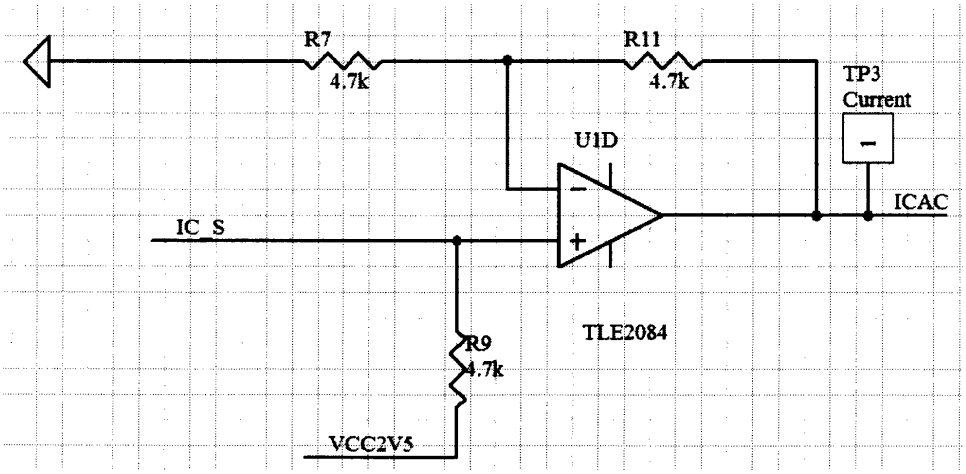

Figure B.27: Anti aliasing filter and level shifting for Phase C current sensor

B.7 Relay

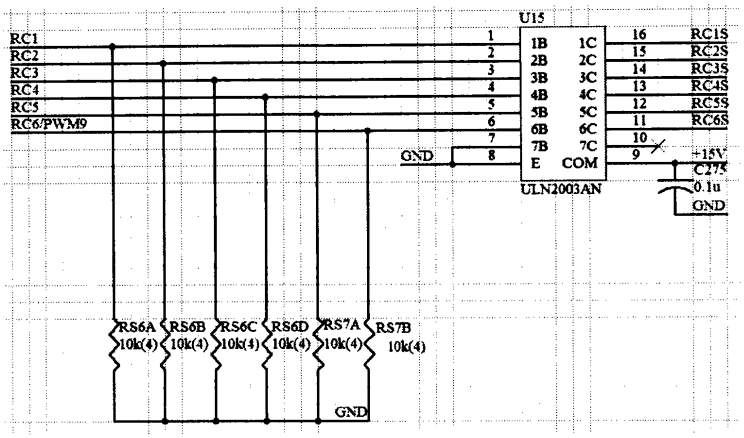

Figure B.28: Relay drivers Signals

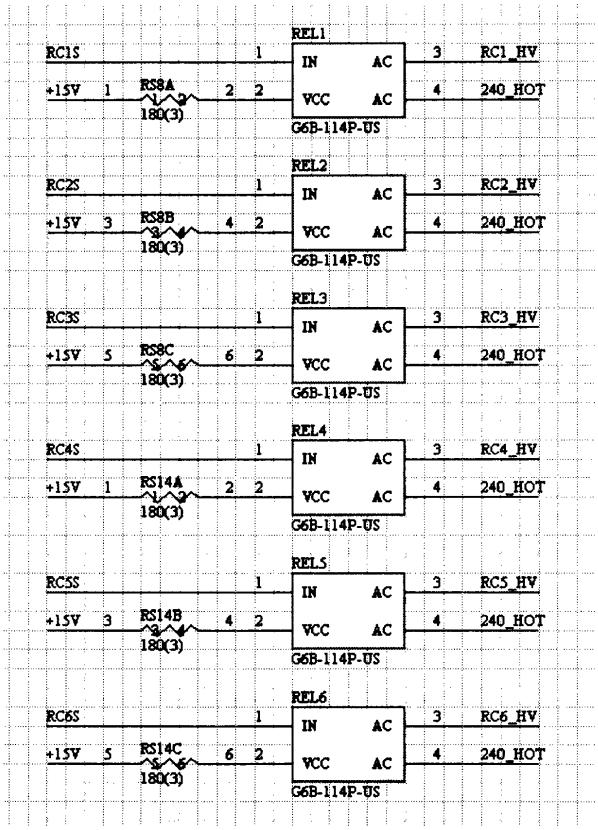

Figure B.29: Relay circuits

B.8 Clock circuit

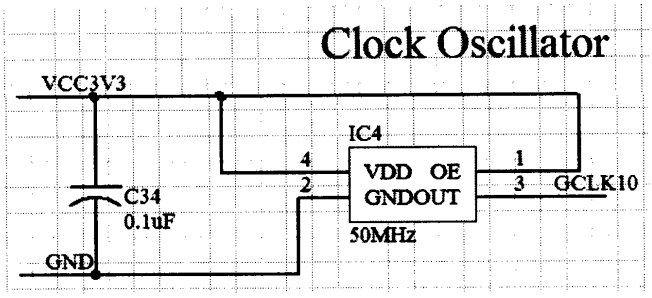

Figure B.30: Clock circuits

B.9 Phase Select

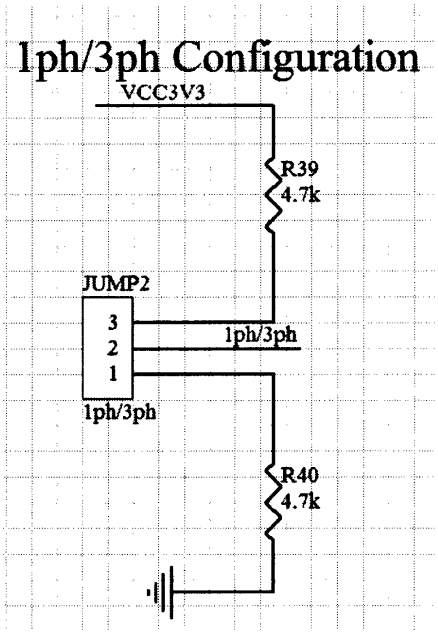

Figure B.31: Phase select circuits

B.10 LCD Displayer

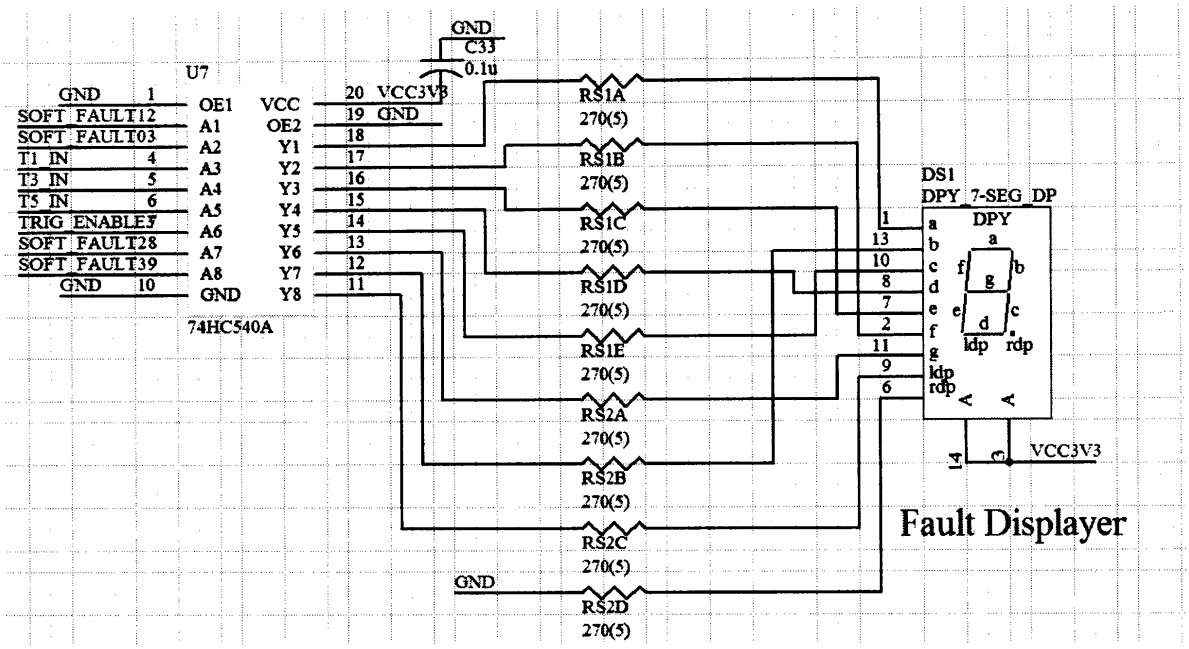

Figure B.32: LCD displayer circuits

B.11 Other

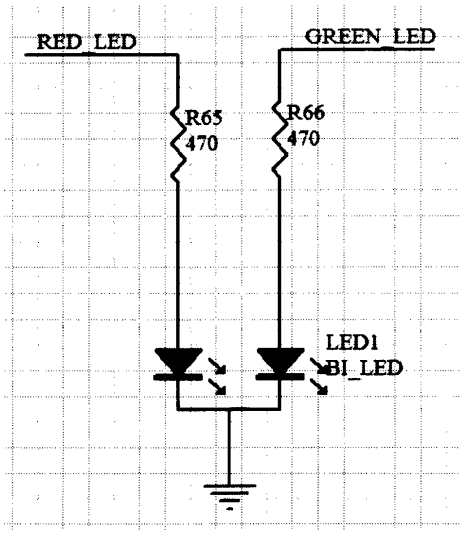

Figure B.33: Status LED

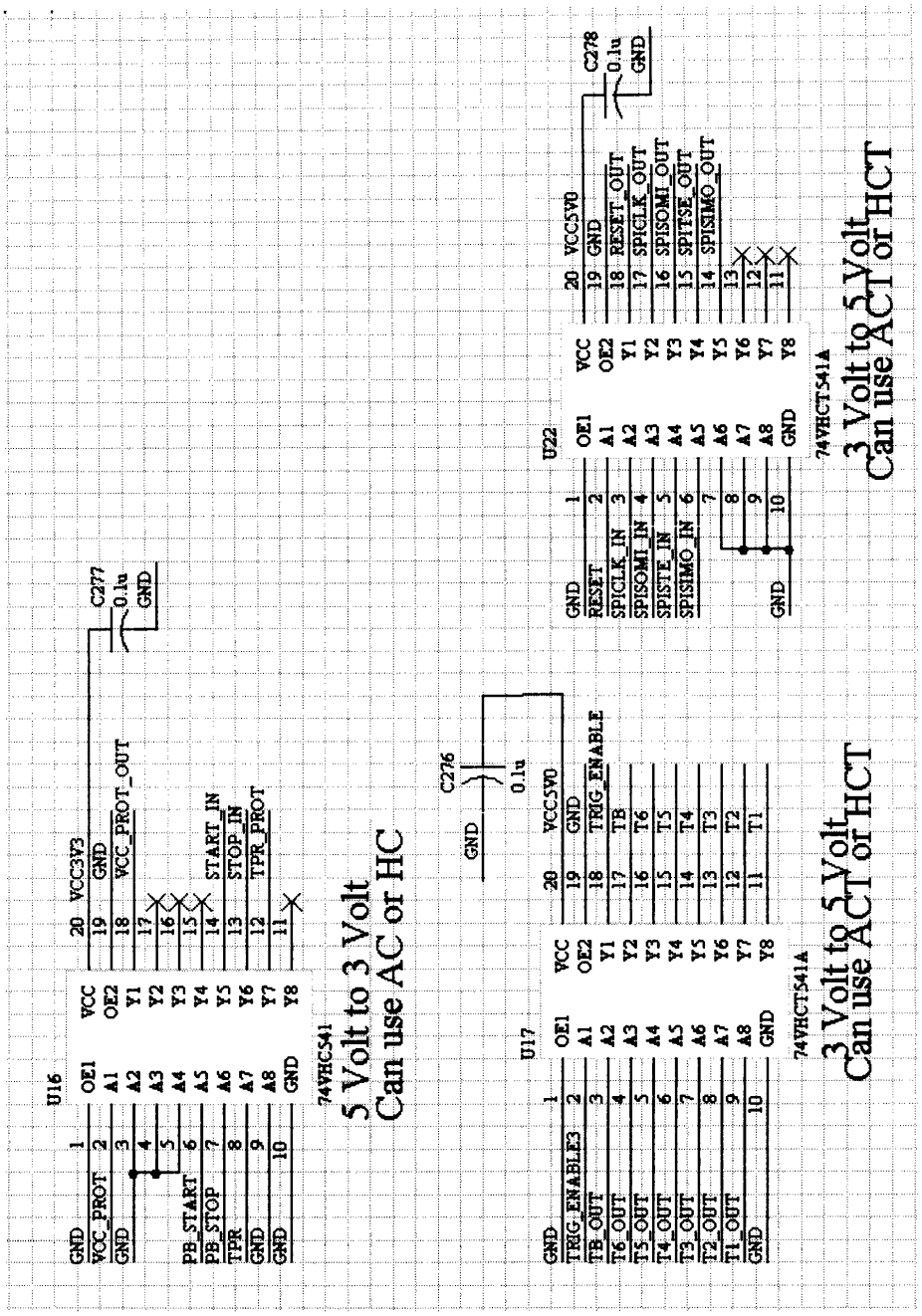

Figure B.34: Level conversion circuitry

Vita

Candidate's full name: Haider Mohomad A R

University attended: University of New Brunswick, B.Sc.E., 2009

Publications:

Haider Mohammad, Christopher Diduch, Yevgen Biletskiy, and Liuchen Chang, "Filtering Out Spikes From Sensors In Power Converters Using Discrete Wavelet Transform," *IEEE Canadian Conference on Electrical and Computer Engineering (CNSR 2012)*, 2012.