



**Alexandria University**  
**Faculty of Engineering**  
*Electrical Engineering Department*

**EE432 VLSI Modeling and Design**  
**Lab#4: System-Level Design of Combinational Circuits**

---

## Objectives

The purpose of this exercise is to design and examine combinational and arithmetic circuits that add, subtract, and multiply numbers. Each circuit will be described and verified in Verilog and implemented on an Altera FPGA board. Upon the completion of this Lab, you should be able to:

1. Design combinational circuits at different levels of abstraction.
2. Use Quartus II software to capture, synthesis, and implement digital FPGA systems and verify your design with Modelsim.
3. Download the design bitstream and test your implementation on the CIC-310 CPLD-FPGA kit.

## Requirements

Lab 4 will expose students to the design of some combinational circuits performing arithmetic operations such as adders, subtractors, and multipliers. A 4-bit adder/subtractor will be developed in Verilog using data flow modeling. A 4-bit multiplier will be developed in Verilog using structural modeling where the full adders can be instantiated.

You are required to write testbenches in Verilog to verify the functionality of the developed circuits. Use Modelsim to verify the functionality of the circuits and Quartus to configure the Altera FPGA. Test the implemented circuit using a set of on/off switches, pushbuttons, LEDs, and 7-segment displays offered by the CIC-310 CPLD-FPGA kit.

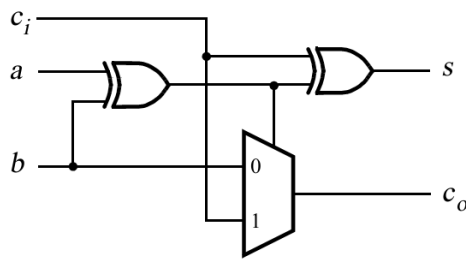
## Full Adder Description:

Figure 1-a shows a circuit for a-bit full adder, which has the inputs  $a$ ,  $b$ , and  $c_i$ , and produces the outputs  $s$  and  $c_o$ . Parts b and c of the figure show a circuit symbol and truth table for the full adder, which produces the two-bit binary sum  $s = a + b + c_i$  and  $c_o$ . Figure 1-d shows how four instances of this full

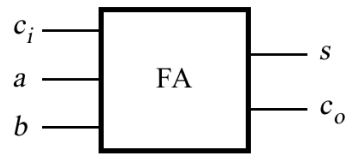
adder module can be used to design a circuit that adds two four-bit numbers. This type of circuit is usually called a ripple-carry adder, because of the way that the carry signals are passed from one full adder to the next.

Figure 2 shows a binary to seven-segment display decoder circuit. The circuit takes an input of 4 binary bits and displays the corresponding output as a hexadecimal value using different combinations of the seven LEDs. The table shows the decoder output for a 4-bit input (active-high LEDs).

A four-bit circuit that implements  $P = A \times B$  is illustrated in Figure 3. Because of its regular structure, this type of multiplier circuit is called an array multiplier. In each row of the multiplier AND gates are used to produce the summands, and full adder modules are used to generate the required sums.



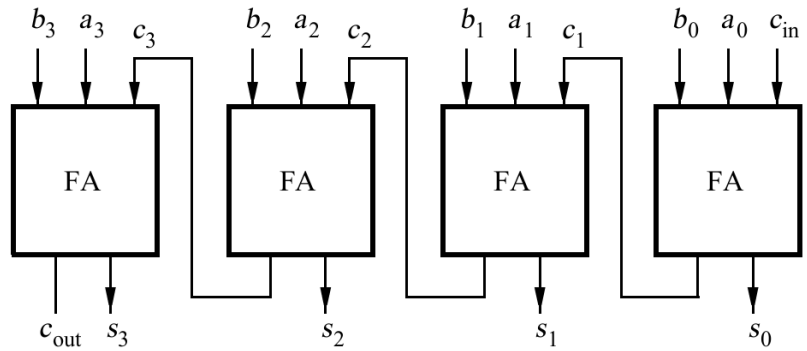
a) Full adder circuit



b) Full adder symbol

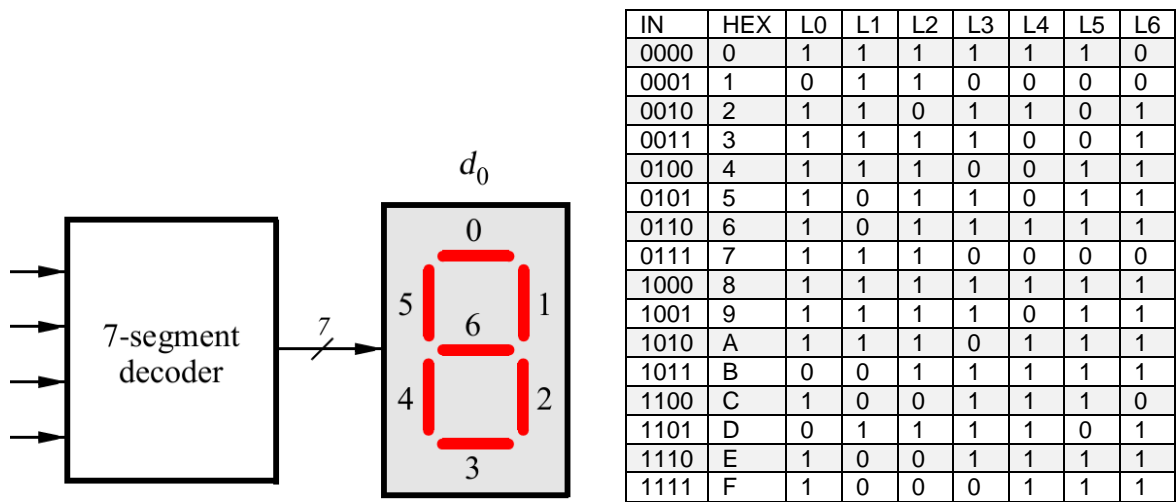
$b$	$a$	$c_i$	$c_o$	$s$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

c) Full adder truth table

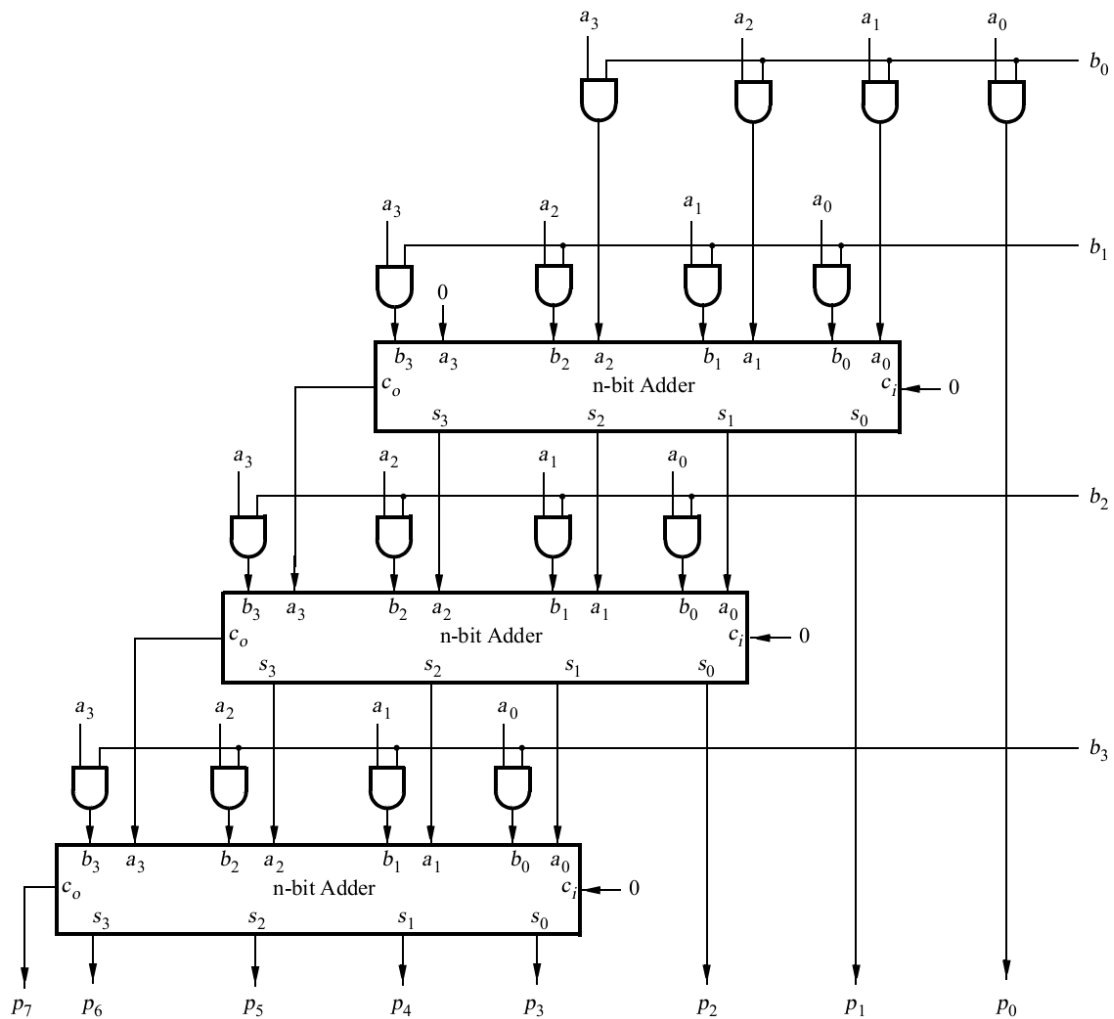


d) Four-bit ripple-carry adder circuit

**Figure 1: Ripple-carry adder circuit**



**Figure 2: Seven-segment decoder circuit**



**Figure 3: An array multiplier implemented using n-bit adders**

## Procedures:

1. Create a new Quartus II project for the adder circuit.
2. Write a Verilog module for the full adder subcircuit using dataflow modeling.
3. Write a top-level Verilog module that instantiates four instances of the 1-bit full adder to design a 4-bit full adder.
4. Use the Modelsim simulator to test your circuit by trying different values for numbers  $A$ ,  $B$ , and  $c_{in}$ .
5. Write a Verilog module for the seven-segment decoder circuit.
6. Write a top-level Verilog module that instantiates the 4-bit full adder and 3 instances of the seven-segment display decoder connected to  $A$ ,  $B$ , and  $sum$ .
7. Use 8 switches to represent the inputs  $A$  and  $B$ , respectively.
8. Use a separate switch for the carry-in  $c_{in}$  of the adder. Use a LED to represent the  $c_{out}$ .
9. Assign the pins on the FPGA to connect to the switches, LEDs, and 7-segment displays provided by the CIC-310 development board.
10. Compile, synthesize, and implement your Verilog code and download the generated bitstream into the FPGA chip.
11. Test the functionality of your circuit by toggling the switches and observing the 7-segment displays.
12. Repeat steps 3-9 to design and test the multiplier circuit.