NC State University
Department of Electrical and Computer Engineering

ECE 310 – Fall 2025

Lab 7

Summary

In this lab, you will implement a single- and 4-digit BCD adder with Carry-in and Carry-out.

Deliverables

1. Pen-and-paper design of single-digit BCD adder
2. Code for model (testbenches are given)
3. Screenshots of simulation waveforms with at least 2 non-trivial 4-digit BCD numbers.
4. Screenshots of synthesis results and discussion comparing to expected results in Deliverable 1.
5. Extra Credit – up to 10 additional points will be added for typesetting your report in LaTeX, depending on usage of LaTeX elements (e.g. figures, tables, sections, etc).

Instructions

1. A testbench is provided to test your single-digit BCD adder (**combBCDadd_digit_tb.v**) and a second testbench is provided to test a 4-digit BCD adder to produce a 5-digit result (**combBCDadd_4d_tb.v**).

2. A structural implementation for a 4-digit BCD adder is provided. It connects 4 instances of a single-digit BCD adder together and manually manipulates the most significant digit to create a BCD digit (4 bits) from the single bit carry-out of the most significant BCD adder.

   Note: You do not need make any changes to the provided files as long as module names do not change and are used properly.

3. Building a BCD adder works very similar to the RCA's built at the gate-level in the past, where each operand is added along with a carry-in to produce a sum and a carry-out. Process starts with building a single-digit BCD adder, with more than one way to implement it. One way is to build a truth table with 9-bit inputs and 5-bit outputs, though this approach results in a very large table. Another approach is to work out a block diagram that connects functional components together to perform the operations necessary to compute a single-digit add.

   Process:
   a. Perform a binary addition operation of the 2 digits with carry-in
   b. If the result is > 9 (or produces a carry), then add 6
   c. The resulting carry is always 1 if the original/corrected sum operations produce a carry (and a corrected sum always produces a carry)
   d. The resulting sum is the original operation if there is no carry-out and the corrected sum if there is carry-out

The operation is purely combinational, so there aren't any steps that would take place over several cycles. This means that you should always be computing the original and corrected sums and using something to pick between the 2 options. An invalid codeword that is corrected will always produce a carry (e.g. decimal 10 (1010_1) + decimal 6 (0110_2) will produce a carry). A starting point you can use as a block diagram is given in Fig. 1 below. The add and correction operations are shown, you need to implement the logic left as a cloud in the diagram.

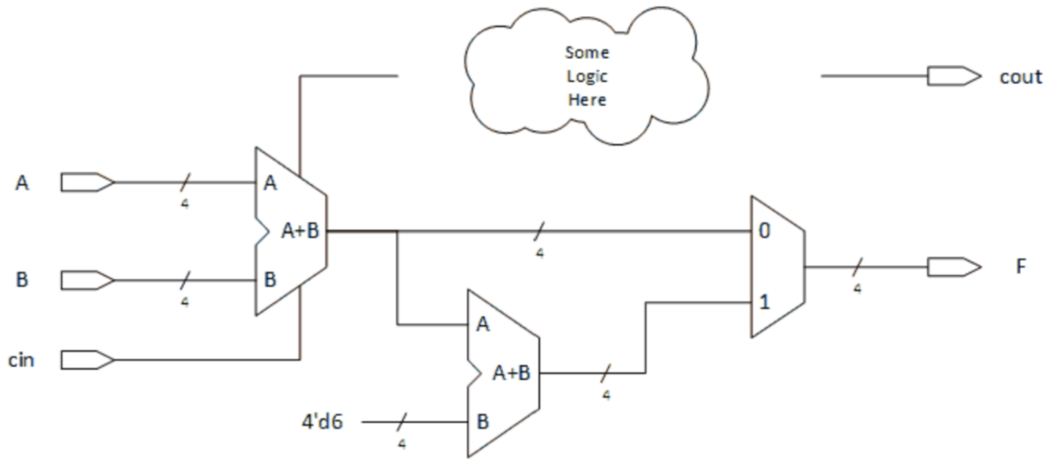

Fig. 1: Single-Digit BCD Adder.

4. You can choose any modeling level to complete the implementation of a single-digit BCD adder with carry-in and carry-out. The module **combBCDadd_digit.v** has been started for you. Please keep the module header intact as it positionally maps to the 4-digit adder.

<u>Grading breakdown:</u>

- Pen-and-paper/digital designs and comparison – 20 pts
- BCD adder model/implementation in Verilog – 40 pts
- Passing tests – 20 pts
- Lab report with as much information present as possible, including test vectors and reasoning, resulting waveforms and discussion, synthesis vs. paper design results and discussion – 20 pts
- Extra Credit – up to 10 additional points will be added for typesetting your report in LaTeX, depending on usage of LaTeX elements (e.g. figures, tables, sections, etc).