**California State University, Fresno**

**Lyles College of Engineering**

**Electrical and Computer Engineering Department**

**TECHNICAL REPORT**

**Experiment Title:** Using Altera’s DE2 with the NIOS II Processor, Assignment 3

**Course Title:** ECE 178 Embedded Systems

**Date Submitted:** March 11, 2015

**Honor Code Statement:**

**“I have done my own work and have neither given nor received**

**unauthorized assistance on this work.”**

|  |  |
| --- | --- |
| **Prepared By:** |  |
| **Christopher Hays** |  |
|  |  |
| **Signature:** |  |

**INSTRUCTOR SECTION**

**Comments:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TABLE OF CONTENTS**

**Section:** **Page:**

Title Page ……………………………………………………………………...…. 1.

Table of Contents ……………………………………………………………….... 2.

1. Statement of Objectives ……………………………………………………….. 3.

2. Theoretical Background …………………….…………………………………. 3.

3. Experimental Procedure ………………………………………………………. 3.

3.1 Equipment Used ……………………………………………………... 3.

3.2 Laboratory Procedure ……………………………………………...… 3.

4. Analysis ……………………………………………………………………...... 5.

5. Conclusions ……………………………………………………………………. 8.

6. Appendix ………………………………………………………………………. 9.

**1. STATEMENT OF OBJECTIVES**

The objective of this assignment was to design an embedded system in an FPGA device and develop a simple assembly language program to run on it. The memory and logic elements were organized using *Qsys*, Altera’s system integration tool. Pre-made IP (intellectual property) cores were used to define or control the on-chip logic to perform the desired functions. Many system elements were included to build this custom soft-processor, and once the system was created, a basic program was created and downloaded onto the development board for testing.

**2. THEORETICAL BACKGROUND**

The NIOS II is a 32-bit embedded processor designed for the Altera family of FPGAs. The *Altera Monitor Program* is used to specify the NIOS II build, compile, and load any user-defined programs. This program visually represents the register and memory contents, and provides debugging tools. *Qsys* is a system integration tool that represents the various components of a system graphically, creating an easy way to make/visualize connections between components. Every piece of the system is modeled and libraries of pre-made IP cores are available to be added to a custom system. Once the system components are connected to each other, *Qsys* is used to generate Verilog code and block diagrams to be used within *Quartus II.*

**3. EXPERIMENTAL PROCEDURE**

**3.1 Equipment Used**

Altera Monitor Program Software

Altera DE2 FPGA Development Board

Quartus II

Qsys

NIOS II CPU

USB Blaster

**3.2 Laboratory Procedure**

First, a new project was created in Qsys, titled “nios”, was set to the specifications of the DE2 development board and saved. Next, a new .bdf file was created to be the top-level entry of the new project, named “nios.bdf”. This block diagram was made to hold the custom system once it was ready. Qsys was opened and the blank file (showing only a system clock) was saved as “nios\_system.qsys”. One by one, system components were added via the library of IP cores. The NIOS II CPU was added, choosing the ‘s’ option so an instruction cache was included. The JTAG debug module option was set to level 3 to allow stepping through a running program.

Two system timers were added, one in the standard milliseconds and one in microseconds. On-chip tightly coupled instruction memory (ram) and an SRAM controller were also added, with the on-chip memory designated as the main memory, with its address manually set and locked at 0x0000. A JTAG-UART core was added, using the default values. Next, various inputs and outputs were added to the system via PIO controllers; two 8-bit output PIOs were used to connect to the red and green LEDs while an 8-bit input PIO was used to connect to the slider switches and a 2-bit input was designated for the middle pushbuttons. Pushbutton 0 was used for a global system reset and pushbutton 3 is tied to a falling-edge, interrupt PIO.

IRQ priorities were assigned, with the JTAG-UART having the lowest priority, and the system clock timer the highest. All components were then connected to the system clock, the system reset, and the JTAG reset, as well as the CPU data master line. Only the SRAM, on-chip memory, and the JTAG debug module on the CPU were connected to the CPU instruction master line. The NIOS II reset and exception handler locations were set to the on-chip memory at 0x0000 and 0x0020, respectively. All PIO conduits were exported for use in QUARTUS II and the base addresses were assigned by the system. The system was then generated and ready for QUARTUS.

The new system was added to the top-level .bdf file and input/output pins were created to correspond to the exported outputs of the .qsys file. The “nios\_system.qip” file was then added to the project. Pins were created for the clock, reset, interrupt, SRAM, and the various PIO cores the controlled the lights and switches. Pin assignments from “DE2.qsf” (from the Altera website) were imported into the “nios.qsf” file to match component names with actual pins on the development board. For example, using the component name of CLOCK\_50 will always correspond to PIN\_N2, the on-board 50 MHz clock. The IO pins were named to correspond with their correct inputs and outputs. It was noted that the SRAM\_DQ bus had to be implemented as bi-directional for the SRAM to work properly. At this point the system was compiled, and using the Altera Monitor Program, was uploaded to the development board and a simple assembly program was compiled and loaded to demonstrate the switches, buttons, and LEDs.

**4. ANALYSIS**

The Qsys file generated without any errors, some warnings are shown and after research, this is considered to be a bug in the software (Figure 1). Figure 2 shows the connections between the system components.

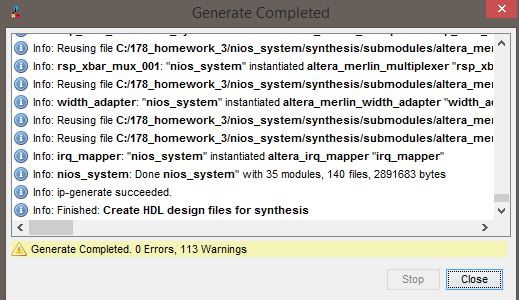


Figure 1: Generation

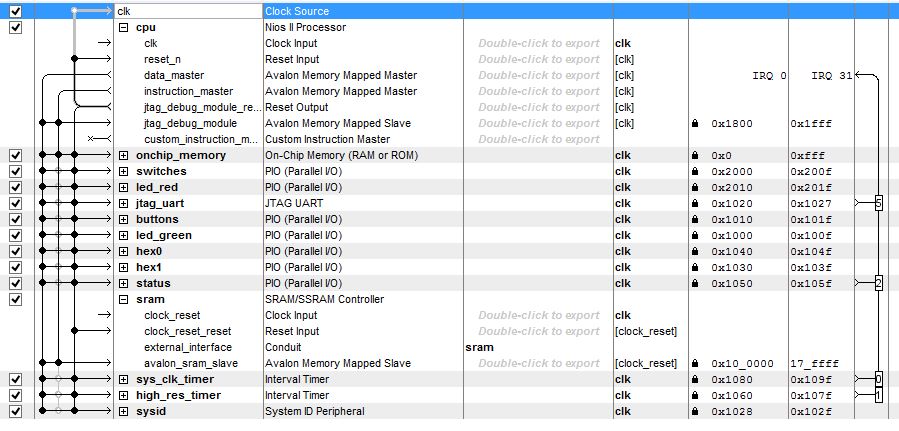


Figure 2: Connections

Figure 3 shows the block diagram and pin connections in QUARTUS II, and a successful compilation of this design is shown in Figure 4.

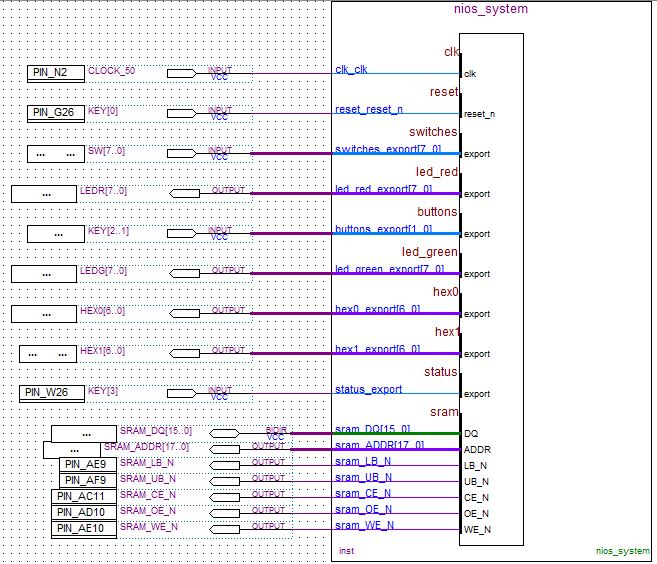


Figure 3: Quartus .BDF File

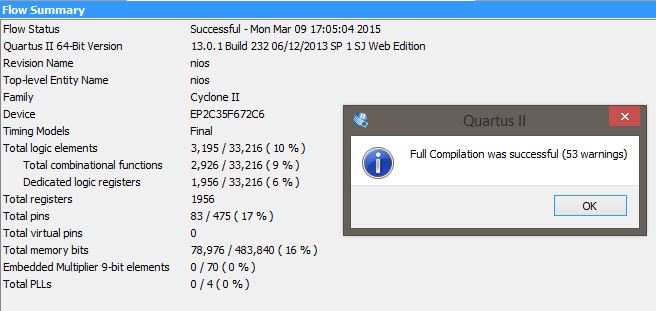


Figure 4: Compilation

Figures 5 and 6 demonstrate successful downloading of the system to the DE2 board and successful compilation and loading of the test program. Figure 7 shows the CPU registers after running the program, indicating success; they display the addresses of the PIO cores. The test program reads the data register of the switch and button PIOs and stores that information in the data registers of the red LED and green LED PIOs, displaying the status of the inputs.

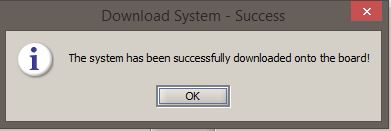


Figure 5: System Download

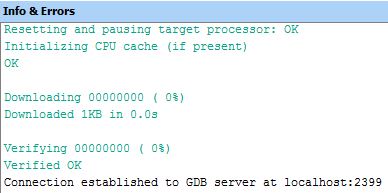


Figure 6: Program Compilation and Download

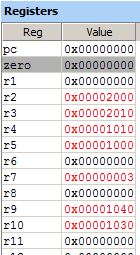


Figure 7: CPU Registers

**5. CONCLUSIONS**

This assignment demonstrated the ability to create a custom system using a NIOS II processor that will run on a DE2 development board using Qsys as a design tool. Both hardware and software fundamentals were used to realize this system, as is the nature of embedded system design. The system works as intended, and is ready to support more robust software than a simple test program. It should be noted that the original assignment required the use of PLL and FLASH controller PIOs, and using these caused errors that could not be resolved in time, although I am still working on them.

**6. APPENDIX**

**Code Used:**

.equ switches, 0x00002000

.equ redleds, 0x00002010

.equ buttons, 0x00001010

.equ greenleds, 0x00001000

.equ hex0, 0x00001040

.equ hex1, 0x00001030

.global \_start

\_start:

movia r2, switches

movia r3, redleds

movia r4, buttons

movia r5, greenleds

movia r9, hex0

movia r10, hex1

LOOP:

ldbio r6, 0(r2)

stbio r6, 0(r3)

ldbio r7, 0(r4)

stbio r7, 0(r5)

br LOOP

.end