BLEKINGE TENKISKE HÖGSKOLA

Graphical Programming and Implementation of the NI-7962 and NI-5781 FPGA Interface

Paper for submission for the Bachelors Thesis in Electrical Engineering with Emphasis on Telecommunication

Anas Al-Daghestani

Mahinour AlKassem

6/29/2016

Abstract

FPGA systems can have a wide variety of applications within electrical engineering, product development, and prototyping. Their flexibility, low cost, and high performance has made it burst into the market with results that exceeded many expectations. National Instruments offers several software and hardware that integrate FPGA systems in their design and implementation. In this thesis work, a NI FPGA system is used along with LabVIEW myRIO 2014 software to run a graphical FPGA code, hence, identifying best practices that must be associated with using the software and the hardware of National Instruments FPGA interfaces and also compare different methods for programming, communication, and data conversion of the FPGA interfaces.

Keywords: Square Wave, National Instruments, NI 5781, LabVIEW, FPGA.

Primary Advisor: Anders Hultgren

Secondary Advisor: Kristian Nilsson

Acknowledgement

Firstly, we would like to thank our primary advisor and great teacher Anders Hultgren for his unrelenting support and continuous follow-up during the project. His patience, motivation, and immense knowledge are highly appreciated and were central to the completion of this project. His guidance helped us at all times, we would not have been able to finish this thesis without him.

We would also like to thank our secondary advisor Kristian Nilsson for his insightful comments and encouragements and the wider perspective that he has given us during the project.

Our sincere gratitude also goes to Ingvar Gustavsson for his help and answering all of the questions that we had for him as an expert.

We would also like to extend our thanks to Blekinge Tekniske Högskola and the Department of Applied Signal Processing for providing us with the Labs, space, equipment, and components to perform the simulations and experiments successfully.

Table of Contents

Figures

1. Introduction

The main topic of the work is to use NI5781 to generate a square wave and to write a manual that includes a detailed description of how to edit and compile three different codes, (VI's), and run the system to measure the results and perhaps. LabVIEW software was used to graphically program a code to generate a square wave from FPGA interface and also to write a code for an oscilloscope to read the measurements. Finally, a physical circuit containing an amplifier acting as a voltage buffer for any circuit to be installed was built, and the results were recorded and discussed. Finally, a user guide called "Using National Instruments NI 5781 System for Feedback Control - A User Guide" is written with information on how to run and compile the code for a square wave generator, a PID controller, and an oscilloscope.

1.1. Literature References

In this thesis report, six literature references were used from web pages and peer-reviewed articles to provide background theory behind the concepts used in the thesis. In the first two references, "NI FPGA" [1] and "FPGA Module" [2] , the webpages provide well-consiced information regarding the Xilnix FPGA systems utilized by National Instruments. In the fourth reference, "A Comparison Between the GPI and PID Controllers for the Stabilization of a DC–DC "Buck" Converter: A Field Programmable Gate Array Implementation", the authors talk about FPGA implementation of Buck Converters [3]; the content of this article helped the team understand better the parallelism of FPGA systems and the differnet ways they can be implemented in power electronics. Finally, the last three references, [4], [5], and [6], are manuals from National Instruments that helped in running the system and understanding the hardware and software used in the thesis work. The conference proceeding "Measuring the Gap between FPGAs and ASICs" [7] helped the team in better understanding how FPGA is improving and how designers must make better informed choices when running FPGA systems in terms of timing and programming.

There is work done around FPGA implementation of Function Generators in tutorials that are provided by National Instrument, such as "Waveform Generation with CompactRIO" [8] and "DDS Waveform Generation Reference Design for LabVIEW FPGA" [9]. Those two articles contain projects and VI's that builds a function generator utilizing the capabilities of the FPGA system. Yet, those do not provide any best practices for using the function generator on a physical circuit or in other applications.

1.2. Background

National Instruments has a history of simplifying engineering and software development projects for engineers and developers all around the world [1] . With the help NI LabVIEW, FPGA programming became relatively simpler to use and even more expandable for users with different knowledge backgrounds [2]. Many applications can be implemented using the NI hardware and software, such as PID Control, Function Generators, Oscilloscopes, Motor Control, and Power Electronics. With the amount of applications that can be implemented with the National Instruments FPGA systems and the flexibility of the interfaces, problems may arise if the coding and implementation processes are not done properly. Since most FPGA interfaces and DA boards that are associated with the FPGA interfaces incorporate input and output impedances, the operation of sensitive circuits connected to the FPGA interface can be affected. Thus, it is always recommended to use voltage and current buffers to avoid problems, such as over damping.

1.3. FPGA

An FPGA system in an NI 7962R is used in this project to generate the signal. FPGA is an integrated circuit designed to be configured by customers and designers after manufacturing, hence the naming "Field Programmable". Modern FPGA systems contains logic gates and RAM blocks that can perform high throughput functionalities and complex digital computations.

FPGA design and programming can be done by either HDL (Hardware Description Language) or by graphical programming. HDL programming is easier to be used in applications where a large number of inputs and outputs are utilized. This is because the specifications can be done numerically. On the other hand, the graphical programming of the FPGA helps in better understanding the functionality of the FPGA because it can be visualized on the computer screen; yet, it can be challenging if the FPGA design will contain a larger number of I/O's, due to the amount of time required to draw out every connection, port, and logical function [7].

FPGA technology continues to gain momentum, and the worldwide FPGA market is expected to grow to $3.5 billion USD by 2013. Since their invention by Xilinx in 1984, FPGAs have gone from being simple glue logic chips to actually replacing custom application-specific integrated circuits (ASICs) and processors for signal processing and control applications [5].

FPGA technology has many different benefits that would make it more attractive to engineers, researchers, and students to use in projects. Some of the advantages are:

- **Performance**: Taking advantage of hardware parallelism, FPGA systems have higher throughput than most integrated circuits and other microprocessor based systems.
- **Time to market**: FPGA technology offers flexibility and rapid prototyping capabilities in the face of increased time-to-market concerns. Motor, medical, aerospace, and many more industries utilize FPGA technology to prototype products and functions to reduce the time needed to get products and services to the market.
- **Cost**: The nonrecurring engineering (NRE) expense of custom ASIC design far exceeds that of FPGA-based hardware solutions.
- **Reliability**: While software tools provide the programming environment, FPGA circuitry is truly a "hard" implementation of program execution.
- **Long-term maintenance**: FPGA chips are field-upgradable and do not require the time and expense involved with ASIC redesign. Thus, maintenance can be within large periods of time.

2. Experiment

The following chapter illustrates the tools, methods and results from the experiments done. First, the components of the physical circuit are explained, then the NI 7962R and its interface are described, then the programming and experiment methods are elucidated, and finally the two experiments results will be shown.

2.1. The Physical Circuit

An amplifier and four SMD connecters were soldered on a breadboard to be used as a prototype to test the code and measure the output from the amplifier and verify that the NI device is providing a proper output.

2.1.1. SMD Connectors

FIGURE 2.1: GOLD PLATED MCX FEMALE CIRCUIT BOARD MOUNTABLE CONNECTOR

4 MCX female connectors, shown in Figure 2.1, were used to mount the cables well and ensure the high quality of the input and output signal in the built circuit. One MCX connecter was mounted for each channel utilized.

2.1.2. Cables

FIGURE 2.2: MCX- SMA CABLES

Four, 0.3m long, 50 Ohm characteristic impedance MCX-SMA cables, shown in Figure 2.2, were used to connect the NI5781 I/O device to the circuit; they are used for the input and the output to the circuit.

FIGURE 2.3: MCX TO BNC CABLES

Two of the MCX to BNC, shown in Figure 2.3 (50 Ohms characteristic impedance) cables were used to test and monitor the signal directly from the NI 5781 to the oscilloscope.

2.2. NI-device 5781

The Baseband Transceiver for NI FlexRIO shown in Figure 2.4, will require the 7962R FPGA module to function as an I/O device.

FIGURE 2.4: NI FLEXRIO 5781 TO BE ATTACHED TO THE 7962R

- Dual 100 MS/s, 14-bit inputs
- Dual 100 MS/s, 16-bit outputs
- 2 Vpp differential I/O (1 Vpp single-ended capable)
- 40 MHz bandwidth (-3 dB)
- External clock input and output

To utilize the full potential of the FPGA interface, the 7962R must be coupled with the 5781 I/O module shown in Figure 2.4. With the adapter modules, implementation of the physical I/O applications would be relatively easier [4]. The I/O utilized in the experiment is the differential analog output. Yet, an analog input channel is used for the user guide.

The I/O module used has high resolution and high sampling rate which can be summarized in Table 2.1

No.	**Analog I/O features**	
	Channel	***Feature***
1	Analog Output	• 16 bit resolution • 50 Ω output Impedance • 2V_{pk-pk} • 100MHz internal Sampling Clock
2	Analog Input	• 14-bit resolution • Input Impedance 50 Ω • 2V_{pk-pk} • 100MHz internal Sampling Clock

TABLE 2.1: I/O FEATURES OF THE NI5781

2.3. The complete interface

After the interface is coupled completely, most of the graphical programming and I/O configurations are done based on the 5781 module rather than the 7962.

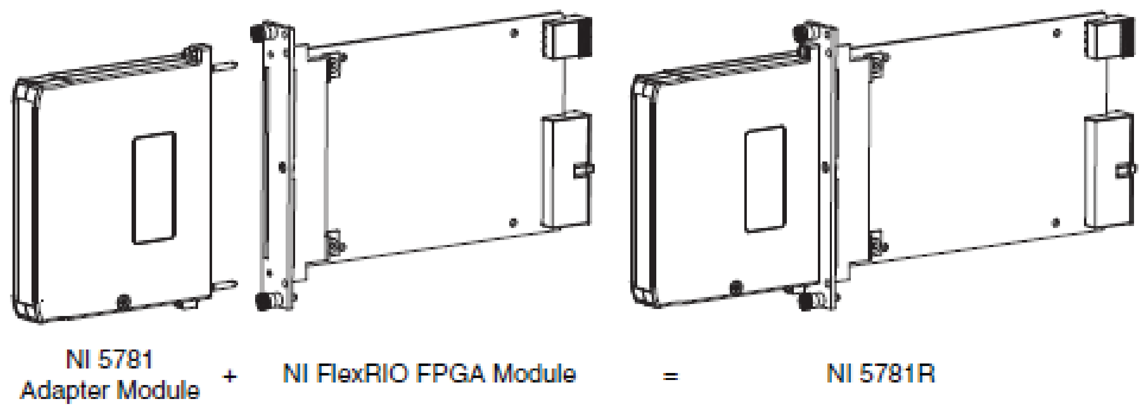


FIGURE 2.5: COMPLETE INTERFACE COUPLING OF THE SYSTEM

Figure 2.5, shows the complete coupling of the 7962R FPGA interface to the NI5781 Adapter Module as done in the experiment.

Application	Example Algorithm
Inline Signal Processing	Thresholding and Peak Detections
Custom Triggering	Logical AND/OR
Deterministic analog closed-loop control and interfacing	PID
FPGA-Based CO-processing /hardware acceleration	Algorithms exploiting FPGA based throughput and acceleration, complementing host processing

TABLE 2.2: EXAMPLES OF APPLICATIONS THAT CAN BE PERFORMED BY THE NI SYSTEM

There are several applications that utilize the features of the NI5781 and the NI7962R as shown in Table 2.2. In this project, the deterministic analog closed-loop control and interfacing, and custom triggering applications were used.

The deterministic analog closed-loop control was used for PID controller placed in the User Guide codes, and the custom triggering was used for the oscilloscope function built for reading the signal.

2.4. Graphical Programming of the Interface

To program the FPGA efficiently and ensure the quick response of the system and fast numerical calculations as well as respecting the memory restrictions of the FPGA slices and the amount of flip flops, the project code must include:

- Target (FPGA) VI
- HOST (PC) VI

The Target VI contains the code that would configure the FPGA and the processes done by the input and output channels. The timing of the process is also done on the FPGA VI, where the clock is chosen and the sequence of the code can be decided as well. On the other hand, the HOST VI, is where the numerical conversions of the input and output signals are done to improve the readability of the plots and measurements. Also, the duty cycle of the, amplitude, and the frequency of the function, in this case the square wave, can be decided. This type of data is better input in the HOST VI, since there will be no need for a totally new compilation process whenever the user decides to change some of the mentioned variables. In contrast, this might affect deterministic process of the FPGA IO, because of the buffer between the FPGA target and the HOST PC.

The target VI is compiled directly into the FPGA memory and takes around 13 minutes to complete, while the Host VI is compiled in the PC and takes a few seconds to complete. Yet, this compilation process can be simplified and the time can be reduced. The discussion section and the User Guide provide some insight in that context.

There are four different methods to program the FPGA and utilize I/O Module to implement the code, the attempted methods are illustrated in Table 2.3:

Communication method	Communicating with FPGA targets from Host	
	Control Mechanism	Common Use
Interactive Front panel communication	Front Panel of FPGA VI	· Simple logic verification · Simple debugging
Programmatic FPGA interface communication (The most common programming method)	VI running on Host	· Operations that cannot be implemented on the FPGA · Integration of FPGA, RT, and desktop components in one application

TABLE 2.3: COMMUNICATION METHODS BETWEEN THE FPGA AND HOST

Each of the communication methods illustrated in Table 2.3 have different communication speed, implementation and memory utilization.

2.5. Programmatic FPGA interface communication

With this method, it is possible to programmatically monitor and control an FPGA VI with a separate Host VI running on the host computer. This communication method is used in the project because:

- Data processing is much more than the FPGA can save.
- Perform operations that the FPGA VI cannot implement.
- Control timing and sequencing of data transfer.
- Log data and plots.

2.6. Target (FPGA) VI

Next, graphical programming was done on the FPGA target. In this block diagram, the "square wave" block was used to output a custom made square wave signal. As shown in Figure 2.6, two I/O module blocks were used to read the input and to output the square wave signal. The duty cycle blocks are also used to define the duty cycle of the wave signal and can retrieve the input from user in the HOST VI. The "Frequency" block is also used to retrieve the input from the user.

As shown in the *block diagram in* Figure 2.6, the block diagram of the FPGA VI has two SCTL structures that wok in parallel; the top structure creates the waveform and sends it to IO Module AO1, while the second structure reads the signal from IIO Module AI0. SCTL structures or Single-Cycle Timed Loop structures is a special use of the LabVIEW Timed Loop structure. When used with an FPGA target SCTL executes all functions and blocks inside within one tick of the FPGA clock that the user selected. Yet, the user cannot dynamically change the timing properties of the Timed Loop when used with an FPGA target .

To perform all the numerical operations on the FPGA interface is considered to be a challenge. This is because any change that is to be done on the block diagram in this VI would mean that the long compilation process would have to be started all over again, and that is considered to be time consuming. That is why the "Programmatic FPGA interface Communication Method" is used to ensure that all changes to the numbers and variables would occur on the desktop or the Host VI without the need for a new compilation process.

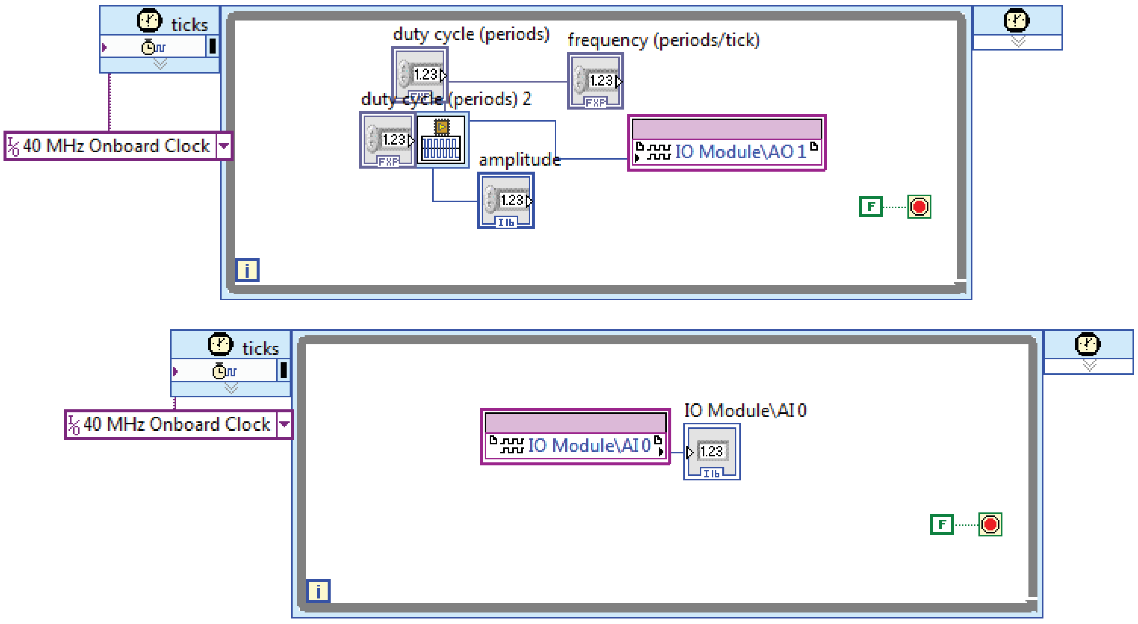
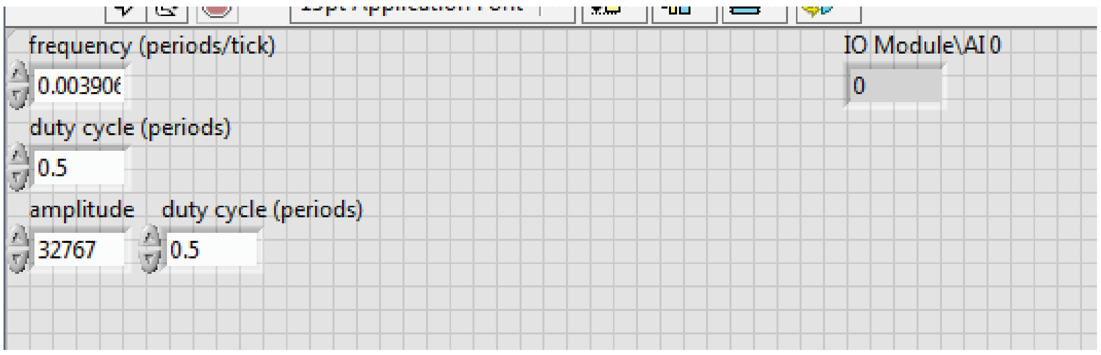


FIGURE 2.6 FPGA INTERFACE GRAPHICAL PROGRAMMING BLOCK DIAGRAM ON LABVIEW

Figure 2.7 gives an in depth look at the front panel (user interface) of the Target VI, where the user can change the variables and also start or stop the process. In this front panel window, the user can change the variables before or during the running of the code.



frequency (periods/tick)
0.003906

duty cycle (periods)
0.5

amplitude
32767

duty cycle (periods)
0.5

IO Module\AI 0
0

FIGURE 2.7: FRONT PANEL CONFIGURATION WINDOW OF THE FPGA VI

2.7. Host (PC) VI

Within the HOST VI, the user has the freedom to change the variables while the code is running without the need for a new compilation process. In here, the signal can be observed through the waveform chart.

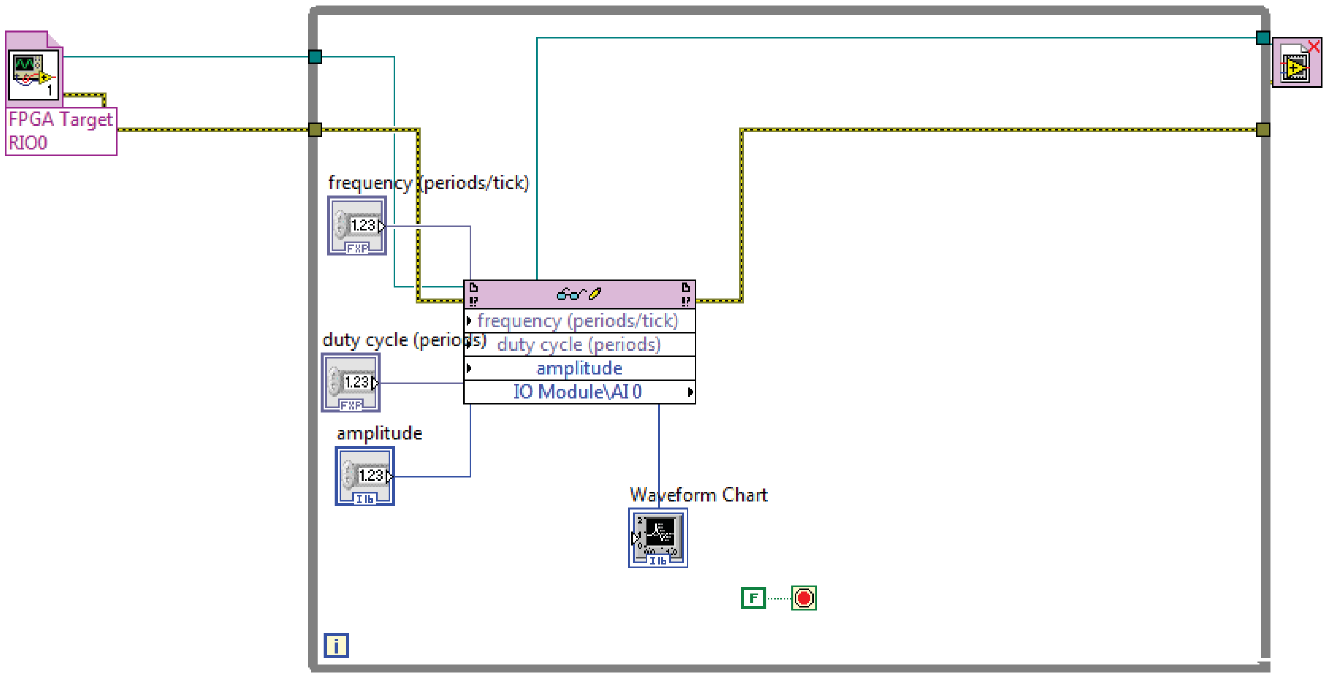


FIGURE 2.8: BLOCK DIAGRAM OF THE HOST VI

Figure 2.8 shows the block diagram including the Read/Write Block which is the direct communicating method between the Host and the Target. As seen in Figure 2.8, the read/write block has several inputs and outputs in which all the calibrated and raw data are sent and received. Several vital arithmetic is done in the Host VI that assist in reading the data correctly. It is also important to keep in mind that the difference between the resolutions of the input and the output channels requires arithmetic algorithms to ensure the calibration is done correctly.

Two kinds of arithmetic operations were done in the Host VI code to convert the integer numbers that the VI would show to the voltage values. This is because FPGA controllers cannot handle floating point numbers, hence, integer numbers are used to represent the input/output signals [6]. This representation is relative to the module used. In this case the NI 5781 has a range of -1V to +1V for the input and the output.

Property	Value
Input Value Range	2 $V_{pk\text{-}pk}$
Output Value Range	2 $V_{pk\text{-}pk}$
Input Channels Resolution	14-bit
Output Channels Resolution	16-bit

TABLE 2.4: NI5781 PROPERTIES USED FOR ARITHMETIC [6]

$$Input\ Voltage\ Value = \frac{Range\ of\ NI5781\ in\ Volts}{2^{Module\ bit\ resolution}} \tag{2.1}$$

$$Number\ for\ output = \frac{Desired\ output\ above\ lowest\ range}{Range\ of\ Module\ in\ Volts} \times 2^{Module\ bit\ Resolution} \tag{2.2}$$

By substituting the values from the Table 2.4 into the equations 2.1 and 2.2, the user can find the exact voltage or integer representation to be read easily by the user and the graphs can be illustrated clearly [6].

It is also important to perform the calculations by hand before implementing the arithmetic in the code. A mistake with the calculations and implementing the code that outputs overvoltage can cause crosstalk or damage to the module.

3. Results

In the verification process, two experiments were conducted at the beginning to measure the square wave amplitude and verify it with the actual output of the NI 5781. The physical configuration of the circuit and the results measured using the oscilloscope are shown in the following sections. For the physical experiment, the team has programmed an FPGA VI on LabVIEW that would execute the waveform in an SCTL or a Single Cycle Timed Loop. When this structure is used, all functions within the loop executes within a single tick of the FPGA clock. In this case, the NI 5781 is 40MHz. This allows for a faster rise time of the step signal since the normal *while structure* would execute at a minimum of three ticks of the FPGA clock, while functions within the SCTL would run within a single tick of the FPGA clock.

3.1. Experiment 1

The first experiment was done to verify the amplitude, frequency, and duty cycle of the generated square wave. Using the MCX to BNC cables to connect the analog output channels of the NI 5781 to the oscilloscope channels as shown in Figure 3.1.

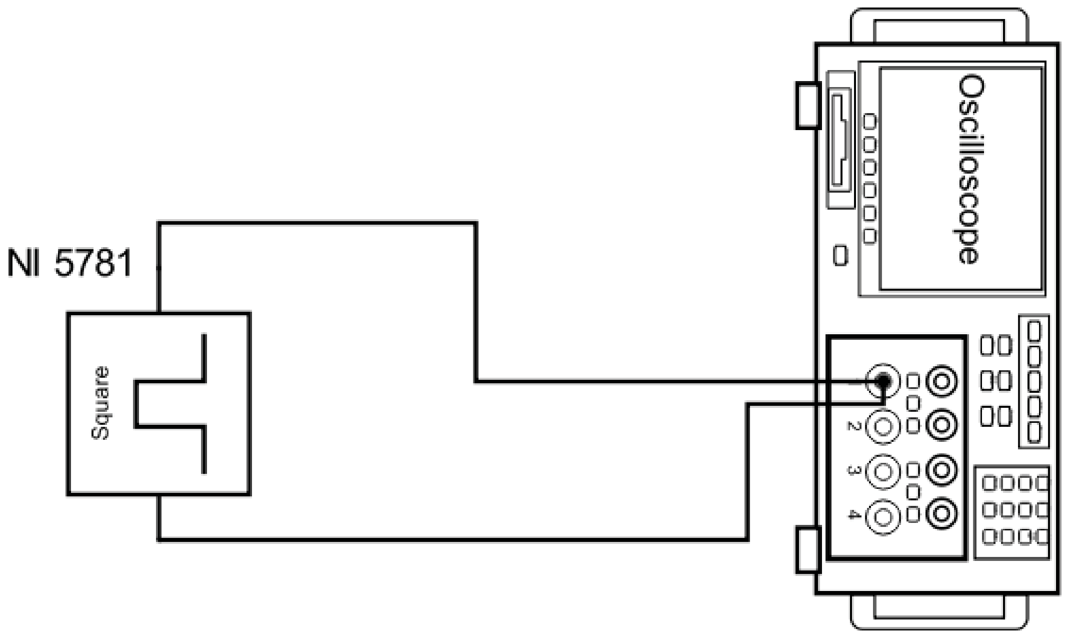


FIGURE 3.1 EXPERIMENT ONE CONFIGURATION AND CONNECTIONS

The results from the first experiment is shown in Figure 3.2: The amplitude, frequency, and duty cycle are as expected

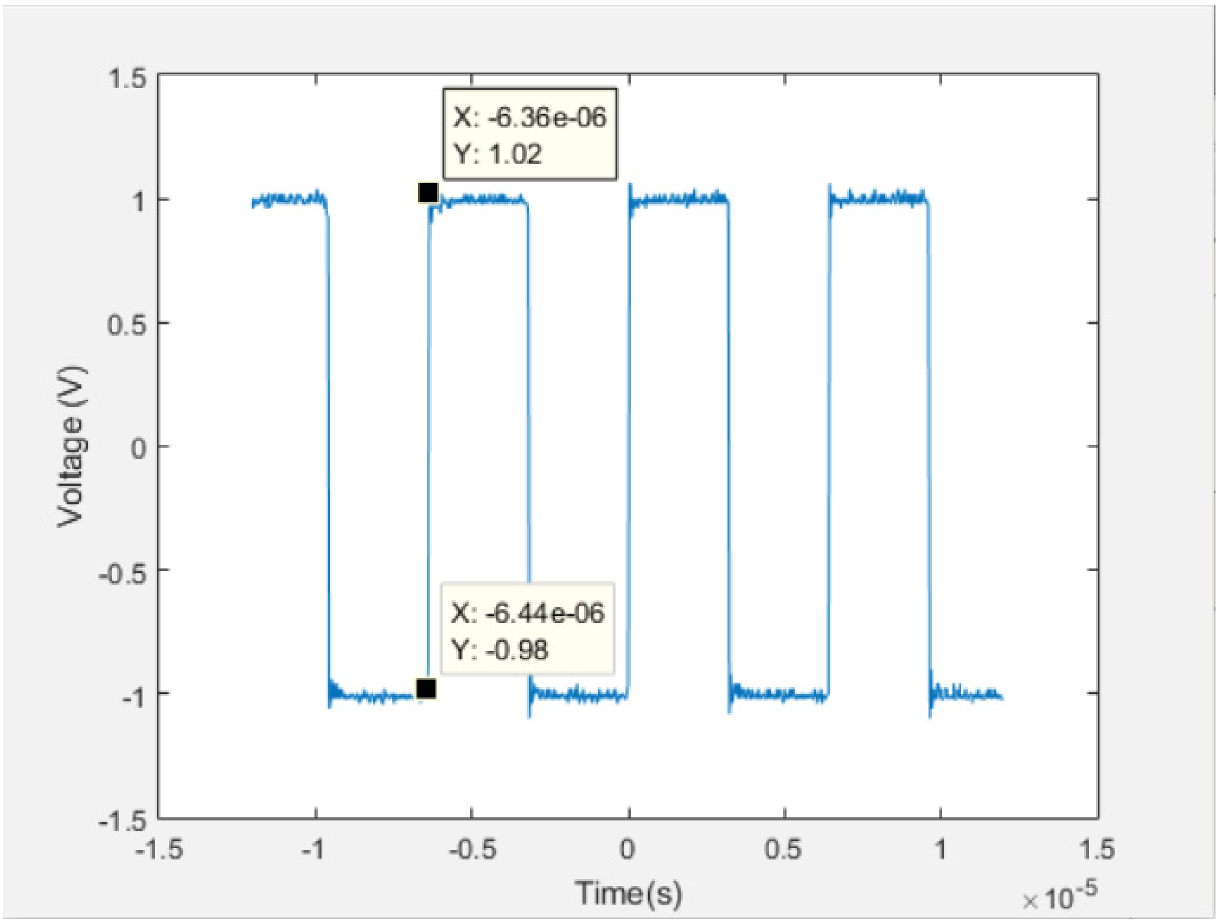


FIGURE 3.2 OSCILLOSCOPE MEASUREMENTS OF THE FIRST EXPERIMENT

The rise time calculated from the first square wave signal is 1.2×10^{-7}s

3.2. Experiment 2

The amplifier was later connected to the circuit, and was used as a voltage buffer between the NI 5781 device and any circuit that the user would install to reduce the damping caused by the impedance at the connectors. In this second experiment, the output from the first amplifier is to be measured and verified. The MCX to SMA cables were used to connect the NI 5781 Analog output channel to the circuit's SMD connectors. A passive Philips oscilloscope probe was used to measure the output from the instrumental amplifier (pin 6). The configuration is shown in Figure 3.3. The REF pin of the AD622 amplifier is connected to the GND of the voltage source where the negative pin of the probe is also connected.

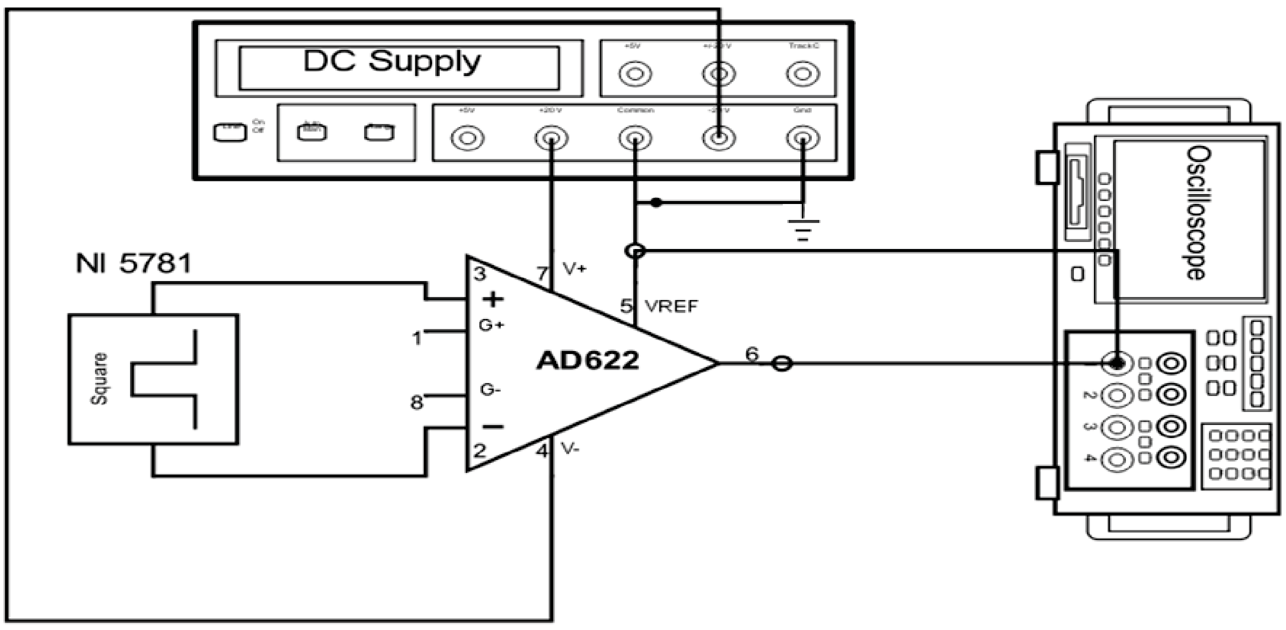


FIGURE 3.3 CONNECTIONS FOR EXPERIMENT TWO

As expected, Figure 3.4 shows that the voltage at the negative and positive channel were doubled to 2V. This is because the differential instrumental amplifier would amplify the difference between the channels, even if the output amplification is set to one.

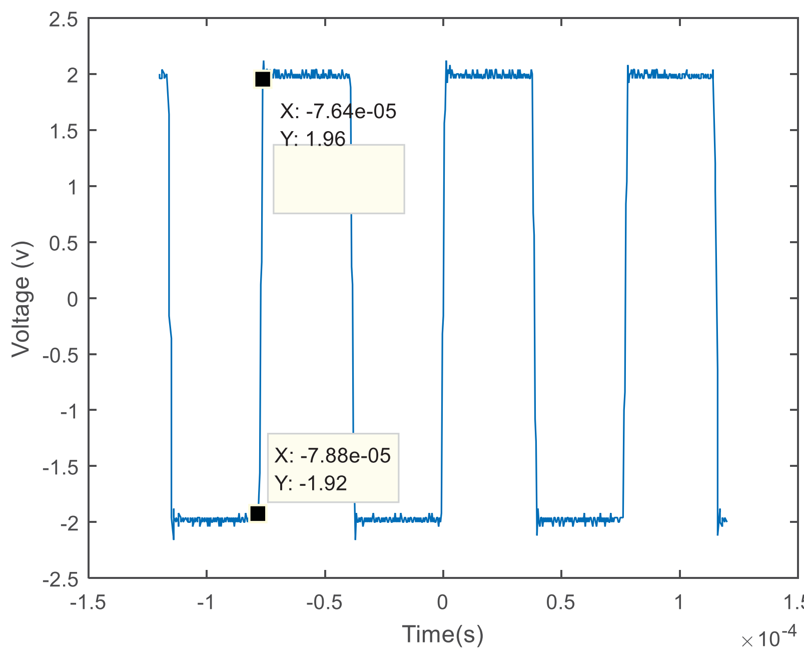


FIGURE 3.4 OSCILLOSCOPE MEASUREMENTS FOR THE SECOND EXPERIMENT

The rise time of the step signal is: **2.4×10^{-6} s**

4. Discussion

The research shows that performing experiments on the circuits using a DA/AD device with input and output impedances can be challenging, especially if the tested circuit has components that are sensitive to damping or over damping. Thus an instrumental amplifier with a very high input impedance an NI device acted as a voltage buffer amplifier so if any other circuit is connected to the NI 5781 it would not interfere with the operation of the FPGA device.

It was also important for the team to troubleshoot all the errors that occurred during the experiments by testing and reading results step by step. This helped in identifying the single point of error that would be even more difficult to pinpoint within a more complex system. Proper grounding was also important for the successful completion of the experiments; during the first experiment, the measured signal was referenced to the same ground plane as the NI Device, which caused ground loops and off measurements. Another recommendation is to always connect or terminate open channels in the multiplex system such as the one in the FlexRIO 5781. If proper termination is not done, then reflections would be mirrored in the next channels. Using proper shielding was important to eliminate any noise or crosstalk from nearby devices. Even with the relatively good quality of cables used for the experiment, having signal generators in close proximity to the NI Device has caused some crosstalk and noise which might have affected the results from the experiments if the team has not noticed it and mitigated the problem. Another problem that might cause crosstalk is an overvoltage on one of the channels; this has occurred on one or more occasions while using the differential amplifier in the experiment. Since the differential amplifier would amplify the difference between the inputs, the signal was over the recommended input voltage of the NI Device. This has caused crosstalk between channels and some misreading. This issue was solved by reducing the input voltage to the instrumental amplifier by half to take into account the amplification in the output voltage. Finally, it is recommended to use as less sampling rate as possible to eliminate a phenomenon called ghosting; which might produce unexpected voltage readings. This can only be caused when a very high sampling rate is used on multiplexed devices such as the NI 5781.

The team would recommend for users that would prototype products and services using NI FPGA to purchase a license for remote compilation using the remote compilation server. This would reduce the compilation time by at least a half, because the LabVIEW would retrieve some precompiled source code from a NI server instead of generating and compiling the entire code from scratch.

It is also important to perform the numerical conversion calculations between the integer values and the voltage values by hand before implementing the results in the code. A mistake with the calculations and implementing the code that outputs overvoltage can cause crosstalk or even damage to the module.

5. Conclusion and Future Work

In conclusion, the work done in this project can be helpful the work of FPGA systems within LabVIEW software and National Instruments FPGA hardware. Also, an explanation of the FPGA system used (NI 7962R) and the DA interface (NI 5781). It was highlighted that most of the graphical programming done on the NI system was only implemented on the NI 5781, making it easier for the user to troubleshoot the code. Four applications of the NI system were introduced and different possible arithmetic operations were explained. Two of the applications, were used in this project and the compilation process and running process are further explained in the User Guide.

In the experiment phase of the thesis, two different physical experiments were done. The first was done to ensure that the FPGA board and the DA board output the correct signals; while the other were to verify the signals and to study the use of a voltage buffer (an instrumental amplifier), connected to the FPGA interface, and observe the results from using it.

If there was no time limitation on the team, the user guide can be expanded further to include a step by step programing and explanation of every block and palette used. This can help first time users in graphical programming of NI FPGA Devices. Also, an LC circuit can be built with different components to verify the Eigen frequency and reduce the damping on the circuit by using the PID controller code in the User Guide.

Also the team can change the communication method between the FPGA and the HOST from the programmatic communication to the FIFO structure. A FIFO structure is based on the First in First out access policy, where the elements are held and transferred in the order as they are received. In some applications, FIFO structures provide more flexibility and processing speed for the user of the FPGA interface.

6. References

[1] National Instruments, "NI FPGA," National Instruments, 2016. [Online]. Available: http://www.ni.com/fpga/. [Accessed 28 June 2016].

[2] National Instruments, "FPGA Module," National Instruments, 2015. [Online]. Available: http://www.ni.com/labview/fpga/. [Accessed 28 June 2016].

[3] J. L.-F. E. G.-R. a. H. S.-R. E. W. Zurita-Bustamante, "A Comparison Between the GPI and PID Controllers for the Stabilization of a DC–DC "Buck" Converter: A Field Programmable Gate Array Implementation,," *IEEE Transactions on Industrial Electronics,* vol. 58, no. 11, pp. 5251-5262, 2011.

[4] National Instruments, "NI FlexRIO FPGA Modules," National Instruments, California, 2014.

[5] National Instruments, "Introduction to FPGA Technology: Top 5 Benefits," L.A, 2012.

[6] National Instruments, "NI 5781R User Guide and Specifications," National Instruments, California, 2014.

[7] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, New York, 2006.

[8] National Instruments, "Waveform Generation with CompactRIO," National Instruments, 30 March 2016. [Online]. Available: http://www.ni.com/white-paper/4783/en/. [Accessed 3 June 2016].

[9] National Instruments, "DDS Waveform Generation Reference Design for LabVIEW FPGA (Archived)," National Instruments, 26 January 2016. [Online]. Available: http://www.ni.com/example/31066/en/. [Accessed February 15 2016].