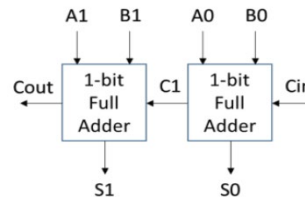



Part 1: Two-bit ripple adder design

Design, synthesize, simulate, and implement a two-bit full adder, as shown in the diagram.

Design name should be `full_adder_2bit`. Inputs are *A* and *B*. the output bus (*S*) is a two-bit bus, while input (*Cin*) and output (*Cout*) are single bits.



Note: All Verilog file names must match module names.

1. Use modular design technique using RTL modeling as follows:
 - a. Start with a single bit full-adder design using concurrent statements (logic functions) for the sum and carry (this is structural). Name this design module `full_adder.v`
 - b. Create the `full_adder_2bit` module as two component instantiations of the `full_adder`
 2. Elaborate your design and examine the generated schematic (under RTL Analysis). Keep expanding it to the lowest level. Take snap shots of the generated schematic(s), for the report.
 3. Follow the following instructions to create a testbench for functional simulation and verification of your design:
 - a. Define a constant (*PERIOD* = 10ns) for providing equal spacing between input values
 - b. Use in-line stimulus to cover all input combinations of A and B.
- Tip:** To include images of the simulations in your report, click on the setting icon  button in the top right corner of the simulator window and you will see the Waveform-Options window. If you click on the Colors tab, you will be able to change the colors of just about everything in the simulator window including the background.
4. Repeat step 3 using nested for loops to achieve the same task.
 5. Create and add a constraint file to assign the package pins according to the following (You can edit the Master XDC file provided with your board documents)
 - a. Input A is connected to sw1 and sw0, where sw0 is the LSb
 - b. Input B is connected to sw3 and sw2, where sw2 is the LSb
 - c. Output S is connected to LED1 and LED0, where LED0 is the LSb
 - d. Output carry is connected to LED3

Tip: Another way to do this is to *open the elaborated design under RTL Analysis in the Flow Navigator*. Then click on the I/O ports tab in the console window. you will see all the ports of your design. Map the ports you need. Make sure you select the appropriate I/O standard.

6. Synthesize and implement the design. View the implemented design (green blocks are DSPs. red blocks are Block RAMs, and blue blocks are Slices). Zoom in further until individual slice details are visible. Highlighted LUTs are the ones used and the green wires are the signal nets (keep a record of the slices and LUTs used and take snap shots). Go back under the *Project Summary* and look under the *Utilization Window Table*. Numbers should match what you had seen earlier.
7. Run *post-implementation timing simulation* (under *run simulation*), zoom into the signals and note any time delays using markers and take a snap shot. Comment on the delay and what might have caused it.
8. Generate the bitstream, program the FPGA board, test your design. Take a picture of the board showing its status when adding $2 + 3$ and include it with your submission.

Part 2: Two 2-bit comparator/multiplier using ROM

Design, simulate, and implement a design called *Comp_Mul_2bit* using two ROMs: one for comparison and the other for the multiplication; and a Multiplexor.
(ROM designs are explained in the introduction file for this activity)

The comparator compares two unsigned 2-bit numbers and asserts outputs indicating whether the decimal equivalent of word A is less than (lt), greater than (gt), or equal (eq) to that of word B.

Use the board resources as follows:

- PBO to select the operation to be performed
 - ❖ Comparison (PBO = 0) or Multiplication (PBO = 1).
- Input A is represented by sw1 and sw0, where sw0 is the LSb
- Input B is represented by sw3 and sw2, where sw2 is the LSb
- LED0 represented the (eq) status
- LED1 represents the (gt) status
- LED2 represents the (lt) status

The multiplier multiplies word A x word B as unsigned numbers and shows the results on LED3 – LED0, where LED0 is the LSb

1. Simulate the design using loops only to cover all possibilities as discussed in part 1. Include readable samples of the simulation waveforms in the submission.
2. Implement the design on the FPGA and test. Include pictures of the board status for each of the following cases

- a. Compare 3 to 2
- b. Compare 1 to 2
- c. Compare 3 to 3
- d. Multiply 1 x 2
- e. Multiply 3 x 2

Part 03 – Binary to BCD conversion

1. Design the C-module according to the truth table as given in the information file. You can implement the module using a ROM instead of developing the logic equations for each output. Since you can't define "x" as a logic level you can use one of the unused values such as "1 1 1 1"
2. Simulate the C-module and make sure it functions correctly according to the truth table.
3. Design a 4bit 2X4 mux to select each digit of the BCD output, to be able to display it on 4 LEDs, according to the following table

Select input		Output
sel 1	sel 0	
0	0	Units
0	1	tens
1	0	huds
1	1	zero

4. Design a top module with the declaration given below using structural modeling.

```

module Binary_to_BCD (bcd, bin);
    output [9:0] bcd
    input [7:0] bin;
    . . .
endmodule

```

5. Use a for loop to simulate the design using all possible combinations and check the waveform for corner cases to make sure the design is working correctly.
6. Implement the design onto the Zybo board according to the following assignment
 - a. Connect the Pmod switch module to the standard Pmod (JE). This will give you 8 switches: 4 on the board and 4 on the Pmod. Use these to represent the input bits (B7: B0) with the least significant bit connect to SW0 on the board.

- b. Make sure the pin connections of the Pmod are connected correctly to the input ports of your design. Use the board reference manual to identify the pin numbers.
- c. Connect the Mux output to the Leds [*LED3 : LED0*] on the board to show the binary representation of each BCD digit selected by the pushbuttons [*PB1 : PB0*]. These PBs are used as the Mux select inputs
- d. Take a picture or a short video to explain the check off of your working design.

Submission

In addition to the requirements listed in the “*Activities Report*” section on Canvas under the “*Course Resources*” module, prepare a separate zip file for each part of the activity and submit all three at the same time

TIP: DO NOT submit the whole project. **No grade will be given**

Name the submissions *LastName_Firstname_A01_P1*, *LastName_Firstname_A01_P2*, and *LastName_Firstname_A01_P3*