Mod-16 Counter Using JK Flip Flops

Yilan Liu

Last updated: January 18, 2025

Initial Motivation

The initial motivation for making a mod-16 counter came from wanting to reduce the complexity of a project I was tackling at the time. This project made heavy use of the Multiplexer-16 gate which I had simply constructed as a collection of sixteen individual Multiplexer gates.

When it came time for me to step back and evaluate my work on the project, I realized that if I wanted to continue, I would need to construct the Multiplexer 16 gate in a much more scalable way so as to drastically reduce the overall complexity of the project. My idea for the revised construction was to only use one Multiplexer gate that would be used sixteen times over. My construction called for some sort of counter that would go from 0 to 15 with every clock cycle, and then return to 0 to start over again.

Thus began my project of making a mod-16 counter.

At first, I wasn't even sure if building such a counter using electrical components was possible. However, after much research, I stumbled across this youtube video that essentially explained how to construct a mod-8 counter using JK flip flops. After understanding the gist of their approach, I extended it to a mod-16 counter (basically going from three to four JK flip flops) and verified its correctness by implementing it on a breadboard.

Specifications

The Mod-16/4-bit counter takes as input a clock and returns as output a high-low value for four bus lines Q_0, Q_1, Q_2, and Q_3. The bus lines are meant to be collectively interpreted as an integer n, where

$$n = Q_0 2^0 + Q_1 2^1 + Q_2^2 + Q_3 2^3.$$

The integer n is initialized at 0 and is incremented with each clock cycle. After reaching the value 15, n returns to 0 with the next clock cycle, thus completing the mod-16 cycle.

Wiring the JK Flip Flops

We begin by considering some fixed present state ($Q_3Q_2Q_1Q_0$) and thinking about which state we want to follow it ($Q_3^+Q_2^+Q_1^+Q_0^+$).

$Q_3Q_2Q_1Q_0$	$Q_3^+Q_2^+Q_1^+Q_0^+$
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	1010
1010	1011
1011	1100
1100	1101
1101	1110
1110	1111
1111	0000

Then, we recall what the J and K inputs of a JK flip flop need to be for each possible bit-to-bit transition, as indicated by the table below. Here, "$\times$" stands for "don't care".

	J	K
$0 \to 0$	0	$\times$
$0 \to 1$	1	$\times$
$1 \to 0$	$\times$	1
$1 \to 1$	$\times$	0

Explanation of the above table:

1. The transition $0 \to 0$ only requires that $J = 0$. This is because the JK pair $(0, 0)$ would keep the state unchanged from 0 and the JK pair $(0, 1)$ would reset the state to 0 from 0.

2. The transition $0 \to 1$ only requires that $J = 1$. This is because the JK pair $(1, 0)$ would set the state from 0 to 1 and the JK pair $(1, 1)$ would toggle the state from 0 to 1.

3. The transition $1 \to 0$ only requires that $K = 1$. This is because the JK pair $(0, 1)$ would reset the state from 1 to 0 and the JK pair $(1, 1)$ would toggle the state from 1 to 0.

4. The transition $1 \to 1$ only requires that $K = 0$. This is because the JK pair $(0, 0)$ would keep the state unchanged from 1 and the JK pair $(1, 0)$ would set the state to 1 from 1.

Then, going back to our first table, we fixate on one row at a time, and within each row, we fixate on the bits Q_n and Q_n^+, where $0 \leq n \leq 3$. We use this to determine J_n and K_n as shown below. For example, take the first row in the first table. We have that $Q_0 = 0$ transitions to $Q_0^+ = 1$. We then look at our second table and notice that such a transition requires that J be 1 and K be anything, which is why we fill in "1 $\times$" under J_0K_0 in the first row. We fill in the remaining parts of the table in similar fashion.

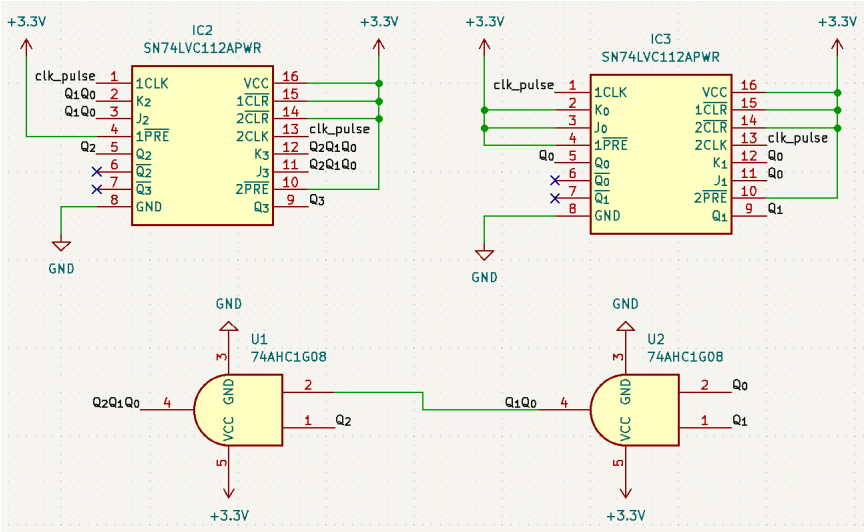
$Q_3Q_2Q_1Q_0$	$Q_3^+Q_2^+Q_1^+Q_0^+$	J_3K_3	J_2K_2	J_1K_1	J_0K_0
0000	0001	0 $\times$	0 $\times$	0 $\times$	1 $\times$
0001	0010	0 $\times$	0 $\times$	1 $\times$	$\times$ 1
0010	0011	0 $\times$	0 $\times$	$\times$ 0	1 $\times$
0011	0100	0 $\times$	1 $\times$	$\times$ 1	$\times$ 1
0100	0101	0 $\times$	$\times$ 0	0 $\times$	1 $\times$
0101	0110	0 $\times$	$\times$ 0	1 $\times$	$\times$ 1
0110	0111	0 $\times$	$\times$ 0	$\times$ 0	1 $\times$
0111	1000	1 $\times$	$\times$ 1	$\times$ 1	$\times$ 1
1000	1001	$\times$ 0	0 $\times$	0 $\times$	1 $\times$
1001	1010	$\times$ 0	0 $\times$	1 $\times$	$\times$ 1
1010	1011	$\times$ 0	0 $\times$	$\times$ 0	1 $\times$
1011	1100	$\times$ 0	1 $\times$	$\times$ 1	$\times$ 1
1100	1101	$\times$ 0	$\times$ 0	0 $\times$	1 $\times$
1101	1110	$\times$ 0	$\times$ 0	1 $\times$	$\times$ 1
1110	1111	$\times$ 0	$\times$ 0	$\times$ 0	1 $\times$
1111	0000	$\times$ 1	$\times$ 1	$\times$ 1	$\times$ 1

Finally, from this table, we determine the values that the J and K inputs to our four JK flip flops should be set to. There's a technique that the youtube video used for J_0, K_0, J_1, K_1, J_2, and K_2, but to figure out J_3 and K_3, I worked out what they should be by observing the pattern and extending it.

We obtain the values as follows:

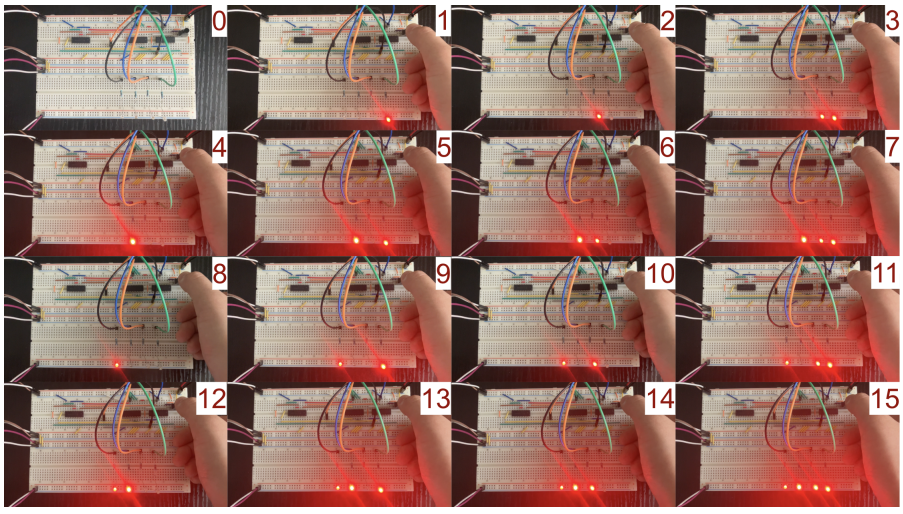
1. $J_0 = K_0 = 1$
2. $J_1 = K_1 = Q_0$
3. $J_2 = K_2 = Q_1Q_0$
4. $J_3 = K_3 = Q_2Q_1Q_0$

We wire the JK flip flops accordingly as shown in the below image. Here, IC2 and IC3 are dual JK flip flops and U1 and U2 are AND gates.

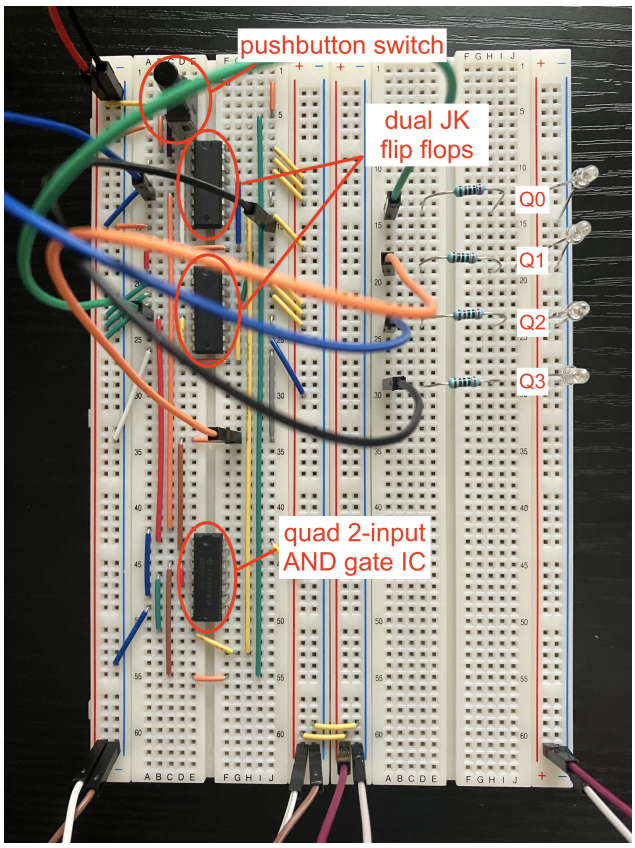


Implementation on Breadboard

A complete clock cycle is demonstrated in the below picture. On the breadboard, instead of using a crystal oscillator, I simulate a clock pulse by pressing down on a switch manually. To visualize the high/low values of Q_0, Q_1, Q_2, and Q_3, I connected LEDs to each line that would light up on a high value and be dark on a low value.



Here's a closer look at the breadboard:



Struggles and Lessons Learned

Compared to the amount of time I spent wiring the JK flip flops on paper and learning the theory, actually implementing the counter on a breadboard took much longer than I thought it would take. I encountered many unexpected hurdles, some of which I discuss below. However, ultimately I came away from the project with some rich experience, so it was all worth it.

1. **Always pair a SPST (Single-Pole Single-Throw) switch with a pull-up/pull-down resistor.** I've always heard hundreds of times in class that a pull-up/pull-down resistor would be needed in certain scenarios, but never internalized why. (In fact, you can see from another project linked on my website that I made the same exact mistake of not including a pull-up/pull-down resistor when I should've.) This time, after struggling with various issues that were traced back to a floating ground, I finally understand the concept.

2. **Perform sanity checks on the power line.** Towards the start of my

project, I kept having issues with unexpected outputs from my two JK flip flops. After much debugging, I discovered that the battery I was using as a power source wasn't delivering the desired output. Luckily, I had an Arduino on hand with 5V and 3.3V outputs built in, so I switched to using that as my power source.

3. **Beware of pushbutton switches without brackets.** I experienced a lot of issues with the first switch I used which didn't have brackets. Firm presses on the switch would sometimes not be registered, and other times, they would be registered as two quick presses. I finally got fed up after a couple of days and took to searching for a new switch. As I was reading the datasheet for a potential suitor, I saw that the manufacturer had made a variation of the switch with brackets. After looking into it, I discovered that switches with brackets are quite common and are designed for users who need to mount the component onto a breadboard.

Future Directions

As I mentioned in the first section, this project was born out of a desire to reduce the complexity of my design of the Multiplexer 16 chip that I was using in another project. Now that I have finished building the counter and verified its correctness, I feel confident proceeding with my redesign of the Multiplexer 16 chip.