|  |
| --- |
| EXPERIMENT: A Special Purpose Processor (SPP) for Computing Euclid's GCD |

References

1. *Digital Design Using Digilent FPGA Boards ─ VHDL / Active-HDL Edition* 
   1. Example 9b: *Quad 2-to-1 Multiplexer (Listing 5.9)*
   2. Example 21: *N-bit Comparator (Listing 5.32) reference only*
   3. Example 44: *N*-bit register *(Listing 7.9)*
   4. Example 48: Clock Pulse *(Listing 7.13)*
   5. Example 52: Clock Divider *(Listing 7.18)*
2. Nexys-4 Reference Manual: *Nexys4\_rm.pdf* (available on class website)

**LAB**

**Part 1**

1. The pseudo-code for an algorithm for computing the greatest common divisor of two numbers, Euclid's GCD algorithm, is given below :

**int** EuclidsGCD (x,y)

**while** (x <> y)

{

**if**(x < y)

y = y – x;

**else**

x = x – y;

}

**return** x;

In Lab 1, you created a digital circuit with a 4-bit bus that would perform the middle part of this algorithm; it computed the difference of the smaller number from the larger number, always resulting in a positive number. Since there were no registers (memory) in the last lab, you were asked to replace the larger input with the answer from the circuit and 'run' the algorithm by hand; we used your memory. In this lab, we will add registers for the variables *x* and *y* so that the circuit can store (remember) and change their values. Figure 1 shows a block diagram for this design. Read through the rest of this lab handout so you can become familiar with the components in Figure 1. Then, **draw in the rest of the connections between components as needed and carefully label every internal signal**. This will become your *design schematic*. Note, you will also need to add a clock and reset input to the design schematic.



Figure 1: *EuclidsGCD*

1. Create a VHDL file called *RegAndMuxControl.vhd,* a behavioral design for the register control component shown in Figure 1. If both *RegisterX* and *RegisterY* are zero (i.e. they have been cleared but not loaded with anything yet), this controller should load the registers with the circuit inputs *x* and *y*. It should load *RegisterX* and not *RegisterY* from the output *z* of *SubMax* when *RegisterX > RegisterY*, and load *RegisterY* and not *RegisterX* from the output *z* of *SubMax* when *RegisterY > RegisterX*. This inherently has a stopping condition, when *RegisterX* and *RegisterY* are equal, neither will load and the answer output on the GCD will be the greatest common divisor between the two inputs.
2. Create a VHDL file called *EuclidsGCD.vhd* and implement your design by port mapping the *SubMax* component that you created in Lab 1, the *RegAndMuxControl* component from above, and two N-bit registers (Listing 7.9), as outlined in your schematic.
3. Simulate *EuclidsGCD* showing that it works correctly for the following inputs: x=15, y=6 and x=4, y=14.

**Deliverables:** Your design schematic, *RegAndMuxControl.vhd, EuclidsGCD.vhd,* *EuclidsGCD\_tb.vhd*, screenshot of functional simulationof *EuclidsGCD.vhd*

**Part 2**

1. In order to test our digital circuit on the Nexys board, create a top-level for our design to make use of the inputs and outputs (I/O) on the board. Use the switches to control the inputs *x* and *y*. This will work well since there both *x* and *y* are 4-bit buses and we have at least 8 switches on the board. For the output, *EuclidsGCD*, use 4 LEDs. We will need a clock to sequence the registers. Let's control the clock with a button so we can step through the algorithm at a slow human-like pace. Using the clock pulse component (Listing 7.13) connected to btn(2) as shown, we can create clock pulses for our EuclidsGCD component every time we press button 3. Remember from class, the clock pulse component needs a relatively slow clock to drive it. Use the clock divider component (Listing 7.18) to divide the 50 MHz master clock on the board down to 190 Hz. Remove the other clock speeds from the clock divider since we will only use the 190 Hz output. Finally, use btn(3) to clear the circuit to start with. Create this top-level, shown in Figure 2.



Figure 2

**Deliverables:** *EuclidsGCD\_top.vhd*

**Part 3**

1. Expand the top level from Part 2 to display the input switches and the output of the EuclidsGCD component on the 7-segment display using the *x7segb* component from Moodle. As discussed in class, it's not important, at this point, to know how the 7-segment display and components work. Instead, treat this component like a 'black box'. Create a new top level called *EuclidsGCD\_top2.vhd* that displays the input *x* on the second display, the input *y* on the third display and the *GCD* on the fourth/rightmost display as shown in Figure 3.



Figure 3: Top Level with 7-Segment Display

1. Demonstrate the operation of your synthesized program on the Nexys board to your lab instructor.

**Deliverables:** *EuclidsGCD\_top2.vhd*