Detailed Design

Introduction

In this section, the design details which are required to build a model of the product are described. Drawings of the design will be laid out in order to show the parts required to build a product. In the bill of materials section, the component part list will be itemized. A flow diagram will also be shown in order to describe how the the parts and assemblies interconnect. The purchased component specifications show the specifications of components that can be purchased on the open market. The detailed design section summarizes the overall product. The calculations required for the performance of the product along with a detailed flow. The final section of the detailed design involves the product lifestyle. The full life of the product will be laid out from beginning to end.

Layout Drawings

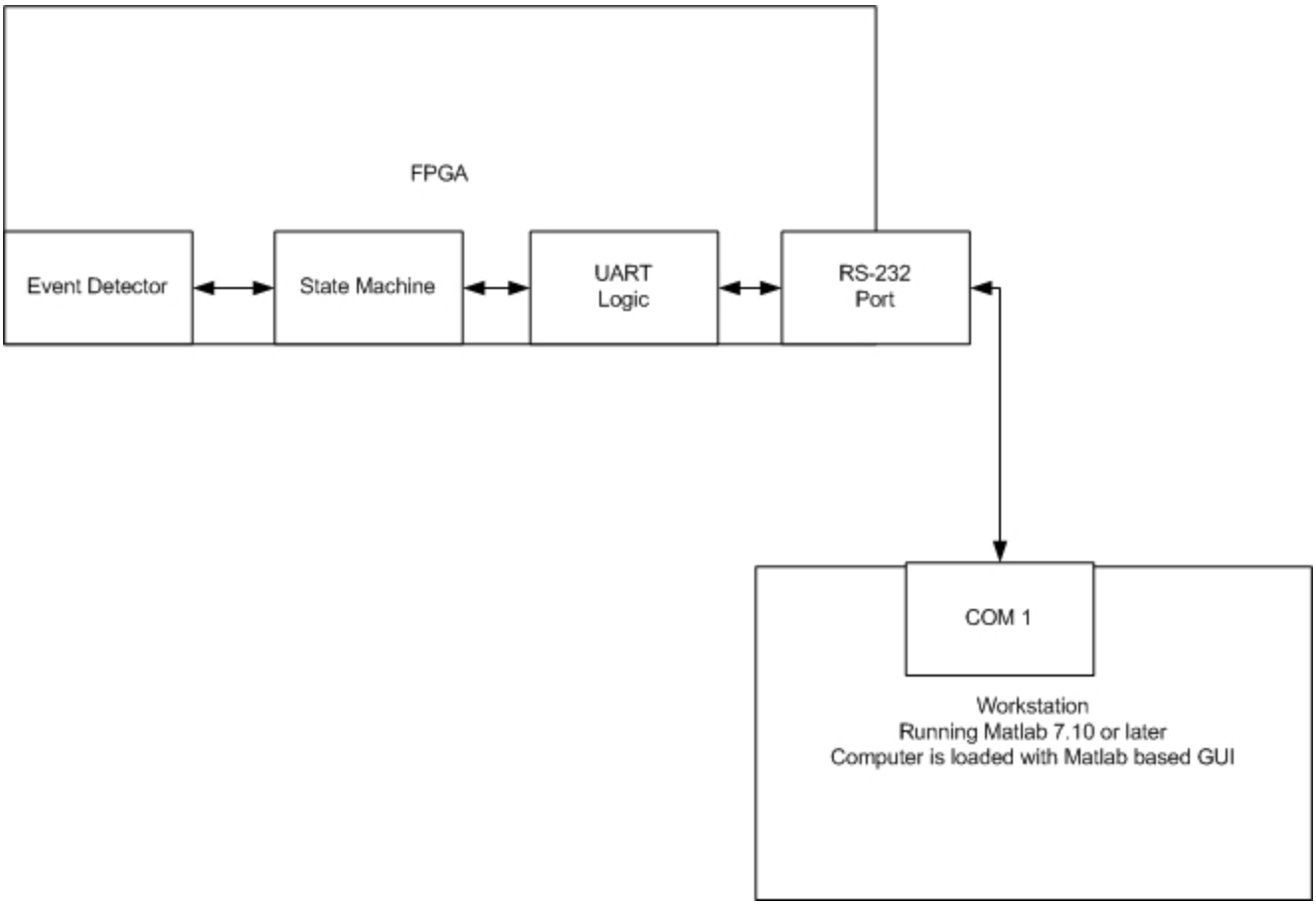


Figure 1: This is a layout drawing showing the basic interaction between the components.

Bill of Materials

Item	*Quantity*	*Item Description*
I	1	Xilinx Virtex 5 XC5VFX70T-FFG1136
2	1	Serial Cable
3	1	Workstation running Matlab 7.10 or later

Purchased Component Specifications

The only purchased component with applicable specifications is the FPGA. The FPGA is a Xilinx Virtex 5 XC5VFX70T-FFG1136. The data sheet specifications and specifications can be found at http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf.

Detailed Design

The event detector will take inputs from 8 sensor channels. The number of channels can easily be expanded to an arbitrarily large number of channels but for the purpose of this project it is being limited to 8 channels because the product demonstration will use RS-232 UART to transmit the recorded data to the computer running Matlab. The UART protocol does not guarantee that the packets are received so in order to ensure all packets sent are shown in Matlab channel information, as well as address, and location within the address must sent along with every bit. This allows for an accurate reconstruction of the received signal by the Matlab graphical user interface.

The event detector must be able to record pulses as short as 5ns in order to ensure that radiation strikes are recorded. This is being accomplished by using four D-flip flops to create a shift register that will turn the input signal into 4 parallel signals at 100MHz each showing a different time slice of the original signal. The D-flip flops will be clocked at 400 MHz. The clock speed of the flip flops was chosen because of the nyquist sampling theorem.

$$Fs >= 2B$$
$$B = 1/5ns = 200\text{MHz}$$

Fs = 400MHz

To provide the 400MHz sample clock a phased lock loop is implemented on the FPGA. This phase locked loop will use the FPGA's 100MHz clock to create a 400MHz clock to drive the flip flops. Since the flip flops will be sampling at 400MHz there will need to be four of them on each channel to then provide a 100MHz output which combines to show the entire 400MHz input

signal. This 100MHz output is then stored into a block of ram that is 4 bits wide, one bit for each flip flop, and 4 bits deep. The reason the address is only 4 rows deep is because the address and location must be transmitted in the 8 bit UART packet. Since it was decided to limit the design to 8 channels that means 3 of the 8 bits of the UART packet are used in representing the channel. Then another 2 bits are used to represent the location within the address and then 1 bit is used to send the bit value at a particular location within an address. This leaves two bits remaining to send an address meaning that the block ram must be limited to four rows deep. There will be one set of flip flops and one block ram for every channel that is being read so the design will use 8 sets of flip flops and block ram.

The block ram will use a counter running at the FPGA's primary 100MHz clock to control the write address for the block ram. During an event acquisition all block rams are being written to simultaneously with one address counter is setting the write address for all of the block rams. The block rams then have an output where the read address is being controlled by a state machine that acts as the interface between the UART and the event detector. The state machine will generate the address and that it wants to read then the ram will output the four bit contents of that address to the state machine.

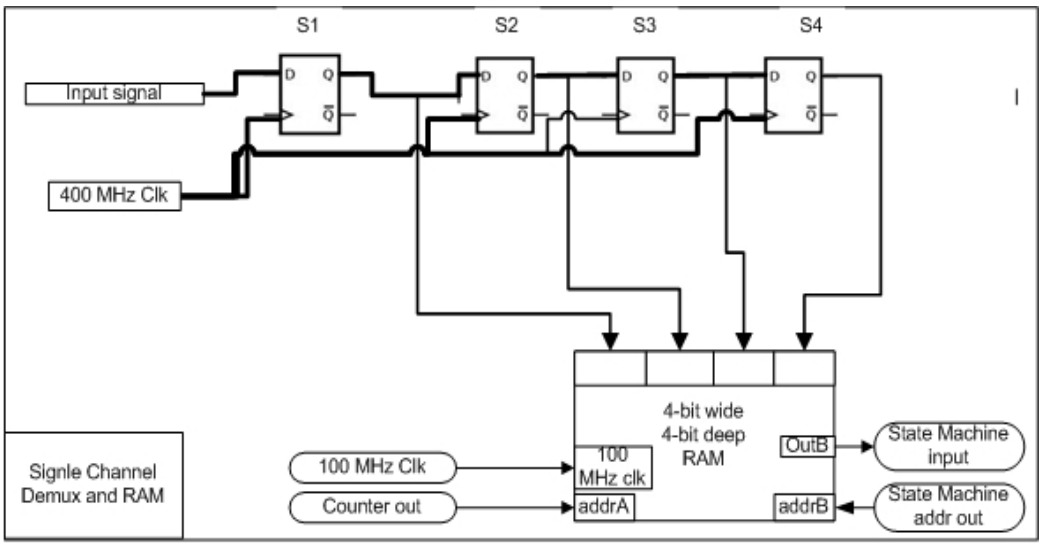
The state machine reads from one channel at a time and goes through the entire memory block and turns it into 8 bit packets that that can be transmitted by the UART block. These packets consist of the value of the bit being sent as well as a location within the address code, the address, and the channel the bit is for. This allows for a complete reconstruction on the other end by the Matlab GUI. The state machine will continually cycle through the memory of each bit of memory for each channel until until one of two events happens. The first and most common event will be a buffer full signal from the UART. If the UART buffer is full the state machine is paused until the UART buffer is no longer full at which point it will resume by transmitting the memory normally. The memory will that is being read will not advance while the state machine is being paused for a full buffer. The second event that can interrupt the output of the state machine is an acquisition signal sent from the computer running the Matlab GUI.

The acquisition signal is received by the UART block on the FPGA through the RS-232 connection. When this signal is received the UART passes the signal to the state machine and the state machine then sends a reset to the counter and enables the block ram write signal. This causes the event detector to start writing at the beginning of the memory and it will record 40 ns of events into the block ram. At this point the block ram write signal is automatically disabled. During this period the state machine goes into a standby state and pauses until the block ram write signal is disabled. Once the write signal is disabled the state machine resumes reading from memory, packaging the bits, and sending them to the UART.

The UART block reads the values sent from the state machine into its buffer then reads from

the buffer to send data over 8 bit UART RS-232 to the computer running Matlab. The user can click a button on the Matlab GUI which will then start the data acquisition process. This sends an 8 bit code to the UART and the UART will then send an enable to the state machine upon receiving this acquisition code.

The GUI will allow the user to start an event acquisition. When this occurs the GUI will send a signal to the UART and start recording the received data from the UART to fill in a table of time values for each of the 8 channels. After all data points have been received the channels are all plotted so the user can graphically see when and on what channels the events have occurred.


S1
S2
S3
S4
D
Q
Input signal
400 MHz Clk
4-bit wide
4-bit deep
RAM
OutB
State Machine
input
State Machine
addr out
100 MHz Clk
Counter out
100
MHz clk
addrA
addrB
Signle Channel
Demux and RAM

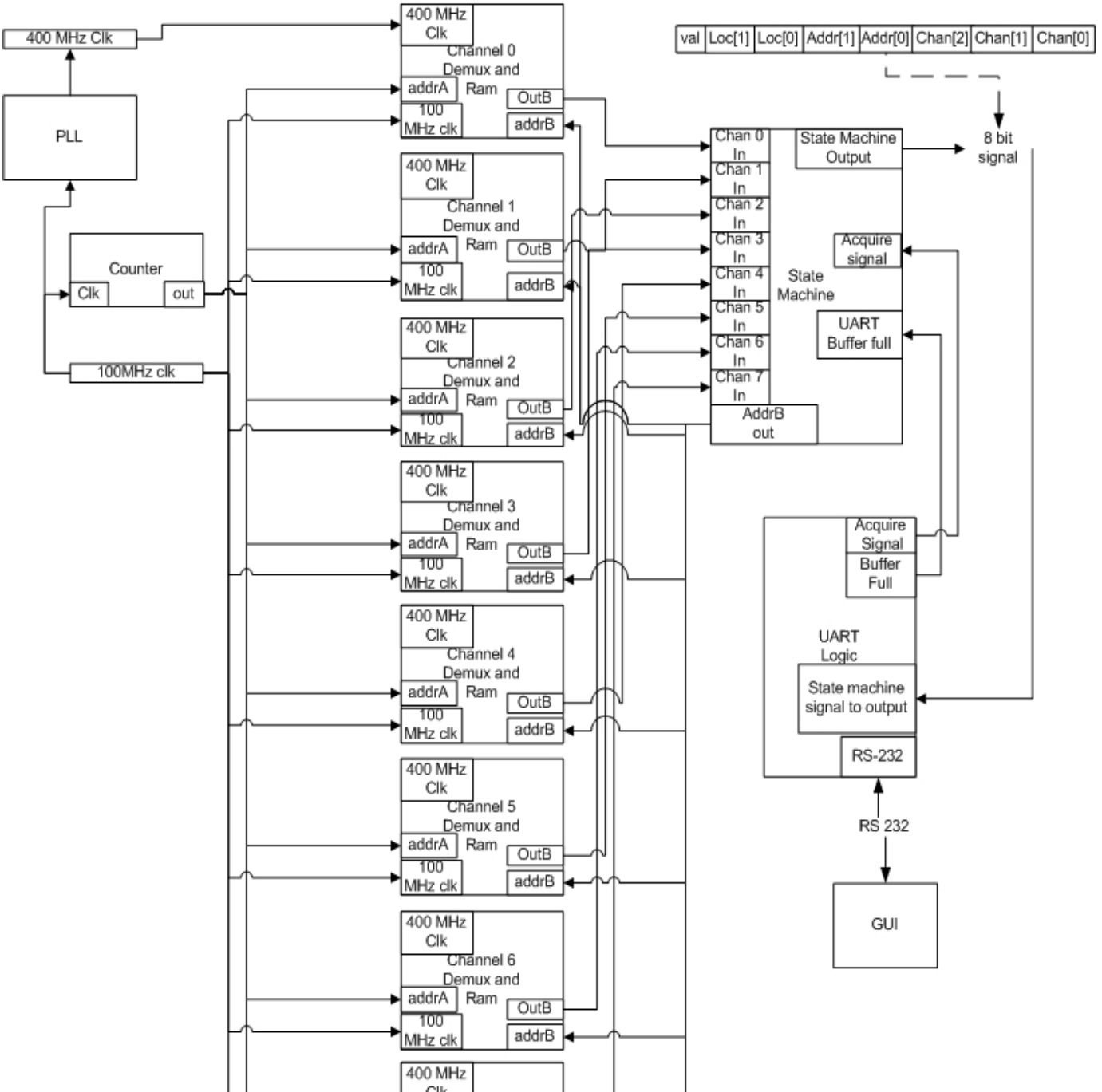

400 MHz Clk
PLL
Counter
Clk
out
100MHz clk
400 MHz
Clk
Channel 0
Demux and
addrA
Ram
OutB
100
MHz clk
addrB
Channel 1
Channel 2
Channel 3
Channel 4
Channel 5
Channel 6
val
Loc[1]
Loc[0]
Addr[1]
Addr[0]
Chan[2]
Chan[1]
Chan[0]
8 bit
signal
Chan 0
In
Chan 1
In
Chan 2
In
Chan 3
In
Chan 4
In
Chan 5
In
Chan 6
In
Chan 7
In
State Machine
Output
Acquire
signal
State
Machine
UART
Buffer full
AddrB
out
Acquire
Signal
Buffer
Full
UART
Logic
State machine
signal to output
RS-232
RS 232
GUI

Figure 2: This figure shows the connections between the pieces of the design

Product Lifecycle

The life cycle of the event detector is actually straightforward. It basically consists of three phases: the design, the operation, and the end of life. The design is the part we are currently working on. This was divided into three parts. The first is writing the vhdl for the fpga to read and store data, then the uart vhdl to synchronize the fpga to the last piece which is the graphical user interface. This is an easy way to divide because there are three team members.

After the product is assembled it begins the portion of the life cycle where it is in active use. During this portion of the products life cycle there is almost no issues to be managed with the product during this phase. As long as there is power provided to the fpga and the workstation is working fine the system will keep running. The finals stage of the product life cycle is when we are done with the event detector. This can go both ways for the product, the fpga can be used for another system, downloading another vhdl code on it to serve other functions. The cables and the workstation could also be used for another for another purpose. The other option is to recycle the products in a way that satisfies all local, state, and national regulations regarding the disposal of electronics. This is important because the electronics can have a number of heavy metals such as lead, mercury, and cadmium that often have their disposal regulated by the government.