

ECE 1315 - Lab #9: Iterative Design

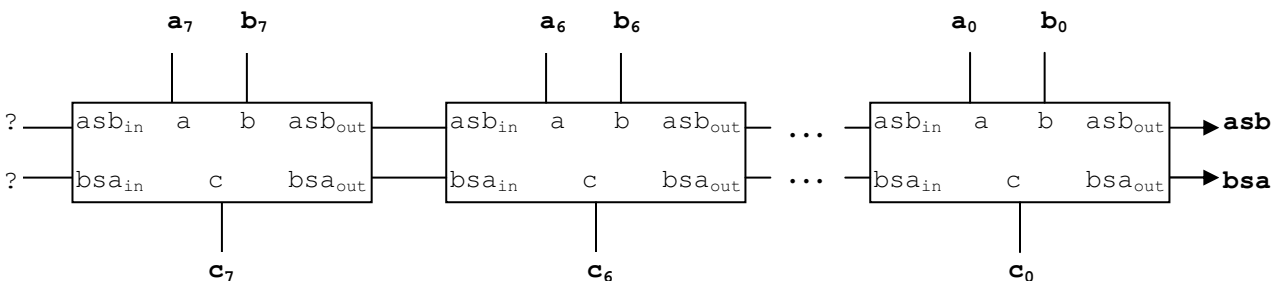
Design a combinational circuit that takes two 8-bit binary numbers a ($= a_7a_6a_5a_4a_3a_2a_1a_0$) and b ($= b_7b_6b_5b_4b_3b_2b_1b_0$) as input, compares them, and outputs the smaller of the two numbers as an 8-bit number c ($= c_7c_6c_5c_4c_3c_2c_1c_0$). In addition, produce two separate outputs, asb and bsa , that are set as follows:

- **$asb = 1$** : 'a' is smaller than 'b'
- **$bsa = 1$** : 'b' is smaller than 'a'
- **$asb = bsa = 0$** : 'a' = 'b'

Note that you will need two carry signals, since there are three cases that must propagate from one cell to the next, i.e. $a=b$, $a<b$, or $b<a$, which can be encoded in two carry signals.

Prelab: First, write a Verilog HDL file that describes the 8-bit comparator. You may write it on paper or in a general text editing file (e.g. Notepad) for speed of implementation during lab.

Next, design your circuit as an iterative network of cells so that the circuit could be expanded to work with any number of bits by simply replicating the iterated cell an appropriate number of times, but build and test the circuit with just 2-bit and 8-bit numbers using QuartusII.



Take advantage of the fact that the carry inputs to the most significant cell are known constants, and simplify the circuits in that cell based on that knowledge, similar to how the “full adder” simplifies to a “half adder” in the case of a ripple carry adder circuit. Be sure to identify the full iterating cell in your lab report, and show the simplified cell as a special case.

Minimize the total cost for the circuit you build that works with two 8-bit numbers, including seven complete cells (that would iterate for additional bits in the numbers) and one simplified cell for the most significant bits that results from the known constant values on the carry inputs according to Table 1. When you compile your design, look at the compilation report and **make sure to mark down how many Logic Elements (LEs) were used.**

Table 1: Available Logic Gates and Associated Cost

Available Logic	Associated Cost
NOT Gate	2
2-input NAND, NOR, XOR, XNOR	4
2-input AND, OR 3-input NAND, NOR	6
3-input AND, OR	8

Test your circuit as you did with combinational circuits in earlier labs, but this time using QuartusII. First, generate a 2-bit number comparator and test all possible cases for it. Then test *at least 5* different numbers using the full 8-bit number comparator. Make sure to take advantage of saving the generic cell as a symbol in order to replicate it for the 7 generic cells of the 8-bit number comparator. Your outputs should be the asb and bsa flags, and each of the 'c' output bits (8 bits for the 8-bit number comparator). **If you use the following naming conventions, you may use a provided vwf file for your testing of the 8-bit comparator:**

- **Inputs:** a[7], a[6], ..., a[0], b[7], b[6], ..., b[0]
- **Outputs:** asb, bsa, c[7], c[6], ..., c[0]

Also, compile your Verilog file by clicking on File → Create/Update → Create Symbol Files for Current File, entering the subsequent symbol into a bdf file and connecting input and output pins, and finally test it using the Vector Waveform File. In your lab report comment on how many Logic Elements were used by each implementation (Verilog vs. bdf).

Be sure to demonstrate your working circuit to your lab instructor and. Also, include your Verilog file and the in addition to snapshots of your block diagram and vector waveforms in you report.

Q#1: Which method (using Verilog or designing the iterative cells) used more Logic Elements (LEs)? Which method took longer?

Q#2: What is an advantage of iterative design? What is a disadvantage?