

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 three, 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.

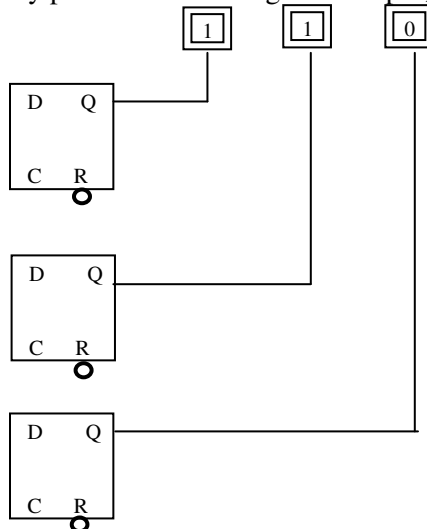
**Task:** Design a 3-bit state machine “counter” that operates as follows: Normally, the counter should begin at zero, count by three at each clock cycle, and then “rollover” when the result no longer fits in three bits, like this: 0..3..6..1..4..7..2..5..0..3..etc. This simulates 3 donuts going into each box, with any extras after the counter reaching 8 going in the next box. (That’s why ‘1’ comes after ‘6’ – after 6, we add 3 to get to 9, but 9 is more than 8. The result then is 1, which is how many donuts are left after taking 8 away from 9). However, the donut machine is not perfect – sometimes two donuts stick together. In this case, a human supervisor immediately eats the two “bad” donuts, and just one donut goes in the box. Fortunately, there is a sensor that detects when this has happened and provides you with a true value on the signal ‘S’. So you must make your counter increase by only one during any cycle when S is true. Examples:

- Count 2, S=0 → Next count = 5
- Count 2, S=1 → Next count = 3 (only 1 donut added)
- Count 7, S=0 → Next count = 2 (10 donuts rolled over to 2)
- Count 7, S=1 → Next count = 0 (8 donuts rolled over to 0)

Thus, you have one external input (S). You will also have a reset signal ‘R’ (see details below)

**Details:**

- 1) Start by looking at this example: (printout also shown on last page) <http://www.usna.edu/Users/cs/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 ( $Q_2Q_1Q_0$ ). So if logically the counter is 6 (110 in binary), then  $Q_2=1$ ,  $Q_1=1$ , and  $Q_0=0$ . Implement each state bit with a LogicWorks D-type flip-flop.
- 3) Make a next state table showing the next state ( $Q_2'Q_1'Q_0'$ ) in terms of the current state ( $Q_2Q_1Q_0$ ) and the input S. Then minimize the logic needed as inputs to each of  $Q_2Q_1Q_0$  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 ( $Q_2$ ), and the bottom is the LSB ( $Q_0$ ). 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:



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.

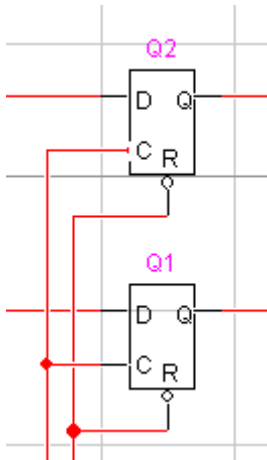
### Requirements:

- 1) All inputs (here just ‘S’ and the reset switch) must be labeled and have switches attached.
- 2) All outputs (here just the 3 counter bits) must have probes attached.
- 3) Place your name and date on the circuit – follow the instructions on the LogicWorks tutorial for printing: <http://www.usna.edu/Users/cs/lmcdowel/courses/ic220/logicworks/LogicWorksTutorial.htm>
- 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:  
Project2-8001-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:



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 5 pts)                                | ___            |
| <b>TOTAL</b>   | <b>___/100</b> |

**Extra Credit:**

**(5 pts)** New consumer pressure is forcing the benefactor to start increasing quality even more. He has noticed that sometimes all three donuts in a batch stick together. Modify your circuit so that it takes a total of two inputs S0 and S1, where S0=1, S1=0 means two stuck donuts (only one good donut to keep), while S0=1, S1=1 means three donuts stuck together (no good donuts to add to the count). Modify the counting appropriately.

