Final Lab ReporT

BY: Ston Yackamouih

System on A Chip Design

Submitted in partial fulfillment for the requirements of System on A Chip Design(CST 455)

Contents

[Abstract 2](#_Toc54595719)

[Design 3](#_Toc54595720)

[Synthesized Hardware Design 4](#_Toc54595721)

[Software Design 5](#_Toc54595722)

[Problems 7](#_Toc54595723)

[Results and Conclusion 8](#_Toc54595724)

# Abstract

The main topic of this lab was utilization of different IP cores. Using the different components of the De1-SoC Development kit we were able to create basic usage of mathematical functions displaying on hexadecimal displays. We also implemented the utilization of buttons to create patterns on a set of LED’s. Then we also created a Verilog module to calculate the Fibonacci sequence and find the location of a given number. The first step was to set-up the wiring for the HPS. The next step was to set up board with Linux and enable the board to run custom software. The final step was to create code that would operate the functions. Once the board was programmed, we could control the data through switches and buttons that would be outputted through the seven-segment displays.**Introduction**

This project we dove into establishing a way to use multiple IP cores, both custom and non-custom, to provide analog outputs on the device. The project was based on the utilization of the IP cores and C coding to provide the anticipated outcome of patterns on the LED’s and decimal numbers displayed on the hexadecimal displays.

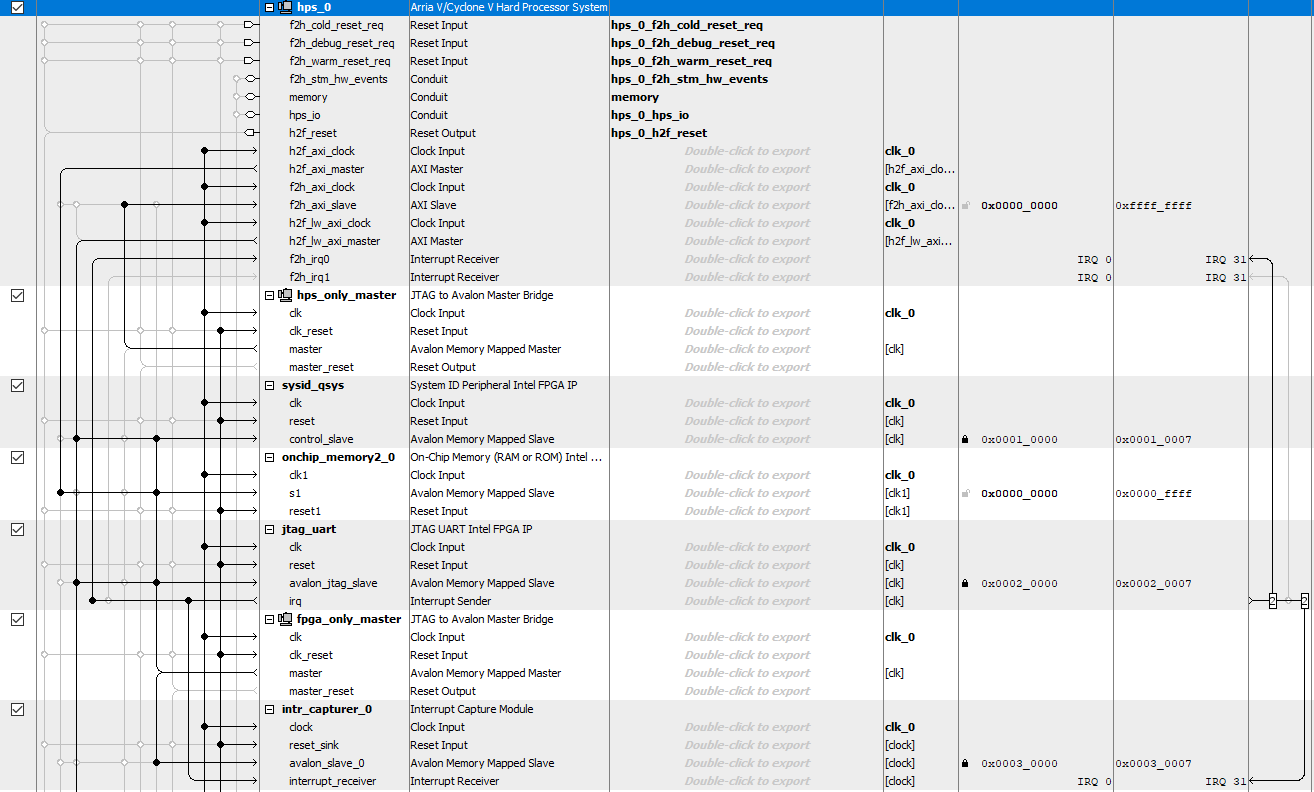
# Design

For the initial design of the Final exam, we had to start with creating basic custom IP cores. These IP cores were created in Verilog and primarily set up the initialization of the buttons, switches, hexadecimal displays, and LED’s. Also, a multiple register custom IP was created to handle the Fibonacci Sequence within the hardware. Once finished, in the Quartus Prime Project Designer we had to use these IP cores and set up the wiring for the base FPGA development as well as the base requirements to use Linux. Once the wiring has been finished, we set up the instantiation of the IP cores and made the corresponding changes to the ghrd\_top.v file and began synthesizing the project.

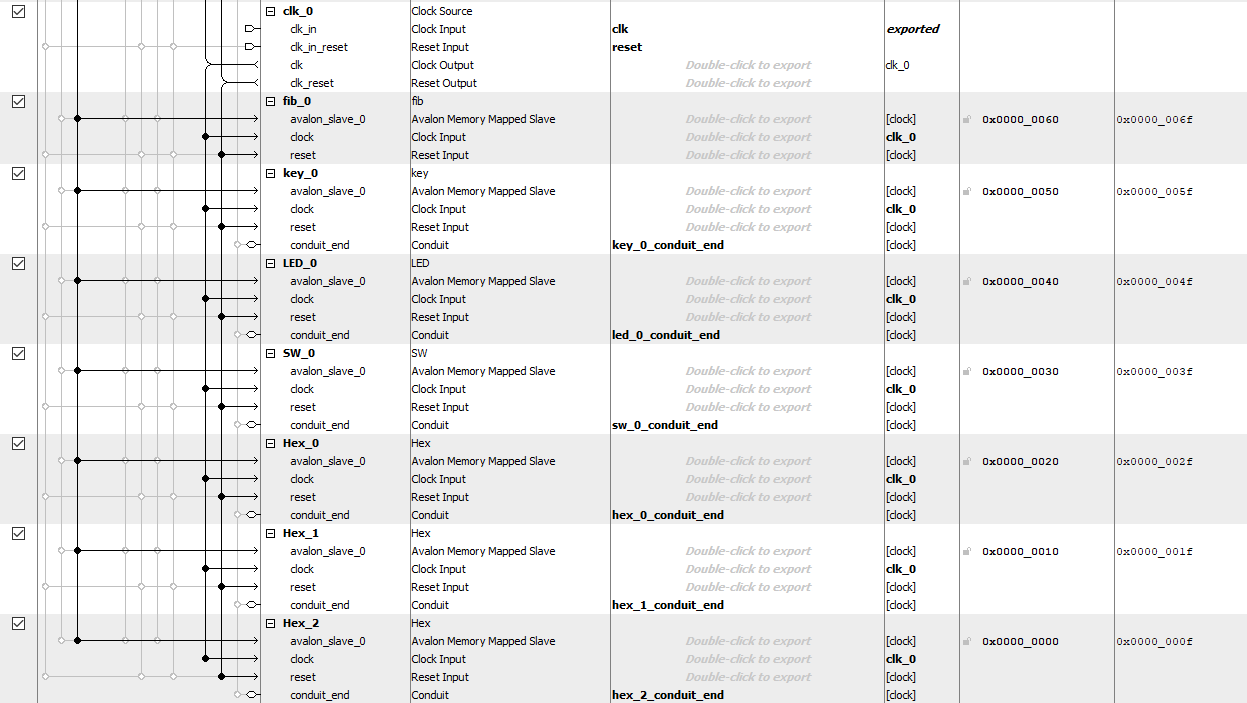
On the next section of the project we set up the files to be programmed to the board. Once finished we began the core program and utilized the libraries: stdio.h, sys/mman.h, unistd.h, fcntl.h, socal/socal.h, socal/hps.h, hwlib.h, and hps\_0.h. Once the addresses were set up the next step was to read the keys and wait for an input. Once an input was given we would check to see which key was pressed. A Key0 press would enter the function for calculating numbers base on the switches. A Key1 press would begin utilizing the Fibonacci sequence module to calculate the number in the sequence a given number is based on the switches. A Key2 press would begin a Cylon Pattern on the LED’s. A Key3 press would begin the “new pattern” function on the Led’s.

## Synthesized Hardware Design

The De1-SoC Development used ten switches, four buttons, six hexadecimal displays, and ten LED’s. The buttons had been used as inputs to light up the LED’s or light up the hexadecimal displays. The switches had been used to control the hexadecimal displays and use them for displaying decimal numbers. In the project designer of Quartus Prime we set up the base wiring for the HPS functionality along with adding components for the custom IP cores (shown in both Figures 1 and 2).

****

**Figure 1 Initial Wiring Setup Part 1**

****

**Figure 2 Initial Wiring Setup Part 2**

## Software Design

#### Main

Starting function of the program. The main function provided the setup of the hardware addresses and begin the infinite while loop. Within the while loop we read the key presses and enters the KeyControl function.

#### KeyControl

Within the KeyControl function we read the value of the keys and take four different routes depending on the button pressed. Key[0] checks the switches values then goes to the SWControl function. Key[1] checks the switches values as well and then generates the Fibonacci sequence number based on the first four bits in the switch register. Key[2] activates the cylon pattern on the LED’s. Key[3] activates the NewPattern function which generates a pattern on the LED’s and Hexidecimal displays

#### SWControl

SWControl used all of the values of the switches to generate basic mathematical functions. The function multiplied, subtracted and added based on the values on the switches. Once done it would convert the values to be legible on the displays in decimal format.

#### HexDecode

HexDecode had decoded the values sent in from a previous function to be legible for the displays to read. Once finished the decoded value is returned.

#### Cylon

Cylon created a unique pattern to be displayed. The Cylon pattern sent 1 bit to be shifted back and forth across the LED’s. The function constantly checked for a button press and exited the function whenever a new button was pressed.

#### NewPattern

NewPattern created a unique pattern to be displayed. The custom pattern started with an LED lit on opposing sides and the four outer displays quickly incrementing. As time went on the next closest LED’s lit up until all LED’s are lit. The pattern then goes in reverse and turns off the inner most LED’s and continues to turn off the next closest LED’s until all are turned off. While this is happening, the outer displays turn off and the inner 2 displays begin incrementing, until the pattern restarts. The function constantly checked for a button press and exited the function whenever a new button was pressed.

# Problems

Throughout the midterm project I ran into two different problems. The first problem was an issue with the Fibonacci Sequence. When I was initially creating the Verilog, I had used a while loop for the generation. I had forgot I was only supposed to use while loop in testbenches so Quartus gave me an error on the issue. I then reworked my logic and had small bugs such as being one off on each calculation. I later discovered I had incremented my logic one more time than I was needing to. The second issue was a bug with the switches. The switches were giving values much higher than they should have been. I later realized I had been reading the switches wrong and had to make some changes to make them correct.

# Results and Conclusion

This final helped provide a better understanding of Linux, custom IP core creation, and HPS coding. The struggles on programming produced a greater knowledge on how to code on the HPS processor. This final helped show some of the advantages of using FPGA’s while also providing a greater appreciation to Linux. Overall, the final provided a successful learning experience and helped me improve as an engineer.