Victor Yuan

October 27, 2016

Lab 9

Introduction:

This lab examined a J-K Flip-Flop circuit as it was used to make a counter to count the following sequence 0000, 0001, 1000, 1001, 0100, 0011, 1010, 1011, 0110, 0111, 1110, 1111, repeat.

In this lab

Team Member Responsibilities:

Lab Partner: Lin Boce

I was in charge of J and K map of A and B and part of the Verilog code. My partner was in charge of the J and K map of C and D and part of the Verilog code.

Materials:

Sample Verilog code, Excel, Putty, UNIX, Verilog simulation and Lab Computer.

Procedure:

The goal of the lab was to produce a J-K flip-flop that would display the sequence 0000, 0001, 1000, 1001, 0100, 0011, 1010, 1011, 0110, 0111, 1110, and 1111. First, a table of the present state values was made. This table had four columns of variables A, B, C and D, all 16 combination of 1 and 0 were listed this showed.

Next to the present state table a next-state table was created and the next-state values was written in, for example 0000 in the present state table would become 0001 in the next-state table since that is what the sequence dictates.

Then 8 Karnaugh Maps were created 2 for each Variable one J map and one K map. Find the sum of product realization of each of these Karnaugh maps then implement the J-K Flip Flop on to paper diagram and then simulate it through Verilog. Look at the provided file “74LS107A.v” as a example. Then compile and run the program with “Verilog 74LS107a.v (your file name here)” in the command prompt.

Questions:

I. Design:

1. Derive the next-state table for the counter.



2. Write out the Karnaugh maps for the J and K inputs.

A) ab

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | x | x |
| 01 | 1 | x | x | x |
| 11 | 1 | 1 | x | x |
| 10 | x | 0 | x | x |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | 00 | 01 | 11 | 10 |
| 00 | x | x | x | 0 |
| 01 | x | x | x | 1 |
| 11 | x | x | 1 | 1 |
| 10 | x | x | 0 | 0 |

cd  
B)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J | 00 | 01 | 11 | 10 |
| 00 | 0 | x | x | 0 |
| 01 | 0 | x | x | 1 |
| 11 | 0 | x | x | 1 |
| 10 | x | x | x | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | 00 | 01 | 11 | 10 |
| 00 | x | 1 | x | x |
| 01 | x | x | x | x |
| 11 | x | 0 | 1 | x |
| 10 | x | 0 | 0 | x |

C)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | x | 0 |
| 01 | 0 | x | x | 0 |
| 11 | x | x | x | x |
| 10 | x | x | x | x |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | 00 | 01 | 11 | 10 |
| 00 | x | x | x | x |
| 01 | x | x | x | x |
| 11 | 0 | 0 | 1 | 0 |
| 10 | x | 0 | 0 | 0 |

D)

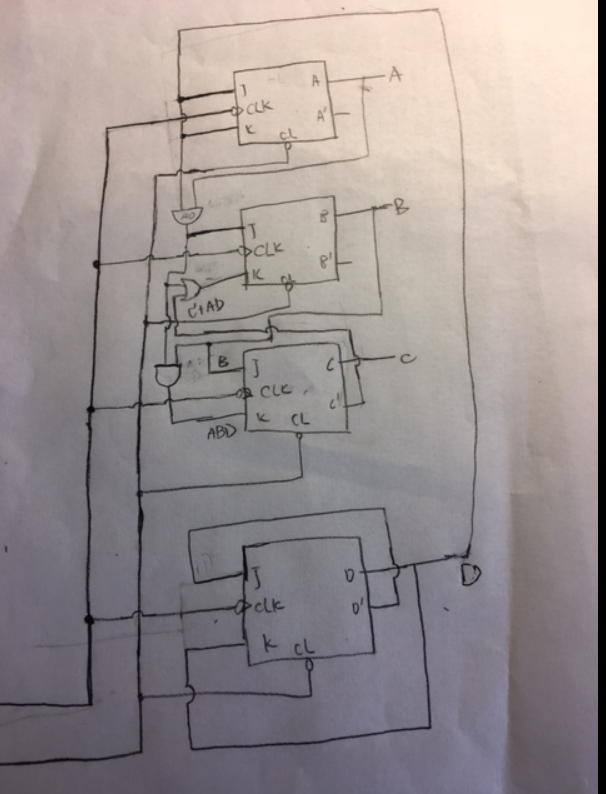
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | x | 1 |
| 01 | x | x | x | x |
| 11 | x | x | x | x |
| 10 | x | 1 | 1 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | 00 | 01 | 11 | 10 |
| 00 | x | x | x | x |
| 01 | 1 | x | x | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | x | x | x | x |

3. Realize the J and K inputs using a small number of AND and OR gates, each with two inputs. Taking advantage of common gates can help reduce the overall gate count. The smaller the number of total gates, the better the design.

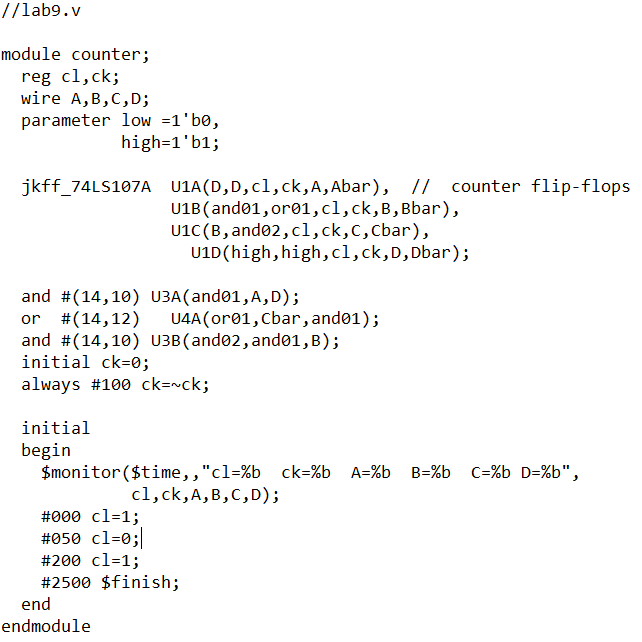
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| J | D | AD | B | D’ |
| K | D | C’ + AD | (AD)B | D |

4. Draw a schematic of your counter that shows the pin-out of all gates.

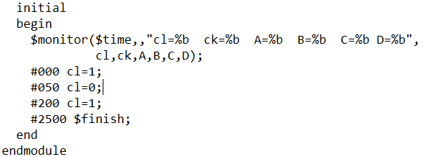


II. Simulations:

1. Develop a Verilog module for your counter design that uses the negative-edge triggered J-K flip-flop module 74ALS107A.v that is available on the class Website. Please do not make any changes to 74ALS107A.v. Be sure to use the rise time and fall times delays for the 74ALS00 family of TTL logic from Texas Instruments given in Table 2 for all other gates.



1. Develop a stimulus pattern that first resets you counter and then clock it through one complete count cycle, stopping in the reset state. Make sure that each clock cycle is high and low for the same amount of time.



1. Indicate how you determined the number of simulations steps for each clock cycle. What happens if the clock cycle is too short?

I noticed it took a time interval of 200 for the counter to switch from 0001 to 1000 and there were still 11 numbers to count so 11 \* 200 = 2200 added to 300 the starting time.

If the clock cycle is too short the counter still works however it will not show a complete sequence.

III. Conclusions:

1. How would using components from a mixture of different 7400 TTL logic families affect the operation of your circuit?

As long as the logic gates come from the 7400 TTL logic family all I would need to do is change the rise and fall time delays in my Verilog code to account of the change in delay times.

The initial goal of creating a counter counting the sequence 0000, 0001, 1000, 1001, 0100, 0011, 1010, 1011, 0110, 0111, 1110, 1111 using a J-K Flip-Flop was met.

