SLAC-TN-10-007

Field Programmable Gate Array (FPGA) Based Trigger System for the Klystron Department

Darius Gray

Office of Science, Science Undergraduate Laboratory Internship Program

Texas A&M University, College Station

SLAC National Accelerator Laboratory

Menlo Park, California

August 14, 2009

Prepared in partial fulfillment of the requirement of the Office of Science, Department of Energy's Science Undergraduate Laboratory Internship under the direction of Michael G. Apte in the Environmental Energy and Technology division at Lawrence Berkeley National Laboratory.

Participant: _____

Signature

Research Advisor: _____

Signature

Work supported in part by US Department of Energy contract DE-AC02-76SF00515.

Table of Contents

Abstract

Field Programmable Gate Array (FPGA) Based Trigger System for the Klystron Department. Darius Gray, Alternative Paper, Science Undergraduate Laboratory Internship, SLAC National Accelerator Laboratory, Summer 2009.

The Klystron Department is in need of a new trigger system to update the laboratory capabilities. The objective of the research is to develop the trigger system using Field Programmable Gate Array (FPGA) technology with a user interface that will allow one to communicate with the FPGA via a Universal Serial Bus (USB). This trigger system will be used for the testing of klystrons. The key materials used consists of the Xilinx Integrated Software Environment (ISE) Foundation, a Programmable Read Only Memory (Prom) XCF04S, a Xilinx Spartan 3E 35S500E FPGA, Xilinx Platform Cable USB II, a Printed Circuit Board (PCB), a 100 MHz oscillator, and an oscilloscope. Key considerations include eight triggers, two of which have variable phase shifting capabilities. Once the project was completed the output signals were able to be manipulated via a Graphical User Interface by varying the delay and width of the signal. This was as planned; however, the ability to vary the phase was not completed. Future work could consist of being able to vary the phase. This project will give the operators in the Klystron Department more flexibility to run various tests.

Introduction/Problem Description

The Klystron Department is responsible for testing klystrons, building klystrons, and for research and development of klystrons. The department has eighteen test stations that can run klystrons (see Figure 1). These test stations are equipped with various equipment to measure voltage, current, and other aspects of the klystrons. However, much of the technology used in the Klystron Department is out dated, about 60's and 70's technology. Upgrades can be implemented to increase the capabilities of the test station. My portion of the project is to program the Field Programmable Gate Array (see Figure 2) for the trigger system to test klystrons at 18 test stands.

Materials and Methods

The materials used consists of the Xilinx Integrated Software Environment (ISE) Foundation, a Programmable Read Only Memory (Prom) XCF04S, a Xilinx Spartan 3E 35S500E FPGA (Field Programmable Gate Array), Xilinx Platform Cable USB II, a regular USB cable, a Printed Circuit Board (PCB), a 100 MHz oscillator, and an oscilloscope. The Xilinx ISE Foundation was used to write computer code in a language called **V**ery High Speed Integrated circuits **H**ardware **D**escription **L**anguage (VHDL) to perform the logic operations

necessary. The prom was programmed with the information from the computer code and loads the FPGA when power is turned on. The Xilinx Spartan 3E 35S500E FPGA receives the code written in the Xilinx ISE Foundation and performs the logical operations necessary. It, also, takes in data from a USB and an oscillator on the PCB and sends out data to the output ports on the printed circuit board. A Xilinx Platform Cable USB II was used to send data from the Xilinx software development platform to the Prom XCF04S. When the project was completed this was replaced with a USB cable that connected to the USB port on the PCB. A PCB contains the circuit elements used including the input ports, output ports, and the oscillator. The 100MHz oscillator was used as the clock to make sure that the logical operations were done in synchronization. It was, also, used to generate the signals for the triggers. The oscilloscope was used to display the output signals. The goal of the project is to generate eight trigger outputs with variable delays and pulse widths by using an oscillator, or clock, input and a trigger input.

My mentor, Ron Akre, designed a test stand trigger system that would meet the needs of the test lab. A major need of the test lab, in order to meet LCLS requirements, is 50 femtoseconds of root mean squared jitter (rms) on the output of a 5045 klystron. Jitter is the deviation, in time, of a signal from a desired location. This design consisted of two parts: 1) the central timing system and 2) the FPGA based trigger generator. The central timing system has a 119 MHz ultra low noise RF oscillator that distributes its signal with superimposed fiducials to each of the 18 test stands, which is used to create triggers. A fiducial marks a starting point in time. The operator sets the value for the delay, width, rate, and, if applicable, the phase shift value of the trigger in a user interface, developed by Kelton Stefan. The data is transported to the FPGA via a Universal Serial Bus (USB) port and the FPGA recognizes the data and sets the final attributes of the trigger. The FPGA based trigger generator will take in a signal and generate

eight independent triggers. In order to do this, the FPGA must be programmed using digital logic, VHDL, and the Xilinx ISE Foundation. The first method used to program the chip was a schematic editor; however, a switch was made to using VHDL due to its greater flexibility. The signals that come into the FPGA include the 119 MHz signal, a 360 Hz signal, and a 1 Hz signal. The operator indicates what the delay and width of the trigger is and, also, what the rate is. Inside the FPGA, the 119 MHz signal is put in sync with the 360 Hz signal. This means that the signal that results has the same pulse width as the 119 MHz signal, which is 8.4 ns, but is pulsing at the frequency of the 360 Hz signal. That signal is used, by dividing it down, to create different rep rates of 180 Hz, 120 Hz, 60 Hz, 30 Hz, 15 Hz, 10 Hz, 5 Hz, 2 Hz, and 1 Hz, which are specified by the operator. The rep rates are signals that have the same pulse width of the 119 MHz signal, 8.4 ns, but come at the operator specified frequency. One of the desires of the test lab is to be able to run and test multiple klystrons at the same time. In order to do this, they must be in phase. This is where the 1 Hz signal, or 1 Hz fiducial, plays an important role. The 1 Hz fiducial is synchronized to the 60 Hz power line that feeds the test lab. This ensures that as long as the 360 Hz signal and the rep rates derived from it are in synchronization with the 1 Hz fiducial, then they are in synchronization with the power line too. The 1 Hz signal plays a similar role as a bar line that indicates when a measure starts and stops in a piece of music played by a symphony orchestra. Depending on the time signature, there are a certain amount of beats per measure and a specific note gets the beat. The first beat after a bar line is labled 1, the second 2, the third 3, and so on until the number of beats fulfills the limit of the time signature. The termination of the count is indicated by a bar line. The first beat after that bar line is the beginning of a new count. The 1 Hz fiducial marks the starting point in time for the other signals and for the klystrons, which is synonymous to a bar line. The number of beats per measure is synonymous to the rep

rate of the pulse. Whatever the rep rate is, there are that many pulses per 1 Hz pulse. For example, if the rep rate is 30 Hz, there are 30 pulses in one second. Therefore, there are 30 pulses per 1 Hz fiducial. The 1 Hz fiducial makes sure that the triggers and the klystrons pulse at the same point in time as the bar lines in a piece of music can make sure that the musicians are at the same point in time. Having the signal's pulse width the same as one cycle of the 119 MHz signal is important. This is true because some of the digital logic elements used run on a signal that is 8.4 ns in width. If any other inputs to these logic elements have less or more than an 8.4 ns pulse width, the elements may work undesirably. If the other signals to the elements are less than 8.4 ns, the logic elements may ignore it. On the contrary, if it is greater than the 8.4 ns, the logic elements may consider it more times than desired. The final signal is a signal that the operator has the freedom to vary the delay and width of as well as the rate at which it comes (see Figure 3). This is true for six of the eight triggers. The other two triggers have a unique quality to them. One of the desires of the lab is to have two triggers where the phase of the signal can be varied on the picosecond level. In order to implement this capability, the Phase Shifter and the Delay Locked Loop (DLL) functions that are a part of the Digital Clock Manager (DCM) on the Spartan 3E FPGA are used. The Phase Shifter takes in a Phase Shift Clock (PSCLK) signal, a Phase Shift Enable (PSEN) signal, and a Phase Shift Increment Decrement (PSINCDEC) signal and outputs a Phase Shift Done (PSDONE) signal, a LOCKED signal, and the desired phase shifted signal CLK0. The LOCKED output is a part of the Status Logic part of the DCM and the CLK0 output is a part of the DLL. The DLL receives its own input clock signal called CLKIN. When the PSEN is logic 0, the PSCLK is disabled and nothing happens. In order for the phase to be shifted, PSEN must be logic 1, the PSCLK must be on the rising edge, and the PSINCDEC must be logic 1 or 0. In the later condition, a PSINCDEC value of logic 1 means to increment the

output signal and a PSINCDEC value of logic 0 means to decrement the output signal (see Table 1 for a truth table and Table 2 for information on DCM variable phase shifting capabilities) [1]. The output signal of the DLL, which is the CLK0 signal, goes into similar logic as the six triggers without phase shift capabilities. As the output signal of the DLL gets phase shifted with respect to its input signal, the final output signal, after it goes through all of the logic, is phase shifted with respect to its zero degrees phase shift position by the same amount as the CLK0 signal. As a result, the operator is able to set the delay, width, and rate of the final output signal as well as shift its phase. This DCM functionality is utilized for two of the eight triggers. There are eight channels that take the trigger signals to various locations to run tests on the klystrons. Four of these channels include the klystron modulator, the low level RF, the scope, and the computer digitizer. The operator is able to run various tests using these triggers generated by the FPGA.

Various challenges were faced while doing the project. Among these challenges, was switching from the schematic editor in the Xilinx ISE Foundation to using a computer language called VHDL. Significant work had already been done on the project when an outside source advised the use of VHDL due to its ability to be used in other software besides the Xilinx ISE Foundation. This resulted in the need to learn VHDL and convert everything that had been done thus far from the schematic editor to VHDL. This process emphasized the importance of the skill to be able to find needed resources and to be able to learn quickly in the research field. A primary source of information was the Internet and Jeff Olsen of the controls department at the SLAC National Accelerator Laboratory. The switch turned out to be for the better as the flexibility of using VHDL instead of the schematic editor was realized quickly. Other challenges that occurred include resolving many error messages given by the compiler. This process took

anywhere from a few seconds to over a day with help from other people to resolve. This process emphasized the importance of using good programming techniques to avoiding getting error messages from the compiler and that in the research field, things do not always go the way one might think they should the first time. Therefore, developing various skills and techniques related to the particular field is of upmost importance. One technique among many that is necessary is the constant testing of the project many times during the process of working on the project. This allows one to see undesirable things in the project as it progresses and helps in the process of isolating problems. This could include testing individual sections at a time. If one does everything at one time it might be hard to find where problems have occurred.

Results

After the programming of the FPGA had reached an ending point, the VHDL code for it was merged together with the code for the graphical user interface and the USB port. Once this process was complete it was tested to see if it worked the way it was supposed to, which it did. The next step was to perform phase jitter measurements. One of the channels that did not use the DCM was tested. This channel had 80 ps of jitter peak to peak, which is acceptable (see Figure 4). After that, one of the channels that used the DCM in the FPGA was tested for jitter using a 180 Hz rep rate. This channel showed 172 ps of jitter peak to peak. Furthermore, the graph on the oscilloscope had two different states each with approximately 80 ps of jitter (see Figure 5). Having two states means that instead of having one line as the signal rises to its peak amplitude,

the signal would bounce back and forth in time between two lines. The reason for this outcome is not known. This outcome is against what was originally desired; however, whether this is acceptable or not depends on what test the operators are running.

Discussion and Conclusion

There is still work that can be done in the future. Originally, the goal was to be able to have different rep rates on different channels at the same time, but ended up only being able to have one rep rate on each channel at the same time. In the future, the code can be modified to accommodate such capabilities. Also, even though the variable phase shifter function of the DCM was programmed to be able to vary the phase of the triggers, the code was not written to be able to use it in conjunction with the user interface. Future work can be done on this to get it running.

I really enjoyed the internship and the project. I believe that much of what I learned I will be using in the future. Working on this project has further encouraged me to pursue higher levels of education.

Literature Cited

[1] XILINX, Spartan-3 Generation FPGA User Guide- Extended Spartan-3A, Spartan-3E, and Spartan-3 FPGA Families, [UG331 (v1.5)]. January 21, 2009.

Acknowledgements

This project was done at the SLAC National Accelerator Laboratory. My acknowledgements go out to my mentor Ron Akre for all of his assistance, Jeff Olsen, who helped me with VHDL and the Xilinx software, my partner Kelton Stefan, and Gregorio Dalit, who helped solder parts onto my PCB board. I would, also, like to thank the U.S. Department of Energy, the Office of Science, and the SLAC National Accelerator Laboratory for creating, organizing, and funding the program. Further Acknowledgements go out to Steve Rock, SueVon Gee, Vivian Lee, and Elizabeth Smith, who ran the program, for all that they have done.

Figures and Tables

Figure 1: A test station with a klystron.

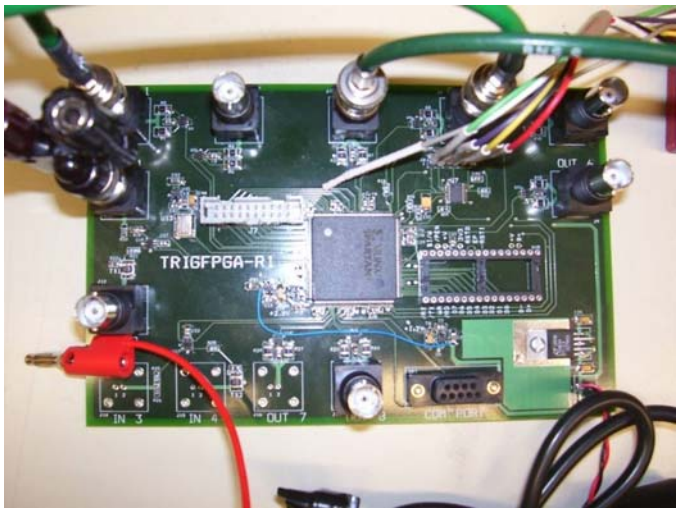

Figure 2: PCB with FPGA in the center.

PSEN	PSCLK	PSINCDEC	Output
0	X	X	Nothing happens
1	0	0	Nothing happens
1	0	1	Nothing happens
1	Rising edge	0	Decrement phase one step
1	Rising edge	1	Increment phase one step

Table 1: Truth table for the Variable phase shifter.

CLKIN Frequency	CLKIN Period	Maximum # of DCM Delay Steps	Typical Step Size	Maximum Variable Phase Shift
119 MHz	8.403 ns	+/- 81 steps	25 ps	+/- 2.025 ns

Table 2: Capabilities of the Variable Phase Shifter at 119 MHz.

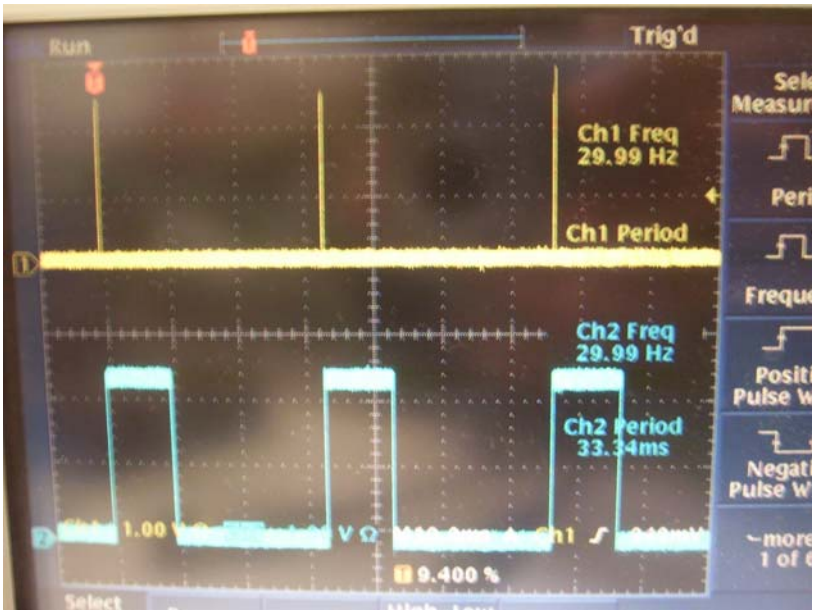

Figure 3: Waveform of one of the triggers.

Figure 4: Non-DCM Channel jitter measurement.

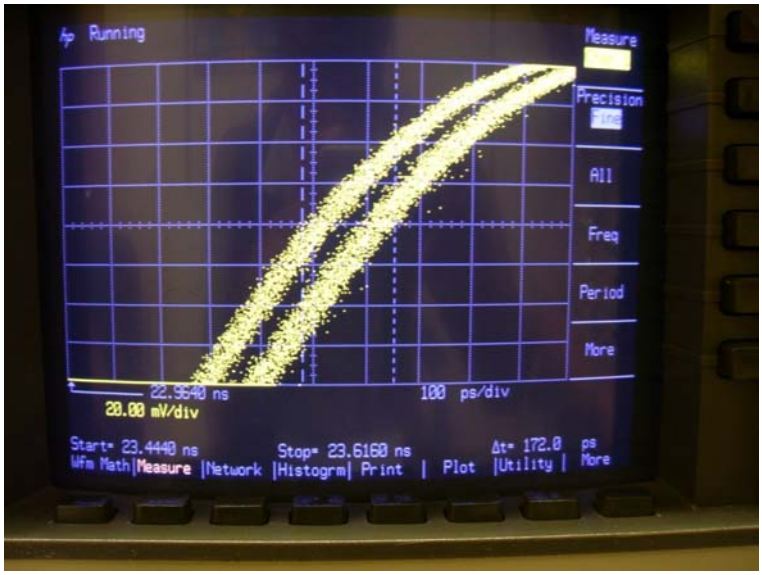

Figure 5: Channel using DCM showing jitter.