Lab 7

Lab 7 assignment is due on Monday, April 3, by 9pm in Blackboard.

Please ask all questions about this assignment during the office hours or post questions on piazza.

Combinational logic design

In this lab, you will begin to design a circuit that you will use to build a vending machine controller. You will design some combinational logic which will generate outputs that tell the vending machine **whether to accept a coin that is inserted into the machine (thus, indicating that the circuit you are building does not handle situations in which a coin is not inserted)** or dispense a product to the customer. You will need to use abstraction to solve this part of the problem, since we have not taught you how to keep track of how much money has been inserted. You must build your circuit as if you had a circuit that kept track of how much money was previously inserted into the vending machine. This abstracted circuit will supply the signals S_2, S_1, and S_0 to the circuit you are building in this lab. These three signals encode information that you will use to generate your output signals.

The vending machine controller will accept only quarters (25 ¢) and dimes (10 ¢), will dispense a product when the customer inserts 35 ¢, and **will accept only exact change totaling 35 ¢** (you need to be extra careful about which coins you accept, so that you can achieve exactly 35¢). Furthermore, the vending machine will also keep track of how much money has been inserted into the vending machine and what coin was just inserted.

Use the specification below to construct truth tables and Karnaugh maps to derive simple SOP or POS Boolean expressions for the two output signals "Accept coin" (A) and "Paid in full (dispense Product)" (P).

- A = 1, when the coin that has just been inserted should be accepted. A = 0, when the coin that has just been inserted should be rejected.
- P = 1, when the product should be dispensed. P = 0, when the product should not be dispensed
- S_2 represents which coin was just inserted. S_1 and S_0 which coins were previously inserted prior to the coin that S_2 represents.

S_2	S_1	S_0	Function/meaning
0	0	0	No money was previously inserted. A dime has just been inserted.
0	0	1	10 ¢ was previously inserted. A dime has just been inserted.
0	1	0	25 ¢ was previously inserted. A dime has just been inserted.
0	1	1	Undefined/Unreachable
1	0	0	No money was previously inserted. A quarter has just been inserted.
1	0	1	10 ¢ was previously inserted. A quarter has just been inserted.
1	1	0	25 ¢ was previously inserted. A quarter has just been inserted.
1	1	1	Undefined/Unreachable

After you generate your Boolean expressions, *implement*, *test*, and *debug* your circuit in **Altera Quartus**. Refer to Lab 5 for details how to use the software for building and simulating circuits.

Note that although S_1=S_0=1 are unreachable, you have to simulate this case to make sure your Boolean expressions are implemented correctly.

Next, use DeMorgan's law and other Boolean algebra properties to redesign your circuit so that it uses only NAND, NOR, and NOT gates. We recommend that you use only NAND gates, but ultimately the decision about which gates to use is yours. Implement, test, and debug this circuit in Altera Quartus. You will use this circuit later in Lab 9, so we highly recommend you make sure this lab is done properly and you get it to work without problems.

What to turn in

Assemble and submit the following parts as a single pdf file:

- Truth tables **and** Karnaugh maps that define your values for the output signals A and P.
- Your simplified Boolean expressions for the output signals A and P.
- Printout of your Altera Quartus circuit schematics for **both** implementations of the circuit. (For example, if you choose SOP form, then your submission should include AND-OR and NAND-NAND circuits schematics and their waveforms; if you choose POS form, then your submission should include OR-AND and NOR-NOR circuits schematics and their waveforms.)
- Printout of your simulated waveforms for **both** implementations of the circuit.