# CSS 422 Hardware and Computer Organization Sequential Circuits Lab: D Flip-Flops Instructor Rob Nash

Notes: Pick a group you haven’t yet worked with and attack this part of the lab. Have one person submit this to canvas or in-person when complete.

**Group names:**

Boyang Zhao

Thuan Tran

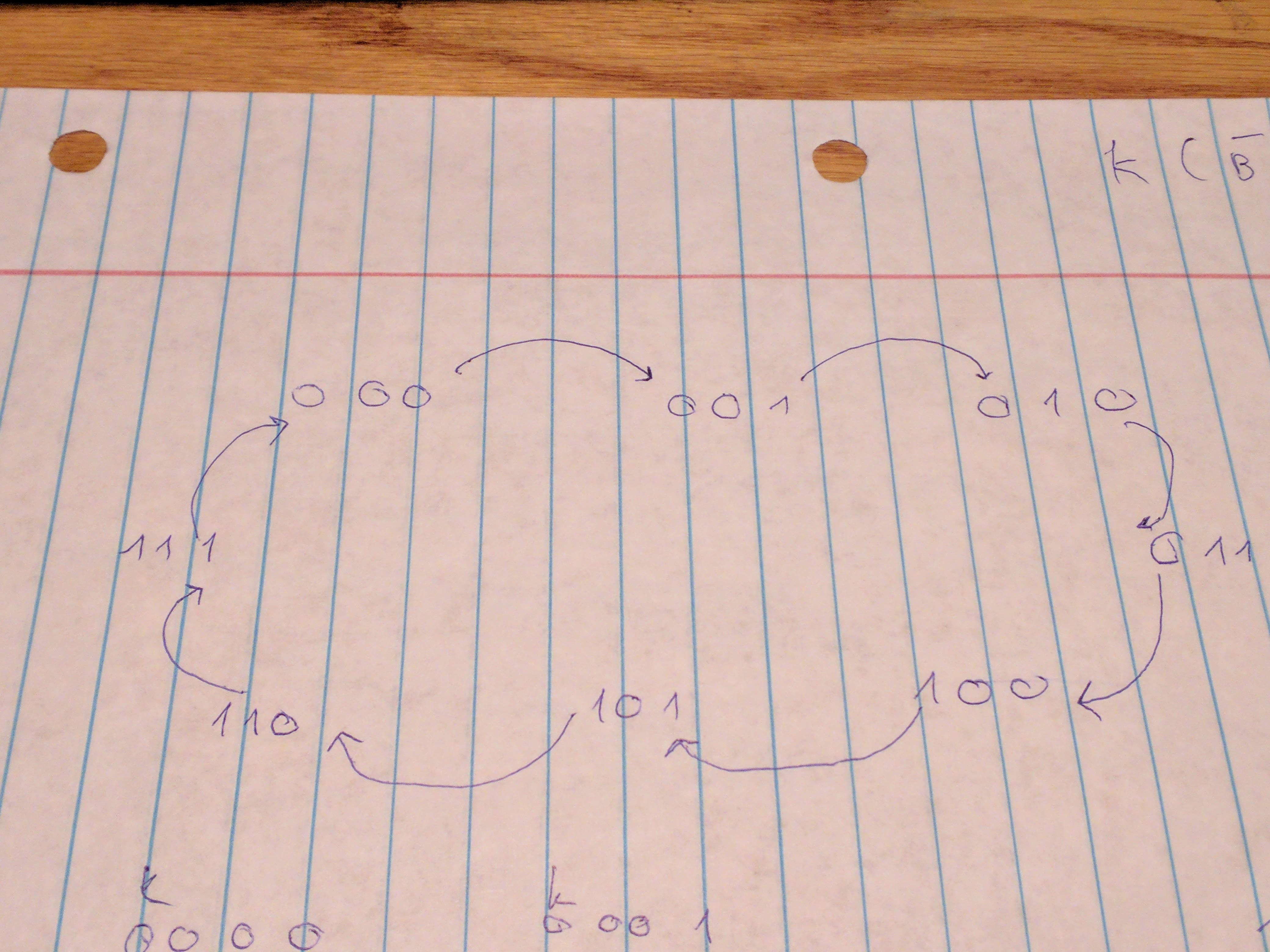
Aaron Vega

JD Mauthe

## Circuits That Count

Build a 3-bit count-up circuit, similar in spirit to the counters we built in class, but without the ripple (or frequency divide-by-two) behavior. Make a counter using D flip-flops and don’t build the ripple counter version of this sequential circuit. Notice how we’ll design a state diagram, a state (or excitation) table, or a circuit, and translate from any one of these specifications to the others, as they’re each a view for the same system.

(1) Make a state diagram for the 23 = 8 states of your counter. For simplicity, map the state with memory 000 to be at a count of 0, 001 a count of 1, and 111 a count of 7. Do note, however, that any state could be mapped to any count ahead of time, and so the state machine doesn’t *have* to be defined this way.



(2) Based on your state diagram (an FSM), complete the following state table for times t and t+1.

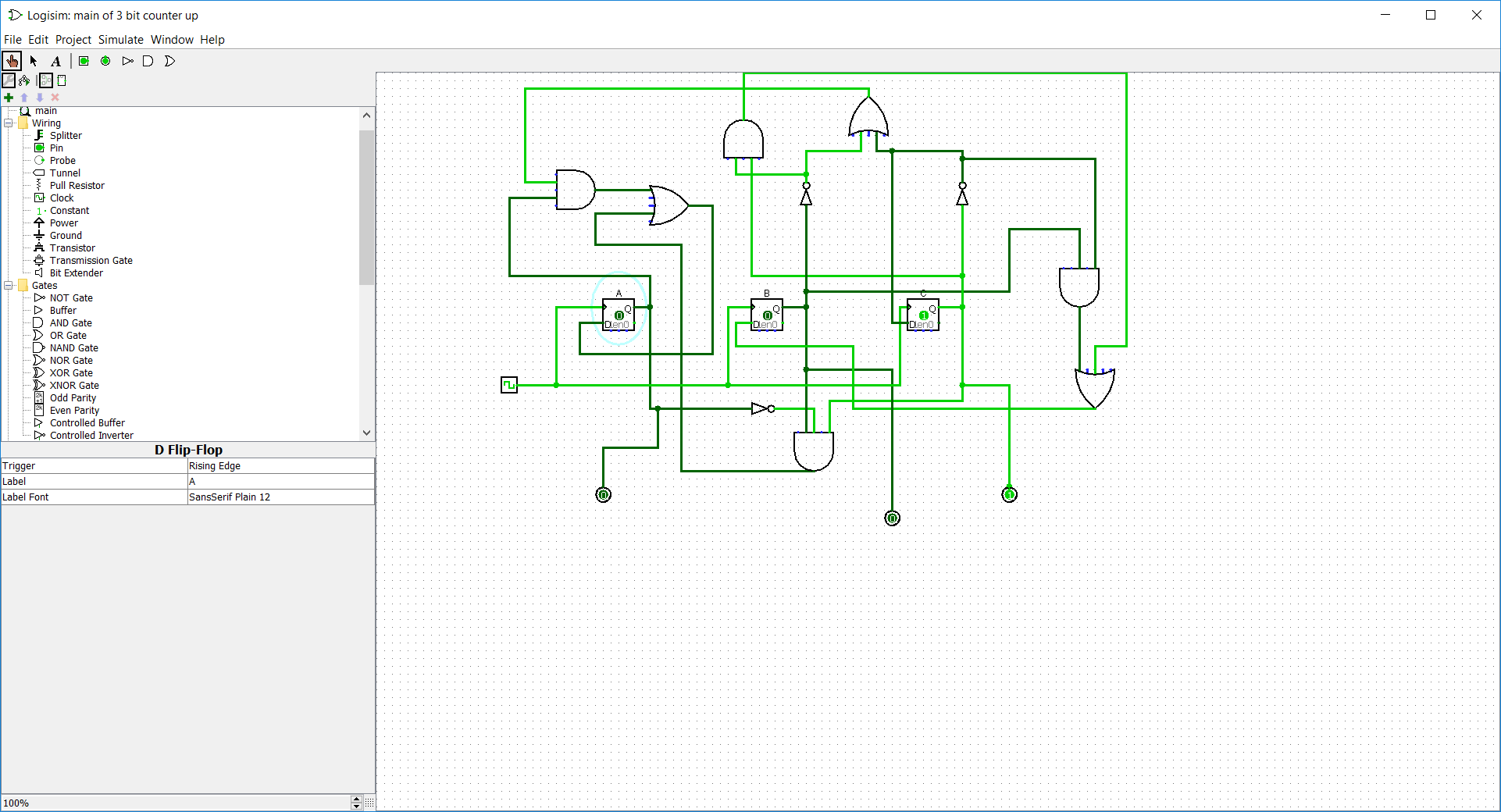
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Time t | | | Time t+1 | | |
| Q2 | Q1 | Q0 | Q2 | Q1 | Q0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

(3) Build an excitation table to signal the memories based on the desired states for t+1 and the provided input.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Time t | | | Time t+1 | | | D-FF in | D-FF in | D-FF in |
| Q2 | Q1 | Q0 | Q2 | Q1 | Q0 | D for Q2 | D for Q1 | D for Q0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

(4) Draw the circuit here. Do this by hand before proceeding to Logisim.

Here is my “drawn” circut

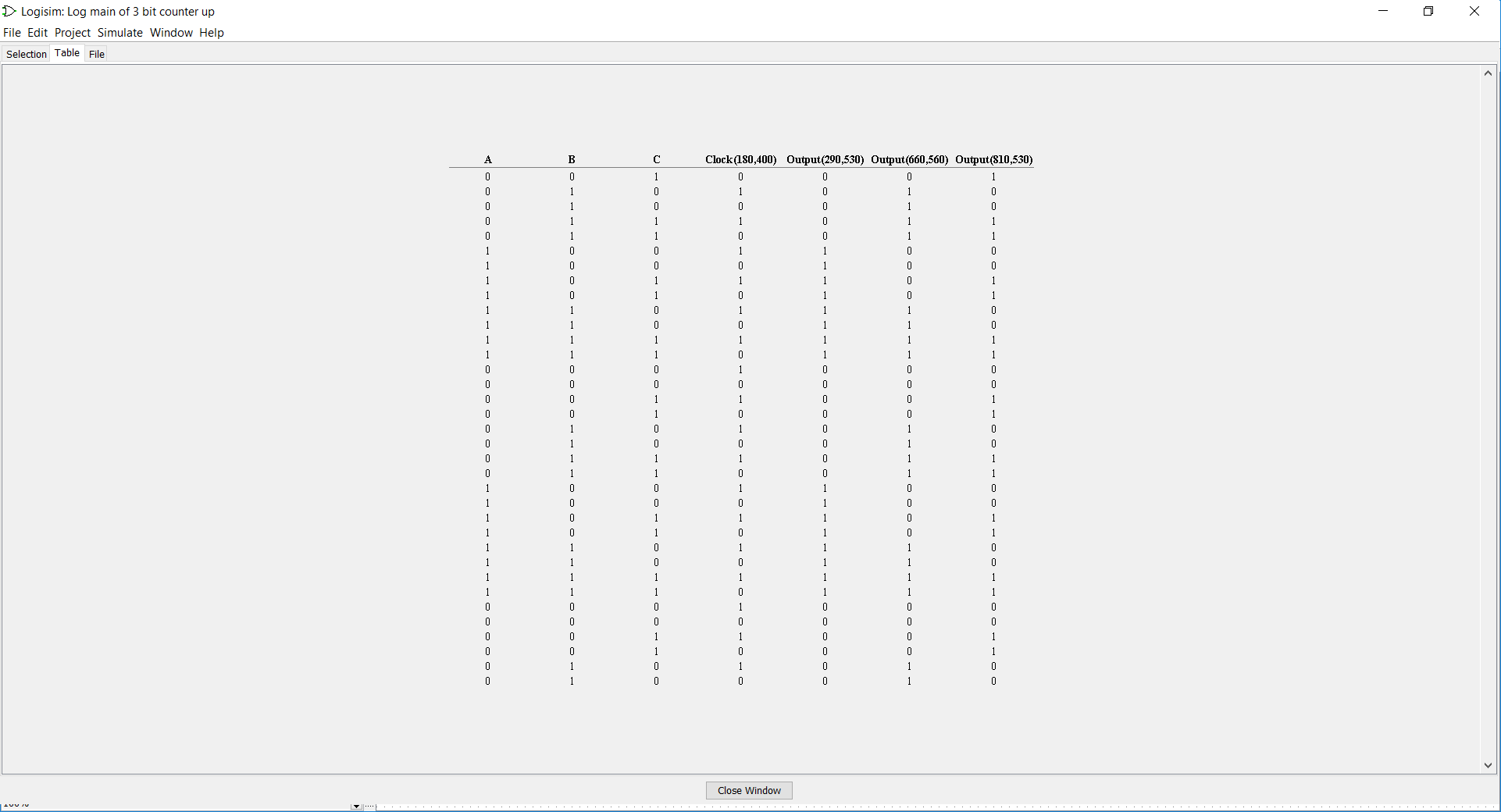


(5) Build it in Logisim and test it. Log your output and submit both the circuit file and the logged output.

D(A) = A(!B + !C) + !ABC

D(B) = B(!A+!C) + A!BC

D(C) = !C



(6) If we wanted to extend this counter to be a 4-bit counter, how would we do this?

1. What does this imply regarding the hardware, and what would we need to add?
2. Describe how you would test this circuit?

If we want to make this a 4 bit counter, we just need to add another D Flip flop. This implies that as the problems get more complex, we can solve it by adding more hardware/small components into it

I will test this circuit by logging the output and manually “poke” the circut

1. Build the 4-bit counter in Logisim and submit it.

Let the new bit to be the left most and name it K. Using K map, we have

D(K) = K(!B + !A) + B(K!C + !KAC)

A little bit messy but does the job

