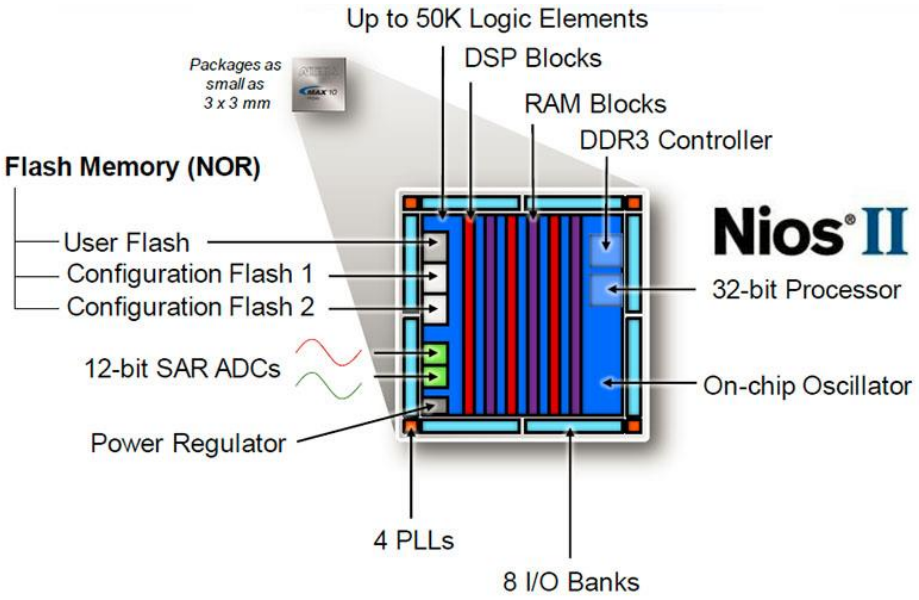
University of Colorado Boulder

Embed a Softcore Processor into Altera Max 10 and using Onboard Analog IP to Interface to Real Variables



For

Project 1 of Programmable Logic Design (ECEN-5863) at

University of Colorado Boulder, Spring 2018

By

Anay Gondhalekar and Vikrant Waje

Date: 21st February 2018

University of Colorado Boulder

Contents

Executive Summary

In this report, we aim to Embed a softcore processor into Altera MAX10 on DE10-Lite development board using Analog IP to interface peripherals and build a fully functional Embedded system. The DE10-Lite features MAX10 chip which can perform a wide variety of functions and is flexible enough to support many applications. The MAX10 FPGA is used to design wide variety of application with use of schematic and Hardware Description Language with Quartus prime software.

Module-1(Developing a Mixed signal system) deals with varying the intensity of LED's using PWM principle and then connecting the PWM output to ADC block whose output is connected to 7 segment displays. The entire system which is a functional voltmeter is controlled by switches, depending on the position of switches, the PWM output changes which in turn changes the output of ADC and the corresponding value is displayed on 7 segment displays.

Module-2 (Display Hello World and Toggle LEDs & display names using Push button using Qsys and NIOS-II) focuses on integrating Hardware and Software together on the same device board. The Hardware is based on writing a Verilog code to toggle a LED with a switch. It also gives an insight to design a system in Qsys environment to a single master and multiple slave devices to build a high efficient way of designing an embedded system. Using NIOS-II soft-processor, a software based toggling of LED with switch is implemented and displaying "HELLO WORLD" as well as name of partner on every push button press was executed with Eclipse tools.

Module -3(Create a SOC with programmable H/W and soft-processor) concentrates on building a custom processor based Embedded System. It connects the NIOS II soft processor with number of peripherals such as memory, timer, SD RAM controller, PLL to implement a counter using Qsys. Thereby, a working design of up/down, multiple of 3/5/-10 counter is implemented with visual user interface on NIOS-II console.

Thus, using most aspects of Quartus prime design flow including Qsys and NIOS-II embedded design suite, we develop an Embedded system using Real variables.

	MODULE 1	MODULE 2	MODULE 3
Objective	Developing a Mixed signal system	Display Hello World and Toggle LEDs & display names using Push button	Create a SOC with programmable H/W and soft-processor
Device Used	10M50DAF484C6GES on Altera DE10-Lite		
Fmax(in MHz)	A) 416.15 B) 50.38	119.47	86.22
% Utilization (Out of 49,760)	A) 33(<1%) B) 4734(10%)	1,626 (3%)	7778(16 %)
Registers	A) 31 B) 3233	886	4606

Module 1- Developing a Mixed signal system

Objectives

- **Project objectives**
  1. Create a PWM generator that controls intensity of LED using a set of switches. The switches are used to increase/decrease the PWM duty cycle in steps of 12.5 %
  2. Using an ADC to read PWM generated voltage and report the results on 7-segment LED's

- **Learning objectives**
  1. Become familiar with FPGA development flow
  2. Learn how to build systems using Qsys system design tool and create schematic capture input
  3. Design and build several hardware examples using MAX10

Procedure

PART A) Controlling brightness of LED using PWM technique

1. In this module we follow the procedure given in Prj1M1PWM. We start by using the Quartus 2 Software and importing the two Verilog files for debouncer, PWM generator and one schematic file named led_top.

2. We then select the MAX10 chip in Development kit named MAX10 DE10-Lite. After this is done, a PLL block is created using IP catalog option and then configuring its clock settings and various characteristics given in pdf. We now use the Verilog files that we have imported to generate symbol files.

3. The various symbols are then connected together using wires to generate the final block diagram. Having done this, we then move to section where we have to simulate the whole design. Since, the schematic file cannot be simulated, we first need to convert the schematic diagram into HDL design.

4. We compile the design and then open the Modelsim tool for viewing the waveforms in order to determine the correct functionality. After simulation, we use the sof file that is generated after compilation to program our board and verify the correct functionality on the board itself.

PART B) Using ADC

1. We start by using the Qsys tool to generate the ADC block. We follow the procedure given in the guide and repeat it for other blocks like Pipeline bridge, SEG7 block. After generating various blocks, we connect them by using wires to complete the schematic part.

2. Having done this, we will compile our design to generate the SOF file which is then used to program our development board. After downloading the SOF file onto the development board, the correct functionality is verified.

Test Results and Questions

Part A)

1. **What is the Fmax of compiled design?**
   Ans: Restricted Fmax = 416.15 MHz

2. **Total Registers(Flipflops) and percentage utilization?**
   Ans: Total Registers= 31
   Percentage utilization= 33 / 49,760 (< 1 %)

Part B)

1. **What is the fmax of compiled design?**
   Ans: Restricted Fmax: 50.38 MHz

2. **Total Registers(Flipflops) and percentage utilization?**
   Ans: Total Registers= 3233
   Percentage utilization= 4,734 / 49,760 (10 %)

3. **What could you change in either board hardware or software to make it perform better?**
   Ans: Change in hardware:
   ADC resolution: The resolution of ADC can be increased so that it can provide reading with precise accuracy

   Change in software:

   Debouncing: The software can be modified to take care of debouncing effect so as to reduce any noise that arises due to switch debouncing.

Lesson Learned

1. We demonstrated the ability of Quartus prime to generate a Schematic diagram and add various IP blocks into design. We learned how we can simulate a design to verify correct operation of the given HDL code. We learned how we can create a SOF file that can be used to program the development kit.
2. In conclusion, we can use various IP blocks to enhance the functionality of our design without having the need to design those IP blocks ourselves. For simulation, we cant use the schematic files directly but first we need to convert those schematic file into HDL code so that the Quartus is able to simulate the whole design. Also, there are two ways to program the development board, first by using the JTAG and second by using the configuration memory itself. In this module, we relied on the first approach to program our development board.

Module 2- Display Hello World and Toggle LEDs & display names using Push button using Qsys and NIOS-II

Objectives

- **Project Objectives**
  1. The main objective of this module is to create a completely functional embedded system that can display hello world, toggle LEDs with switches and display partner names based on push button press, using NIOS II and QSYS.
  2. Integrate a hardware based switch toggling to toggle LEDs and a QSYS exported software based switch toggling to the same device board.
  3. To design a system in Qsys environment to a single master (NIOS II processor) and 5 slave devices (on chip memory, jtag uart, Parallel IO-switch, led & push button) to build a highly efficient way of designing with or without the processor.
- **Learning Objectives**
  1. To become familiar with the FPGA development and design flow and create an embedded system using Altera's "soft" Nios II processor.
  2. Learn to build systems using the Qsys platform designer tool.
  3. Learn to integrate hardware and software in the same device.

Procedure

- **Hardware Design**
  1. Firstly, all the files are extracted and saved in the right folder. Then a new project is created with hello_world_lab as top-level entity.
  2. Then we select the project template from New Project as DELITE_10_Golden_Top.par file.
  3. Then, we create a Qsys design where the components- Qsys Clock, Nios II/e, On-chip Ram, Jtag Uart and Parallel IOs for switches, LEDs and push buttons were added. All the connections are made with setting the Nios II as master and others as slave to complete the connections
  4. Qsys base address, interrupt is added and reset and exception vector memory are set to onchip_memory.s1 to complete the Qsys design which is then saved and generated.
  5. Once the Qsys is done we make suitable changes in the newly added Verilog file hello_world and write the code for toggling the led and exporting push buttons and another switch-led to software.
  6. hello_world.v is set as the top-level entity and parenthetic expressions from Nios II generated system are added to the Verilog file.
  7. Thereby we complete our Hardware design.
- **Software Design**
  1. The NIOS Software Eclipse is opened which is available in Quartus.
  2. Then after selecting the workspace the working window opens. Then we select NIOS II Application and BSP from Template as hello_world small to restrict Ram size and add the hello_world.sopcinfo file and assign project name as hello_world_sw

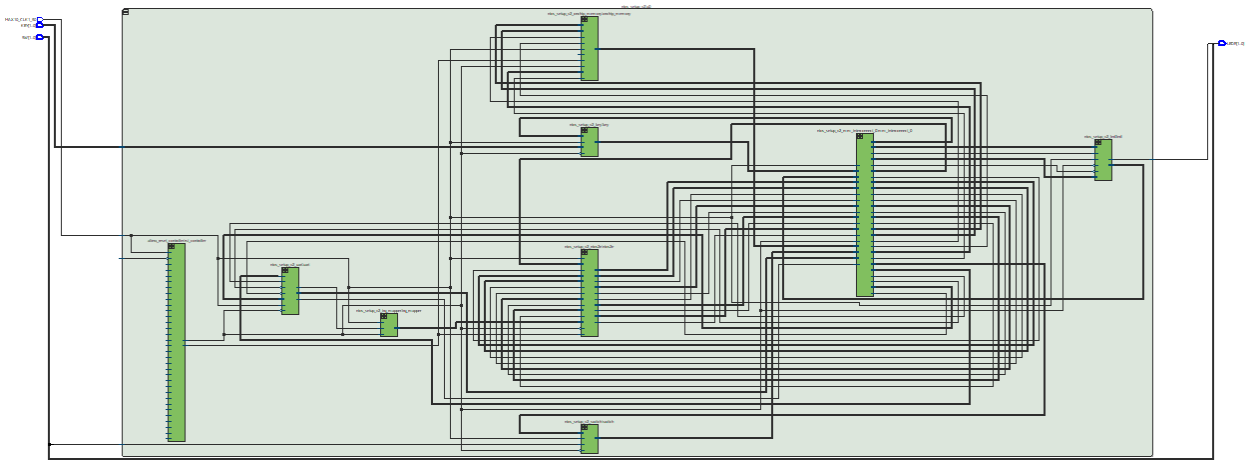
3. Then we write the code on hello_world_sw.v to toggle the exported switch and pushbuttons based partner name display as well as for writing Hello World from Nios II on the display console.
4. After making the changes we build and run the project with the generated .elf file.
5. Thus, we complete the hardware design.

Test Results and Questions

1. **The restricted Fmax for this module project is 119.47MHz.**

2. **The percentage utilization is 3% (1,626 / 49,760).1626 out of 49760 logical elements used.**

3. **Screenshot of second layer in the RTL viewer:**



4. **Record your observations of the board behavior once FPGA is programmed. Does it behave as you expected?**
   Answer: The board behaves perfectly as expected once programmed. Switch 1 toggles the LED inversely whereas switch 2 toggles the LED directly. Also, when the push buttons are pressed, partners names are seen on the NIOS II Console which also displays hello world on it.

5. **Questions:**
   1. **Which component is the instruction master of the NIOS II connected to? What is the need for this and why is it connected only to one component? (slave)**

   Ans. Instruction master is connected to Avalon memory mapped slave in On-chip memory component as the instruction master is a 32 bit bus which fetches executable instruction from executables, it has only function to fetch instruction using the Avalon memory map its connected to only 1 slave or peripheral.

2. How are the LED and the switch connected using S/W?

Ans. Altera Avalon Macros are used for reading from switch 1 and writing to the LED. Macros used are

- **IORD_ALTERA_AVALON_PIO_DATA (**for reading the input from switch 1)
- **IOWR_ALTERA_AVALON_PIO_DATA (**to write the data at the output port that is the led 1)

3. How are the LED and the switch connected using H/W?

Ans. The LED and the switch are connected in hardware by writing a Verilog code for it. 'Assign' statement is used to get inverted output between SW [0] and LEDR [0] with a not condition.

Lesson Learned

- Intel's NIOS II provides a host of useful features for FPGA development and design. As it is a soft core it provides better flexibility between software and hardware. Thereby building a fully functional embedded system with interfacing peripherals like leds, switches and push buttons was learnt from this module.
- Qsys is a next generation integration tool in Intel Quartus Prime software. It saves time by automatically generating interconnect logic to make connection between IP functions and sub systems. Thereby developing a master slave system with Qsys was learned in this module and system generation was learnt.

Module 3- Create a SOC with programmable H/W and soft-processor

Objectives

- **Project objectives**
  1. How to build a NIOS 2 processor using Qsys
  2. Downloading the design using Jtag onto development board
  3. Building a software application to design a counter(up/down)

- **Learning objectives**
  1. How to create a Qsys system
  2. How to add clock, NIOS 2 processor, memory and peripherals
  3. The purpose of bus bridge and how to implement it in a Qsys design

Procedure

1.First of all, we start the Quartus prime software to import the template from the URL specified and then start Qsys. In Qsys, we start by adding various peripherals such as SDRAM PLL, NIOS2 processor, ADC PLL, ON chip RAM, ON chip flash, Parallel I/O, SDRAM controller, Avalon crossing bridge etc.

2.We set their respective configuration parameters such as clock frequency, data width, memory size, timer etc. Also, at the same time, we start connecting various blocks together in Qsys as given in the project guide. Add a System ID to validate that the software application is being built from correct hardware system. After all this is done, we need to export signals to top level. We then generate the HDL which completes the design for our custom processor Qsys system.

3. After this is done, we can make the Qsys file as top-level entity. We compile the design, and after no errors are found, a SOF file is generated which is used to program the development kit.

4. We will then export the Embed uo block by defining it in the DE10_LITE_Default Verilog files or copying the content from some other file. Having done this, we run the Analysis and Elaboration to complete the design entry for custom processor system.

5. Now, we will program our development board using the sof files as in above two modules. This completes the hardware part of the given part. We now start with building the software application to build the ELF file which is then downloaded into memory.

6. In software part, we build a counter whose counting sequencing (Whether up or down counting) depends on the input from user which is a particular key. So by, pressing different keys, we control the up/ down counting of counter. This completes the software part of Module 3.

Test Results and Questions

1. **What is the Fmax of compiled design?**
   Ans: Restricted Fmax = 86.22 MHz

2. **Total Registers(Flipflops) and percentage utilization?**
   Ans: Total Registers= 4606

   Percentage utilization= 7778/ 49,760 (16 %)

3. **What do you think about NIOS2 architecture and what advantage does it possess by designing it with programmable logic?**

   Ans: NIOS 2 is a 32-bit embedded processor architecture which is designed for Altera family. It has many advantages over original Nios architecture, making it suitable for wide range of embedded computing applications.

   Advantages of designing with programmable logic is it offers flexibility to modify the design even after manufacturing. Thus, it does not suffer from disadvantage of being obsolete.

4. **What do you observe about coding style and use of comments?**
   Ans: The style of commenting was good and comments were given at appropriate places that provided us with good explanation of code. The coding style was also good.

5. **When you modified the software, how did the LED display change?**
   Ans: Initially, the LED were glowing in pattern of upcount i.e from 0 to 1023. However, when the code was modified, the LED's glowed in reverse pattern i.e from 1023 to 0.

6. **Record your observations of the board behavior once the FPGA is programmed. Does it behave as you expected?**
   Ans: Yes, the board behaved as per it was programmed. Depending upon the key that was pressed, the LED's were glowing in patter of either upcount or downcount.

7. **Answer the following**
   a. **Why is the instruction master of NIOS II connected to multiple components? Explain the usage of each of them.**

   Ans: It is connected to memory so that it can send request to memory or fetch instructions from memory through the Avalon bus and as the name dictates it acts as master to all the components

   Also, it serves to control every slave device to which is connected.

   b. **What's the function for exporting external connection for different components?**

   Ans: By exporting the external connection, we mean that we can get those components in Eclipse workspace and use it for building the software application.

c. What is requirement for multiple clocks in this design?

Ans: Use of multiple clocks eases the routing since a single clock won't be able to sustain to provide clock for the entire design. Also, there are various components that work on different frequencies

Lessons learned

1. Qsys provides a good interface to build the softcore processor which can be interfaced with various peripherals to make your application more sophisticated. We learned how a softcore processor can be built using the Qsys tools.
2. We got to know how to place and route the design. Another important thing that we learned from this module is how to bring the Qsys design into top level to be compiled. Last but no the least, we learned two approaches in programming our board, namely the software and hardware based approach to program our development board.

Conclusion

In this project we practiced designing, deploying, and interfacing various blocks in Qsys ecosystem. By using the Quartus prime software we could design various digital circuit using schematic and convert it into Hardware description language before simulation. We got to know the potential of Qsys design tool by modelling a softcore processor. This softcore processor is customizable, configurable, has a long life and is very high speed in design.

On other hand, hardcore processor doesn't offer the same functionality as softcore processor. Softcore processor has the advantage of migrating on to latest technology FPGA board and hence softcore processor does not suffer from any issue of obsolesce.

Given the example set in these modules, softcore processor can be developed and deployed as fully functional processor which can be as efficient as the Hard IP processor. Therefore, Quartus demonstrates that its tool suite is suitable for development of such system.

References

http://www.alterawiki.com/wiki/Category:Qsys

https://www.altera.com/en_US/pdfs/literature/tt/tt_my_first_fpga.pdf

http://www.alterawiki.com/uploads/f/fc/Hello_world_DE10_Lite.v

Appendix

Project Staff

Vikrant Waje	Anay Gondhalekar
Graduate Student Embedded Systems Engineering University of Colorado Boulder vikrant.waje@colorado.edu	Graduate Student Embedded Systems Engineering University of Colorado Boulder anay.gondhalekar@colorado.edu