# Lab 9: K-Maps and Sequential Logic
# CS/CoE 0447: Fall 2015

Each person must turn in their own copy of the lab. If you choose to work with a neighbor/partner, put your partner's name on your submitted copy of the lab. Submission timestamps will be checked. Late submissions will not be accepted. **Follow instructions.**

## Part 1. One-bit Difference

In this section, you have to develop a circuit that can detect whether two two-bit binary numbers, A and B, differ by exactly one bit. To help you develop your circuit, use this worksheet.

Populate the Y column of the truth table. If binary numbers A and B differ by exactly one bit, then the output Y is 1. If they differ by two bits, then Y is 0. If binary numbers A and B are the same, then we do not care whether Y is a 0 or a 1. Thus, for those rows, enter an 'x'.

| Inputs | | | | |
|---|---|---|---|---|
| A | | B | | Output |
| $A_1$ | $A_0$ | $B_1$ | $B_0$ | Y |
| 0 | 0 | 0 | 0 | x |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | x |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | x |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | x |

| $A_1A_0$ \ $B_1B_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | 1 | 0 | 1 |
| 01 | 1 | x | 1 | 0 |
| 11 | 0 | 1 | x | 1 |
| 10 | 1 | 0 | 1 | |

G = ______________________________

You have to derive functions for Y in terms of $A_1$, $A_0$, $B_1$, $B_0$ and implement the corresponding circuit to get output Y from inputs $A_1$, $A_0$, $B_1$, $B_0$. Use the "Combinational Analysis" feature from Logisim to automatically draw the table, fill out the K-map, simplify, and implement the resulting functions. Consider verifying LogiSim's **combinational analysis** by comparing its work to your own. Submit your circuit as **lab09part1-<your Pitt Mail ID>.asm".** For example, "**lab09part1-xyz12.asm**". Clearly label your input and output pins. The circuit will be graded, but this worksheet will not be.

## Part 2. Grandma Ann's Cookie Watcher™

Grandma Ann, owner and founder of Grandma Ann's Crumbling Cookies[1], recognizes that to stay competitive in today's marketplace, she must integrate technology into her businesses in order to keep quality up and prices down.

To assist her business, she enrolled last year in CS 0447. Using her acquired knowledge, she designed her famous Cookie Watcher™. The cookie watcher "watches" multiple batches of cookies baking simultaneously. It is precisely timed to an individual recipe's cooking time. The cookie watcher alerts an employee to when a batch is ready to be taken out, as well as when a batch is ruined (overcooked). It is easily scalable to watch multiple batches at once.

Thanks to the Cookie Watcher™, fewer cookies are being burnt and the cookies are being consistently baked the proper amount of time. As a result, Grandma Ann's profits have soared and she has since retired to a private island in the Antilles.

In this section of the lab, you will recreate Grandma Ann's Cookie Watcher™ circuit in Logisim. Naturally, Grandma Ann first tested her design in Logisim and then eventually soldered her own physical version. You will recreate her Logisim version.

**First, look at the final design on the next page. Then, start the following steps:**

1. Start a new Logisim circuit up.
2. Click "Project" then "Add Circuit" to create a subcircuit. The benefit of using a subcircuit is that we can design it once, then instantiate multiple copies of it. Name this subcircuit "CookieWatcher". We will design this circuit first. It will be capable of watching exactly 1 batch of cookies.
3. Add 2 input pins (square) and give them the proper labels, "Clock" and "BatchHandled".
4. Add a counter (available from the "Memory" section in the left-hand pane). Change its "Action on Overflow" property to be "Stay at value." This way, when the counter reaches its maximum, it will not wraparound back to zero. Change its "Data Bits" to 10.
5. Connect the input pins to the counter as shown in the CookieWatcher design (Fig. 1). Do not let the counter's control pins float (although LogiSim mitigates it). Connect its Load pin to ground (⏚) or the constant 0. Connect its Count pin to power (upwards arrow) or the constant 1. The constants (constant, ground, power) are available in the "Wiring" folder.
6. Draw a short wire coming out of the "Q" output of the counter. This is the counter's value. The output is, by default, a "bundle". A bundle represents a group of several wires (in this case, 10 wires) all cabled together. Using a bundle can save us space when working with multiple bits at once. Bundles are always **black** in color, so we are unable to see the values on the individual wires within the bundle.
7. Connect a splitter (from the "Wiring" category) to the end of the bundle. Change the splitter's "Bit Width In" to 10 (because the counter's output is 10 bits). Change the splitter's

[1] They sell the finest cookies this side of the Mississippi River.

"Fan Out" to be 10. This will split the 10-bit input into 10 separate individual wires. A 10-bit input, split 10 ways, results in ten 1-bit outputs.

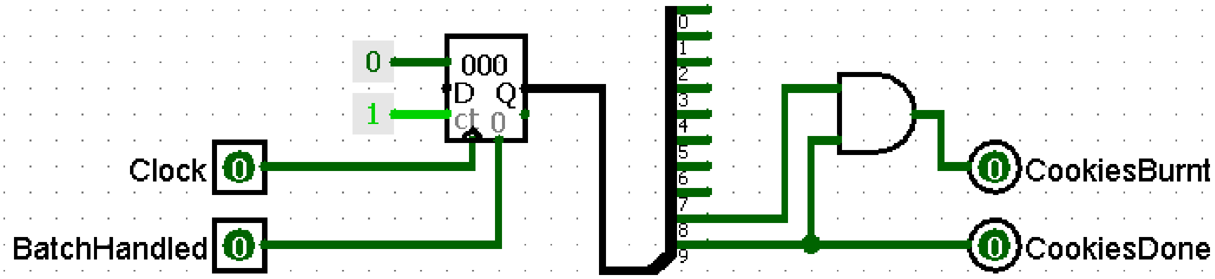

***Figure 1.*** *A schematic of the CookieWatcher subcircuit. One input is whether that batch of cookies was handled. The other input is the clock signal. The inputs are fed into a counter. The counter's value is used to determine whether the cookies are done and whether they are ruined (overbaked).*

8. Connect the most significant two outputs from the splitter to the AND gate as shown. Connect the output of the AND gate to an output pin. Use the output pin's "Label" property to label it "CookiesBurnt". When the CookiesBurnt pin is 1, the cookies are burnt, of course!
9. Connect the most significant output of the splitter, alone, to another output pin and label it "CookiesDone".
10. **Before moving on, think about the circuit you've just built.** Why are the outputs from counter connected up in the way that they are? What do you think will happen as the counter changes? When will the outputs of the cookie watcher become true? false?

**You have now completed the CookieWatcher subcircuit.** To test it, we will force the clock to tick, thus updating the circuit state. We can poke the clock pin to have it switch from low (0) to high (1) and vice versa. This lets us step through time, one cycle at a time.

We observe that the counter counts as expected and that poking the BatchHandled button resets the counter for a new batch.

But a single Cookie Watcher™ we have to keep poking isn't very useful or efficient! What we *really* want to do is to connect multiple CookieWatcher subcircuits to create a Cookie Baking Notification System (CBNS) capable of watching any number of baking batches automatically.

* * *

**Switch to the "main" circuit,** which should currently be blank. To do this, double click on the word "main" (which should be above the words "CookieWatcher" on the left-hand pane).

11. Add a clock component (available in "Wiring").
12. Add 6 buttons (available in "Input/Output").
13. Add 6 CookieWatcher subcircuits. To insert one subcircuit, **single-click** on the "CookieWatcher" entry in the left-hand pane. Then, click in the main circuit to place it somewhere. It will appear as a small box. Do this 6 times.
14. With the arrow tool active, hover over the four dots on a CookieWatcher circuit. Notice that

the left-hand ones are blue, meaning that they are unconnected, expecting an input. The right-hand ones are dull green right now, meaning that they are outputting a 0. We call these dots "pins". Hovering over a pin tells you the purpose of the pin, assuming you gave the subcircuit's pin a label. For example, hovering over the top pin on the right-hand side of a CookieWatcher should show that the pin is the CookiesBurnt pin.

15. Add 12 LEDs (available in "Input/Output").
16. Connect the components as shown in the figure.

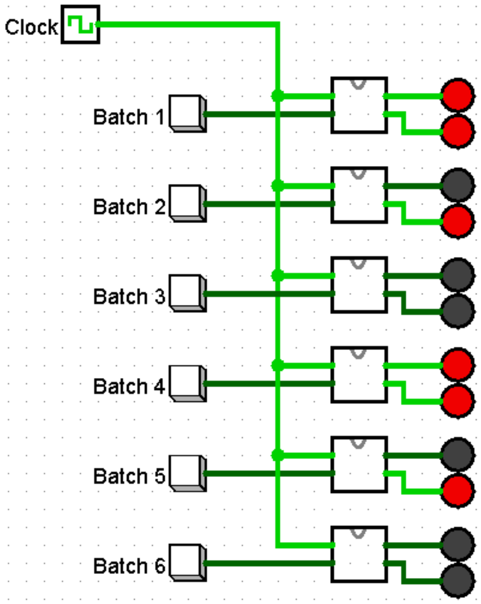

***Figure 2.*** *Six cookie watchers. Batches 2 and 5 are ready to be taken out of the oven. Batches 1 and 4 are burnt. Batches 3 and 6 are not yet ready to be taken from the oven.*

**Your circuit should now be done.** Like before, we can test it by poking the clock to force it to tick. To really test this circuit though, let's have the clock update automatically at some fixed frequency, say 128 times a second, or 128 Hertz. Under the "Simulate" menu, go to "Ticks Frequency" and select "128 Hz".

Now, under "Simulate", choose "Ticks Enabled" (keyboard shortcut **Ctrl-K**). The clock will automatically tick away, incrementing the counter by advancing the circuit state.

As a result, each CookieWatcher circuit will, eventually, turn on its corresponding lights. At 128 Hz, the bottom light should come on in about **8 seconds,** which informs the dutiful Grandma Ann's employee that a batch of cookies are done. The employee would then take out the cookies, put in a new batch, and press (i.e., poke) the batch's button to reset the timer.

If the employee, instead, is distracted and accidentally waits too long, the top light will come on. This lets the negligent employee (and their manager!) know that the cookies have been burnt. The employee should then throw out the batch, put in a new one, and press the reset button. At a clockrate of 128 Hz, the top light will come on after about **12 seconds.**

**Make sure that your circuit is working as designed.** You should understand what each of the components are doing and why they are doing what they do.

Ask yourself the following (you do not have to submit these answers):

- Under what conditions, exactly, does the CookiesDone light turn on?
- Under what conditions, exactly, does the CookiesBurnt light turn on?
- When the user pokes a button, what happens? How does the counter respond?

Save your Logisim circuit file as **lab09part2-<your Pitt Mail ID>.asm".** For example, "**lab09part2-xyz12.asm**" and submit via CourseWeb.