University of Colorado
Boulder

ECEN 5863: Programmable Logic Design

Project 1

Date: 17th October 2017

Team Members:
Richard Noronha
Nelson Rodriguez
Gaurav Chhabra

EXECUTIVE SUMMARY

The aim of this project is to get familiar with the MAX10 FPGA using the DE-10 Lite evaluation kit. In this project the NIOS II embedded processor, SDRAM, on chip ram, JTAG UART, LEDs, switches etc. were implemented

OBJECTIVES

- Become familiar with the FGPA development flow, particularly in the case of a SoC with software development flow included.
- Appreciate the capability of the MAX10 to create whole systems on a chip.
- Learn how to build systems using the Qsys system design tool.
- Learn to create hardware using schematic capture input.
- Learn how to integrate software with hardware in the same device.
- Understand the rationale for each phase of the hardware development flow, including timing constraints, simulation, and programming.
- Design and build a several hardware examples using the MAX10.
- Consider hardware and software tradeoff possibilities available in the SoC architecture.

MODULE TEST RESULTS

MODULE 1A:

<u>Procedure</u>

This module uses block diagram entry to complete an FPGA project. The design consist of a PLL, debouncers and a PWM generator. The circuit controls LED intensity from the settings of three slide switches. The design is implemented on the DE10-lite development kit using Quartus Prime.

Analysis

The circuit was completed starting form a partial .bdf file. This file is provided with connection only. Components were created to complete the design. The PLL was created using the IP Catalog. The PLL was configured with a 25MHz and 30KHz clocks, and converted into a symbol file. Additionally the debouncer and PWM generator, original Verilog files, were converted into symbols using Quartus functionality. See attachments for finalized block diagram.

The completed .bdf was converted to verilog for compilation and simulation. The compilation reports are included to show Fmax and the % of utilization. Additionally, the design was simulated and the pulse width changes were verified by setting different values using Switches 0-2. Finally the board was programmed using the Quartus programmer tool.

Conclusion

The implemented design produced the desired results. Changes of the settings for switches 0-2 vary the intensity of LED0.

MODULE 1B:

This module adds functionality to the PWM circuitry. This module creates an ADC circuit to control Seven Segment display output. The design display the switch setting values on the Seven Segment display, and maintains LED0 intensity control, from the previous module.

Analysis

This module uses Qsys to create an ADC circuit. Qsys uses IPs to add parts to a design. We created an ADC circuit, including Clock, PLL, Interface Bridge and Jtag. These components were interconnected on the Qsys diagram. The completed Qsys design was generated into Verilog files and block symbol.

The products from Qsys were added to the block diagram and connected. See attachment for the finalized block diagram. The block diagram was converted to Verilog for compilation and analysis. The screen shots, attached, show FMax and % utilization.

The compiled design was programmed in the Max10 using the Quartus Programmer tool. The Quartus generated .sof file was added to the programmer. Additionally a jumper was placed, on the development kit, to enable the display of values on the Seven Segment display.

Conclusion

The implemented design produced the desired results. Changes of the settings for switches 0-2 vary the values on the Seven Segment display.

Answers to the questions:

Note : See attachments for detailed compilation reports, Fmax and % utilization.

1) Part 1: Fmax: 562.11MHz & 615.01MHz ; 450.05MHz (restricted)
2) %Utilization: <1%
3) Record observations: Reference screenshots PWM pdf. Yes it behaved as expected.
4) Part 2: FMax: (1)50.44MHz; (2) 76.02MHz; (3)500.25MHz, 450.05MHz(restricted) ; (4)572.08MHz , 450.05MHz(restricted)
5) %Utilization : 10%
6) Record Observations:Reference screenshots ADC pdf. Yes it behaves as expected.
7) Good voltmeter? No, The voltmeter could have been better since we are using only 3 switches to get the duty cycle to vary and there's less sensitivity so if there's a value in between it will be less accurate hence more bits should be used. Moreover, the value of the voltage is not accurate and raw hex numbers should definitely be converted to get exact voltage values

LESSONS LEARNED:

1a. This module gives experience in defining circuits using block diagrams. The tools, for creating block symbols from IP and Verilog files, is practical. Connecting block symbols is a good introduction to FPGA project integration.

1b. Qsys easily adds circuits to a design. This tool is practical for making and connecting more complex designs. Error and warnings are tracked as the components are added making error control more manageable. This module was a good introduction to building more complex FPGA projects.

MODULE 2:

<u>Procedure</u>

1) All the steps were followed in the module 2 question
2) NIOS II soft core processor was instantiated for the FPGA using the Qsys tool
3) The memory and IO ports were also added to the design and configured
4) Qsys is then used to generate the HDL file
5) The .sof is programmed onto the FPGA
6) The C code was compiled for the NIOS II embedded processor and run on the FPGA board
7) The console of Eclipse displayed the “Hello from NIOS II” as expected

Answers to the questions:

1) Fmax: 119.42MHz
2) % Utilization: 3%
3) Implementation can be observed in the screenshot
4) Observations of the board and behavior expectations. Reference image. The observation is as expected. The LED0 turns ON by default. This is because the switch inverts the data. The LED0 is pulled Low when the Switch0 turns on. The LED1 can be turned on by moving the Switch1 to ON position. For the demo part, the pushbuttons are used to display on the terminal.
5) The NIOS II is connected to the onchip memory. The onchip memory is the slave to the NIOS II processor. The onchip memory is the only memory element in this design. The instruction master sends the address to the onchip memory.
6) In software the IO is read and stored in a variable called switch datain. This data is then written out to the LED_BASE pin.
7) The LED is connected to the switch by exporting the pins in hardware.

LESSONS LEARNED

1) This module gives us experience in the implementation of a softcore embedded processor onto an FPGA board.
2) The various connections made and the reason why these connections were made was also learned in this module
3) How the code is to be programmed to a custom made embedded processor
4) Parameters needed to enable the device to function as needed

MODULE 3:

Procedure

Part1 Hardware:

1) For Part 1 of this module, the FPGA has to be configured with the NIOS II embedded soft processor interfaced with onchip ram, flash and sdram on Qsys.
2) Multiple clocks were interfaced to the FPGA to run the various peripherals at varying clock speeds.
3) A 1ms timer was also used in the design if needed by the user. The communication interface spi was used to connect the on board accelerometer.
4) The connections were made based on the requirements.
5) The interrupts and reset vector table are setup
6) Finally the HDL file is generated using the Qsys tool.
7) The necessary connections are made in the HDL file(here Verilog)
8) The sof file is then Programmed to the De10-Lite

Part2 Software:

1) In part 2 the C code is compiled for the NIOS II embedded processor.
2) Care should be taken about the names of the pins of the device. If the pins are not named properly, errors will display
3) The code is built and run on the device.

Answers to the questions:

1) **NIOS II questions:** The architecture is as it is because it should be configurable in the FPGA. We should be able to modify what we require for the architecture based on the implementation in the design. [1] The advantages of using an FPGA is that FPGAs are flexible and have a quicker time to market as compared to a microprocessor. They can be programmed over and over for different tasks making them cost efficient by avoiding recurring expenses.
2) The c and h files are well documented with proper tabs at the right place. The files contain comments for every line of code.
3) Observation of the LEDs after modifying the software: The terminal displays commands that the user can enter to obtain the necessary counter. The character ‘u’ is upcounter, ‘d’ downcounter and ‘3’ up by 3 counter. According to the input the counter increments. This is observed both on the screen as well as the LEDs.
4) Fmax: 65.77MHz. Highest speed clock: 80MHz for the NIOS II processor
5) %Utilization: 23%
6) Recorded observations and behavior as expected:
7) The instruction master of the NIOS II contains the instructions for the slaves. The slaves connected to the instruction master of the NIOS II are: altpll_0, altpll_1, onchip ram, onchip flash, sdram. We observe that all these devices are also

assigned slave addresses. The instruction master is used to address these memory locations. [3]

8) Exporting external connection is for making the signals visible to the top level. This is for obtaining external pins for the device. These pins need to be named appropriately so that they can be referenced later. These names must match the names in the top level Verilog file
9) Multiple clocks are used to clock the various peripheral and memory devices being used. The circuit board clock generator generates two 50MHz clocks and a 10MHz clock. These serve as clock sources to the FPGA. The PLL clocks are used to clock the ADC module(PLL 1) and the NIOS II processor, onchip ram, spi, clocking bridge, and sdram(PLL0)

LESSONS LEARNED:

1) In addition to all the lessons learned in module 2, this module gives experience in configuring the NIOS II processor with multiple memory devices.
2) Also, a spi device, jtag uart, adc were also interfaced
3) In this module the interrupt vectors were modified. The reset vector table was also modified to fit the application.
4) About half of the memory elements in the FPGA were used while 23% of the Logic elements were used. This gives us the scope of this project

CONCLUSION

In module 1, the voltmeter operates by using 3 toggle switches on the kit to increase or decrease the PWM duty cycle which drives one of the LEDs on the kit and vary the LED intensity. To this an ADC module is added which reads the PWM generated voltage and reports the results on 7 segment LEDs.

In module 2 the various modules were instantiated based on the requirements to display a string via UART. The final outcome of this module was a FPGA based soft processor running as simple C code. Finally, in module 3, many of the features of the De1Lite SoC were utilized to obtain a hardware design for a counter which also displays to the UART terminal. The outcome of this is a FPGA design which is capable of using the onboard accelerometer via SPI and also a timer if it has to be used by the designer.

REFERENCES

[1]Benefits of using FPGAS:
https://duotechservices.com/fpga-vs-microcontroller-advantages-of-using-fpga

[2]NIOS II
https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2cpu_nii51002.pdf

[3]Instruction master
https://www.altera.com/en_US/pdfs/literature/hb/nios2/edh_ed51008.pdf