EE 410 / ECE 510 Hardware Project

A 10-bit Synchronous Up/Down Counter Combined with an ALU using the MicroBlaze Microcontroller

Introduction

* **This project requires the use of the Basys3 FPGA development board by Digilent & is due 12-09-19 (Monday).**
* **UofL confirmation of your design must be verified by Eugene or Ben.**
* **WKU confirmation of your design will be verified by Ben. He will be at WKU most of the day on 12-09-19 for help and sign off.**
* **Email Dr. Naber a single hardware project report for your team.**
* **Late reports will be accepted until 12-11-19 (noon Wednesday), but 5 pts per day will be taken off.**
* **You will need to make use of the Basys3 Reference Manual for configuring the switches, buttons (de-bounced) and four 7-segment displays WITHOUT the flicker.**
* **In this project, you will combine elements of a counter design and a MicroBlaze into a single design. Then you will interface the design with most of the I/O available on the Basys3 development board. You will develop a 10-bit counter that has the capability to count up and down from 0 to 1000 (10-bit) and include a MicroBlaze microcontroller, which has the capability to communicate via GPIO and UART. The design will utilize the slide switches as inputs and it will utilize the 7-segment display and LEDs for displaying the output.**

**Objectives**

The objectives of this project are as follows:

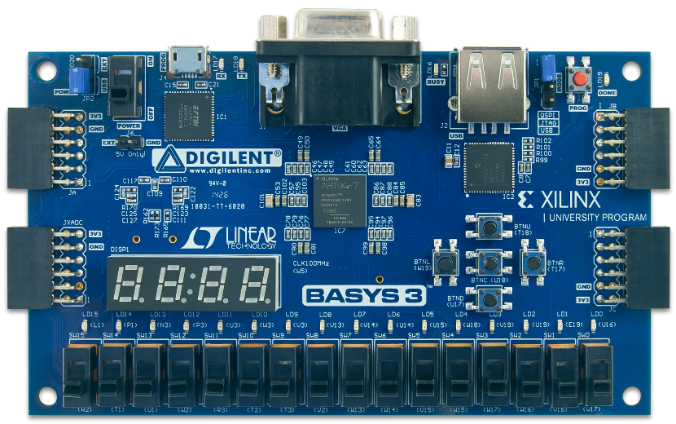
* To develop the understanding of the 7-segment display and how to interface with it.
* Show your understanding of VHDL design components such as counters and how to implement them into existing designs.
* Integrate IP core cells with your existing designs and incorporate SDK tools to drive hardware created using VHDL.
* Test your understanding of VHDL and your ability to apply concepts taught in class to real-world engineering applications.

Figure 1: Basys3 FPGA board.

**Design Description**

This project consists of two main components. A counter programmed with VHDL and an ALU based on MicroBlaze running a C code to perform ALU functionality.

1. This project requires the students to implement a 10-bit counter design using VHDL. The counter will have the ability to count up and down from 0 and 1000. This requires the utilization of 16 slide switches and the center push button of the Basys3 board. The 10 right-most switches (SW0 – SW9) will be used to input a preset value to the counter or the ALU. SW0 represents the LSB and SW9 represents the MSB of the input value.

The center push button (BTNC) is used as the manual clock for the counter. The count should increase or decrease (depending on the selector switch, SW13) by 1 on every rising edge of this button.

SW14 will be a **synchronous preset to the counter**. SW15 will be an **asynchronous reset**. If the asynchronous reset goes “high” (logic ‘1’, switch slides to the top), the count should reset to 0 regardless of the clock signal. As long as the reset stays “high”, the count should stay at 0 even if the clock is toggled or the preset is applied.

The synchronous preset should set the count value to the value on the slide switches SW0-SW9. This is synchronous, so nothing should happen to the count until the preset goes “high” (logic ‘1’) AND the clock signal (BTNC) is toggled.

1. Using the knowledge from Lab 8 implement a MicroBlaze design which has GPIO and UART capabilities. GPIO input must be 22bits wide and should read the value of the **counter** **out** and the slide switches SW11 and SW0. The GPIO output must be 16bits wide and should be assigned to 16 LEDs (LD0 – LD15) on Basys3 Board.

Reset of the MicroBlaze Clock Wizard should be connected to the SW15. Modify the C code given in Lab 7 to do the following.

Assign the **counter** **out** to the variable “numA” and assign SW0-SW9 to the variable “numB”. Then, perform different arithmetic operations based on the value of SW10 and SW11.

|  |  |  |
| --- | --- | --- |
| SW10 | SW11 | Output |
| 0 | 0 | numA + numB |
| 0 | 1 | numA - numb |
| 1 | 0 | numA \* numb |
| 1 | 1 | numA / numB |

Table 1: ALU output depending on SW10 and SW11

Display the output value on the 16 LEDs (LD0 → LSB and LD15 → MSB).

***NOTE:*** When performing the subtraction, if numB > numA, the LEDs will automatically display the two’s compliment of the result. When using the seven segment display, you can display the decimal value of the two’s compliment when you have negative numbers as your answer.

Display the values of “numA”, “numB” and the result of the selected arithmetic operation in the SDK terminal. **If the result needs more than 16bits (result > 65535), the terminal must display a message indicating that LED output maybe incorrect due to the bit width limitations.**

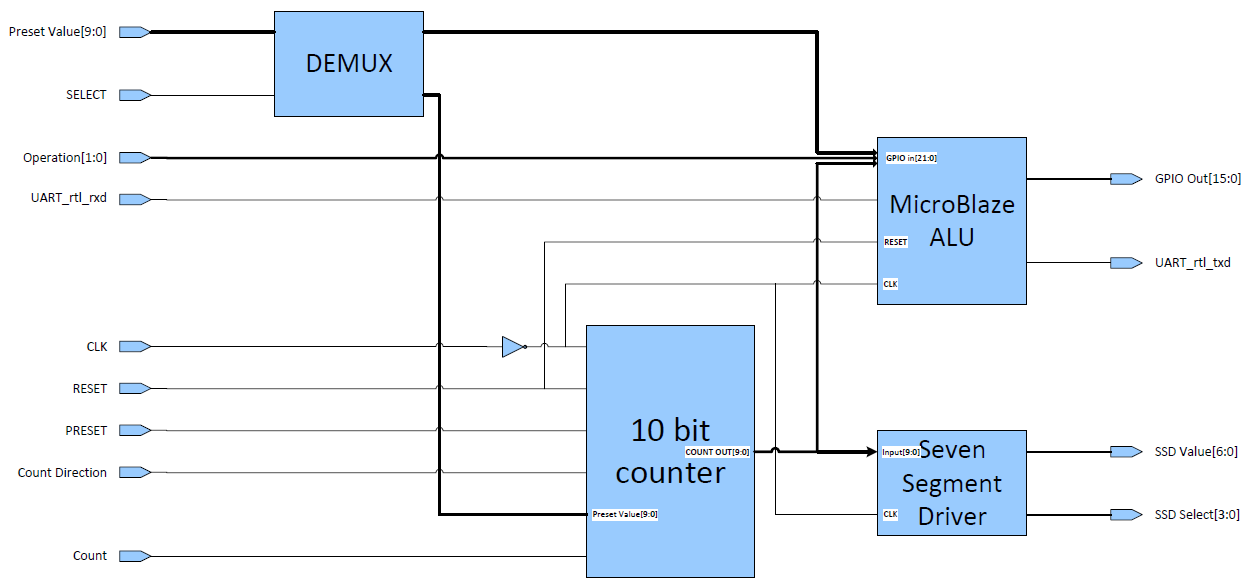


Figure 2: Simple schematic of the design, showing 4 components and the black-box outline.

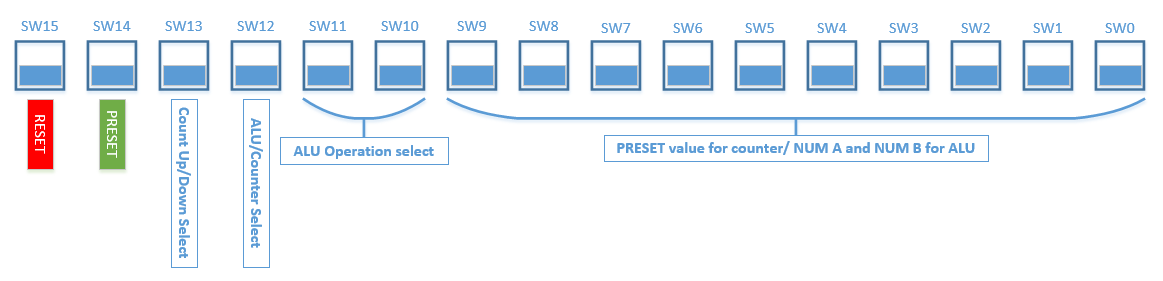
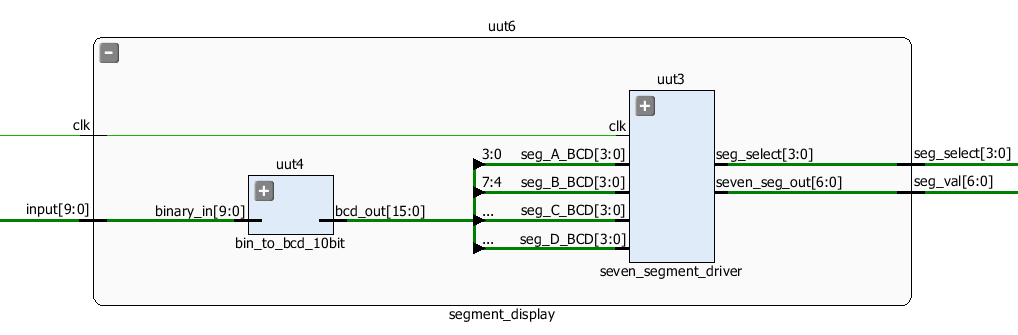


Figure 3: Slide switch map for the project

**Helpful hints:**

* **Read the CLK through a NOT gate** as shown in the figure 2 to avoid a possible error in synthesis.
* There is no need to de-bounce the slide switches. However the push button (BTNC) that you use as the clock has to be de-bounced. You can do this by simply using a D-FlipFlop. For a better de-bouncing circuit, you can use a provided language template within Vivado. To do this within Vivado, go to Window → Language Templates. Then go to, VHDL→ Synthesis Constructs → Coding examples → Misc and choose Debounce Circuit.
* It would be helpful to develop a binary to BCD conversion algorithm to display the count easier. The output of the BCD conversion should be a 16-bit long vector (4 bits per digit). Then you can perform a BCD to seven segment conversion to display the BCD value as a number in the SSD. (For more information on driving the SSD read section 8.1 in the Basys3 reference manual)



* Using a hierarchical design with multiple vhd files (one for the counter process, one for the binary to BCD conversion, one for the SSD control, etc…) would be helpful to simplify your design process. These can be called using component declarations in the architecture block before the signal declarations and begin statement.

**Project Sign-off Requirements**

In order to receive full credit for this project assignment the design must be able to complete the following:

* 1. If SW13=0 and BTNC is pushed, the SSD value must increase by 1.
  2. If SW13=1 and BTNC is pushed, the SSD value must decrease by 1.
  3. The SSD flicker must no longer be present.
  4. The count value must count from 0 to 1000 then reset back to 0. It must not count any higher than 1000.
  5. The counter reset must be asynchronous and the preset must be synchronous.
  6. If the reset switch (SW15) is toggled “high”, the count must reset to 0 immediately and stay at zero until the reset switch is toggled “low”. Once the switch is “low”, the counting process can resume as normal.
  7. The preset must be dynamic. In other words, if the synchronous preset slide switch (SW14) is set to “high” and the BTNC is pushed, the output of the counter must be set to the 10-bit binary value read in from the slide switches (SW0-SW9).
  8. If a preset value higher than the decimal equivalent of 1000 is entered using the slide switches (i.e. “1111111111” or 1023), count must be set to 1000 during preset function.
  9. The ALU must read the output of the counter (10bits) and the slide switched SW0-SW11. Counter out represents one 10bit number while SW0-SW9 represents the other 10bit number to the ALU.
  10. Based on the values of SW10 and SW11, different arithmetic operations are performed. (See table 1.)
  11. Binary value of the ALU output must be displayed in LEDs LD0 – LD15.
  12. Decimal value of the ALU output must be displayed in the SDK terminal.
  13. If the binary equivalent of the ALU out contains more than 16bits the LEDs cannot display the value accurately. Therefore all the LEDs must turn on and the SDK terminal must display a message providing a warning to the user.
  14. The SDK terminal must automatically update whenever SW11 and SW0 changes their state or the counter value changes.

**Report Requirements**

Minimum of 5 pages not including figures or appendix, which includes your VHDL code. The report must include the following:

1. An introduction with a schematic of the VHDL design from Vivado.
2. A description of the VHDL code and modifications made to existing code from Labs 7 and 8.
3. A discussion of the results and how long the project took in hours.
4. A breakdown of what each partner worked on.
5. How many iterations of this design can fit in the FPGA if I/O weren’t a limitation?
6. Any problems or obstacles that were encountered throughout the process and how you solved them.
7. The VHDL code must be submitted as an appendix to this report and does not count toward the 5-page requirement.

**Project group members must contribute equally to both the VHDL design / implementation and the final report.**