IC220 Project #2 Cover Sheet

Your Last Name:                            Your First Name:                              Section:  2001  3001  4001

Describe any extra credit performed: __________________________________________

The maximum amount of credit this assignment is eligible for (check one):

☐ Full credit     (submitted promptly at start of class on the due date)
☐ 20% off        (submitted before 0800 on the following business day)
☐ No credit      (submitted after that. All assignments must be submitted to possibly earn passing grade)

Honor Pledge: My signature below certifies that

• For this project, I have not given any assistance, nor have I received assistance from anyone besides an IC220 instructor.
• I have not consulted any Internet sources or sources besides my course handouts, notes, or textbook.

Signed:_____________________________           Date:______________________

Feedback (reasonable answers required for full credit but will not otherwise affect your grade)

1. How long did this assignment take you to complete, outside of class?
   0.5 hr   1 hr   2 hrs   3 hrs
   4 hrs   5 hrs   6 hrs   Other:

2. Collaboration – not permitted on this project

3. What was the best part about this assignment?

4. The worst?

5. Do you think your assignment is… (circle one)
   Mostly correct and complete   Mostly correct, but not complete
   Mostly complete, but not all correct   Neither correct nor complete
   Other_______________________

6. If you couldn’t get this assignment mostly correct and complete, what was the primary reason? (circle one or two)
   Didn’t start soon enough   Too busy with other things
   Was harder than I expected   Couldn’t understand directions
   Couldn’t understand the concepts   Other ________________________

7. Was this assignment: (circle one)
   Way too easy   Little too easy   About right   Little too hard   Way too hard

8. In class/lecture, was the discussion of material relevant to this assignment:
   Way too slow   Little too slow   About right   Little too fast   Way too fast

9. Is there anything that I (the instructor) could have done better? Or other comments?
IC220 Spring 2007 Project #2: Donut Assembly

This project is to be completed independently (see details on coversheet pledge).
This means collaboration between students is not permitted. See the instructor for help or clarification.

Background: A mysterious benefactor has heard that you are now an expert in designing important logic circuits, and has requested your help on an important new assignment. The benefactor, apparently a donut manufacturer of some kind, needs help with automating his donut assembly line. He has told you that the donuts roll off the line in groups of five, but must be packaged in boxes of 8. He wants to begin using robots to pack the donuts into boxes, and as a first step requires you to implement the circuit described below. The use of this circuit is explored in more detail in the extra credit.

Task: Design a 3-bit state machine “counter” that operates as follows: Normally, the counter should begin at zero, count by five at each clock cycle, and then “rollover” when the result no longer fits in three bits, like this:
0..5..2..7..4..1..6..3..0..5..etc. This simulates 5 donuts going into each box, with any extras after the counter reaching 8 going in the next box. (That’s why ‘2’ comes after ‘5’ – after 5, we add 5 to get to 10, but 10 is more than 8. The result then is 2, which is how many donuts are left after taking 8 away from 10). However, the donut machine is not perfect – sometimes it produces four good donuts and one bad donut in a group. Fortunately, there is a sensor that detects when this has happened, removes the donut, and provides you with a true value on the signal ‘B’. So you must make your counter increase by only four during any cycle when B is true. Examples:

- Count 2, B=0 → Next count = 7
- Count 2, B=1 → Next count = 6 (only 4 donuts added)
- Count 4, B=0 → Next count = 1 (9 donuts rolled over to 1)
- Count 4, B=1 → Next count = 0 (8 donuts rolled over to 0)

You have a total of 2 external inputs: ‘B’, and a reset signal ‘R’ (see details below).

Details:

1) Start by looking at this example: (printout also shown on last page)
http://www.cs.usna.edu/~lmcdowel/courses/ic220/logicworks/ExampleCountingStateMachine.cct. This is a simple 2-bit machine that counts (by 1) and that has no external inputs (aside from the Reset switch). Two flip-flops hold the state, and some gates (2 AND, 1 OR, and 1 NOT) implement the combinational logic that computes the next state function. You are encouraged to base your solution on this general template!

2) You will keep track of the current counter with three current state bits (Q₂Q₁Q₀). So if logically the counter is 6 (110 in binary), then Q₂=1, Q₁=1, and Q₀=0. Implement each state bit with a LogicWorks D-type flip-flop.

3) Make a next state table showing the next state (Q₂’Q₁’Q₀’) in terms of the current state (Q₂Q₁Q₀) and the input B. Then minimize the logic needed as inputs to each of Q₂Q₁Q₀ using a K-Map. (You don’t need to include the reset switch in your state table – you can just hook up a single reset switch to the reset inputs of your flip-flops). This will define the logic that you should hook up to each of the “D” inputs of the flip-flops. You will have to turn this work in.

4) Use the flip flops labeled “D Flip Flop wo/SQ/” found in the “Simulation Logic” library. They offer input, output, clock and reset pins. (Note that the reset pin is implemented in negative logic – e.g., a reset occurs when this input is false).

5) Arrange the D-FFs as shown below. The top flip flop is the MSB (Q₂), and the bottom is the LSB (Q₀). By attaching the binary probes and viewing their output, you can ensure proper function of your counter.
6) You will need to hook up a clock signal to each of the flip-flops. At first, I suggest just using a binary switch and hooking up the output of that switch to each of the clock inputs on the flip-flop. You can then test your circuit by manually flipping the switch off and on. You should have just 1 clock signal for the entire circuit.

7) Once your circuit is working, delete the binary switch for the clock and replace it with an actual “clock component” from the “Simulation Logic” library. At the bottom of the toolbar there are a set of controls used for managing the clock that look like this:

![Clock Controls](image)

To see your circuit in action, click on the rightmost button on the picture above (between the slider control and the number display at far right). Then move the slider to the right to control the speed of the clock – make it slow at first to make sure things are working.

NOTE: the button on the far left is for “single-stepping”. This can be a little confusing because it does NOT advance time by one clock cycle – it advances time by some number of nanoseconds. This can be very confusing because LogicWorks models the propagation delay of signals through gates – so after a step, some gate outputs may not yet be stable. Thus, I don’t recommend using this button.

8) Once everything is wired up, you will need to reset the flip-flops to get them in a valid state. You should have a single binary switch than can reset all the flip-flops.

**Requirements:**

1) All inputs (here just ‘B’ and the reset switch) must be labeled and have switches attached.
2) All outputs (here just the 3 counter bits) must have probes attached.
4) The circuit must work.
5) Keep your design as simple and neat as possible: the fewest gates and the fewest crossing lines.
6) Your entire circuit must fit within one window and on one printed page.
7) Use only simple gates and the other components mentioned here – do not use adders, registers, etc.
8) Name your file as follows: Project2-section-lastname.cct. Example: Project2-1001-smith.cct

(continued on next page)
Tips

1) **START EARLY!** You have to work on your own and there is always the danger of running into some LogicWorks problem you don’t understand. So be sure to allow enough time to contact/visit the instructor if you have difficulty.

2) Look at the **ONLINE EXAMPLE** of a state machine: This shows a simple 2 bit machine that counts (by 1) and that has no external inputs (aside from the reset switch).

3) Be careful when doing the next state table and during minimization, then check your work before starting to build the actual circuit.

4) LogicWorks can be a little fussy about whether a connection is actually made. To see everything a wire is connected to, use the arrow pointer and click on the wire, then look carefully at the endpoints to see if they all look the same – a “T” at the end of the wire may indicate a bad connection. As another example, look at this screenshot:

![LogicWorks Circuit Diagram]

In this circuit (from an actual problem I had), the clock signal going into the top flip-flop is not connected properly – notice the “T”-like marking at the end (in some cases, this may only be visible once you click on the wire). If you can’t see this in the printed version of this handout, look at the color version online.

**Deliverables:** (1 through 5 handed in, stapled together in the following order)

1) Cover sheet with your feedback and pledge ___/10
2) State transition table showing states (present and next) ___/20
3) Reduction/K-map details, per flip-flop ___/20
4) Print-out of your functioning circuit ___/30
5) Logic Works file for your circuit (saved in your X drive) ___/20
6) Extra credit (up to 10 pts) ___

**TOTAL ___/100**

**Extra Credit:**

(5 pts) A. One use of this counter is to count how many donuts are in the current box, and then recognize when a new box needs to be started. For instance, if the counter is currently at “110” (decimal 6), then after the counter moves forward to “011” (decimal 3), a new box is needed which should initially be given just three donuts (what is left after finishing the previous box). Add new logic to detect when a box is about to be filled, then store this result in a new flip-flop called “NewBox” so that the signal “NewBox” is true during the cycle just after a counter rollover (for instance, when the counter changes to “001”). Note that you can’t just base this on looking at the current counter bits, because the NewBox signal **should not** be true just after a reset. These changes can be part of your original file.

(5 pts) B. New consumer pressure is also forcing the benefactor to start increasing quality even more. He has noticed that sometimes more than one donut out of the three may be defective. Modify your circuit so that it takes a total of two inputs \( B_0 \) and \( B_1 \), where \( B_0 = 1, B_0 = 0 \) means one bad donut in the bunch, \( B_0 = 1, B_1 = 1 \) means two bad donuts in the bunch. Modify the counting appropriately.
Example State Machine: