CDA 3201L - Thursday (3:30 - 4:45PM) Section 005

Lab #07 - Sequential Logic Circuits (3)

11/12/2016 - Steven Romeiro, Jennyfer Munoz

Purpose & Objectives:

Design a nine-step counter to count in the following sequence using J□ flip-flop.
Iterating sequence should be as follows:
0011, 0101, 1001, 1000, 1011, 1010, 0110, 0100, 0111, 0011, 0101, ….
The design will include a means for resetting the counter to 0011. A 7-segment LED will be used to display the output. This will require the use of a BCD-to-7 segment decoder (74LS247) to convert the 4-bit binary values to be displayed in the 7-segment LED.

Components Used:

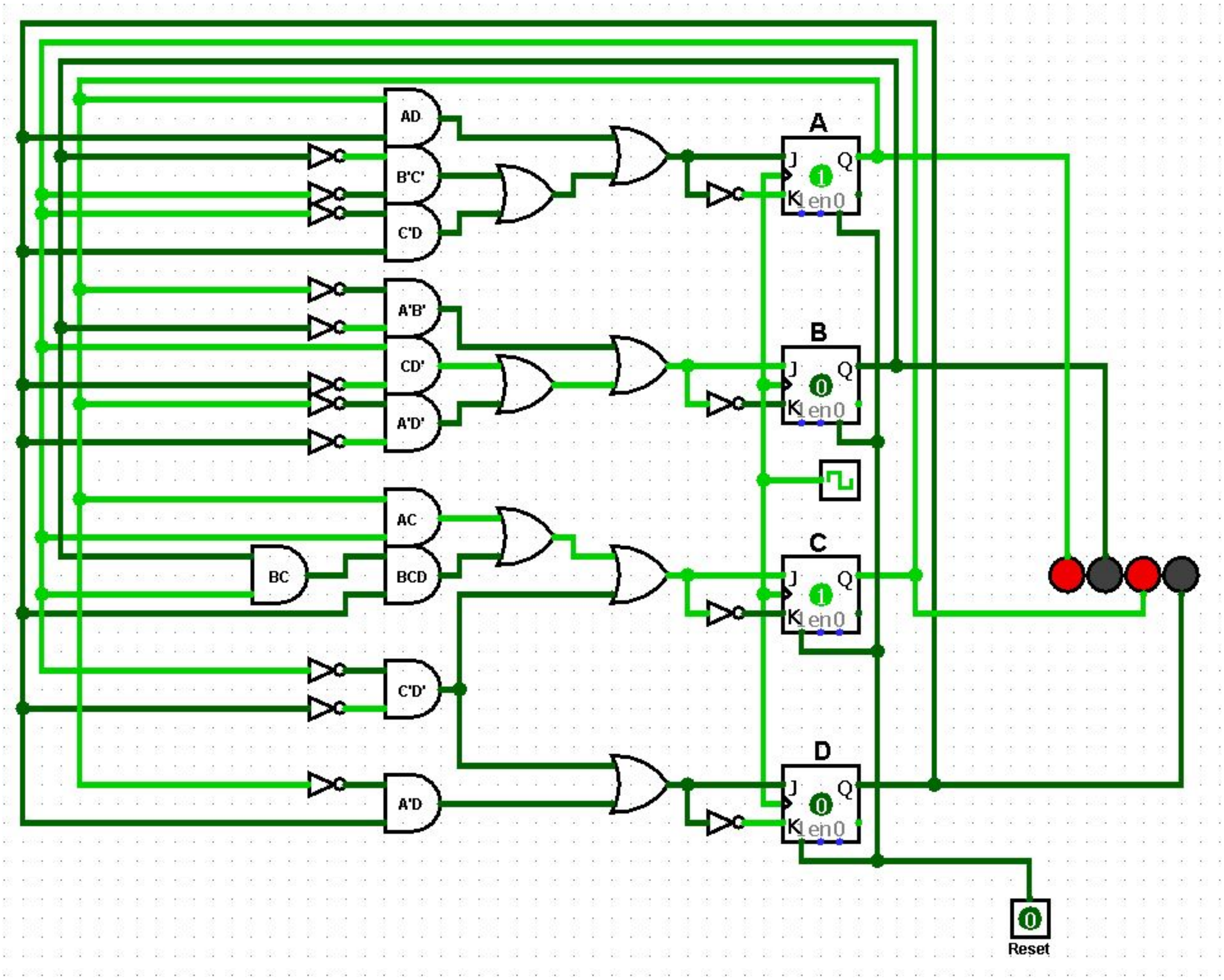
Name	**Type**	**Quantity**
74LS08	AND IC (2-inputs)	3
74LS04	Inverter IC	1
74LS32	OR IC	2
74LS109	JK Flip-Flop	2
BCD to 7-Segment Decoder	74LS247	1
Display	7-Segment Display	1
470 Ω Resistor	Resistor	4
LED	Red LED	4
Power Supply	5v	1
Frequency Generator	Wavetek 4MHz	1
Wire Kit	Assorted	1

Description

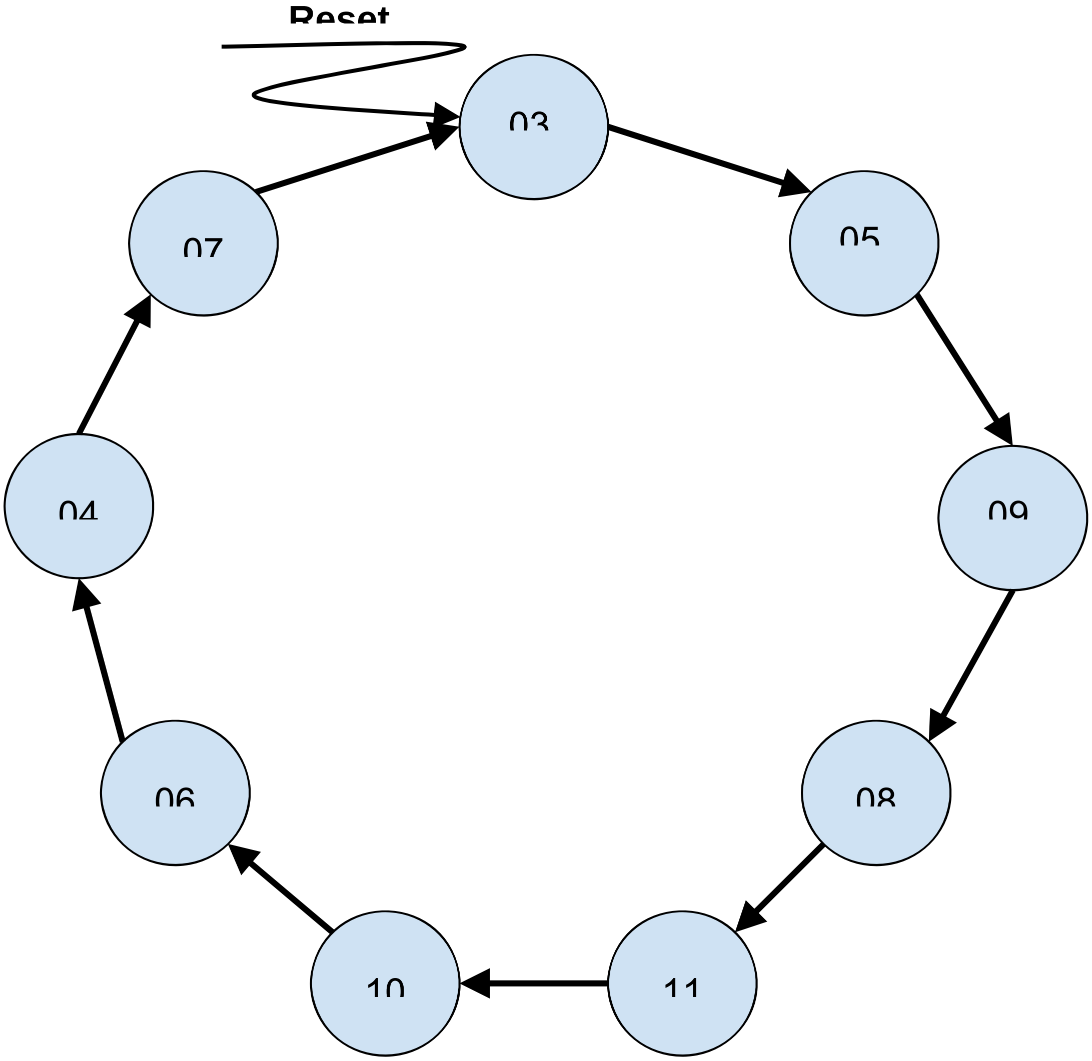
The design we chose implemented JK Flip-Flops as D Flip-Flops by making a small change to the inputs. This counter should iterate through a fixed sequence of numbers without external inputs to modify the results. The system should be completely self-contained, moving from one state to the next. The sequence iterates through the following numbers: 3, 5, 9, 8, 11, 10, 6, 4, 7.

Once the last number in the sequence is reached it loops back to the first number in the sequence (3) and proceeds again with the iteration. The 7-Segment Display will be used to easily identify the current number in the iteration. A method for resetting the sequence back to its initial value (3) will be implemented. The reset will be independent of the clock and will be using the Clear and Preset pins available in the JK Flip-Flops.

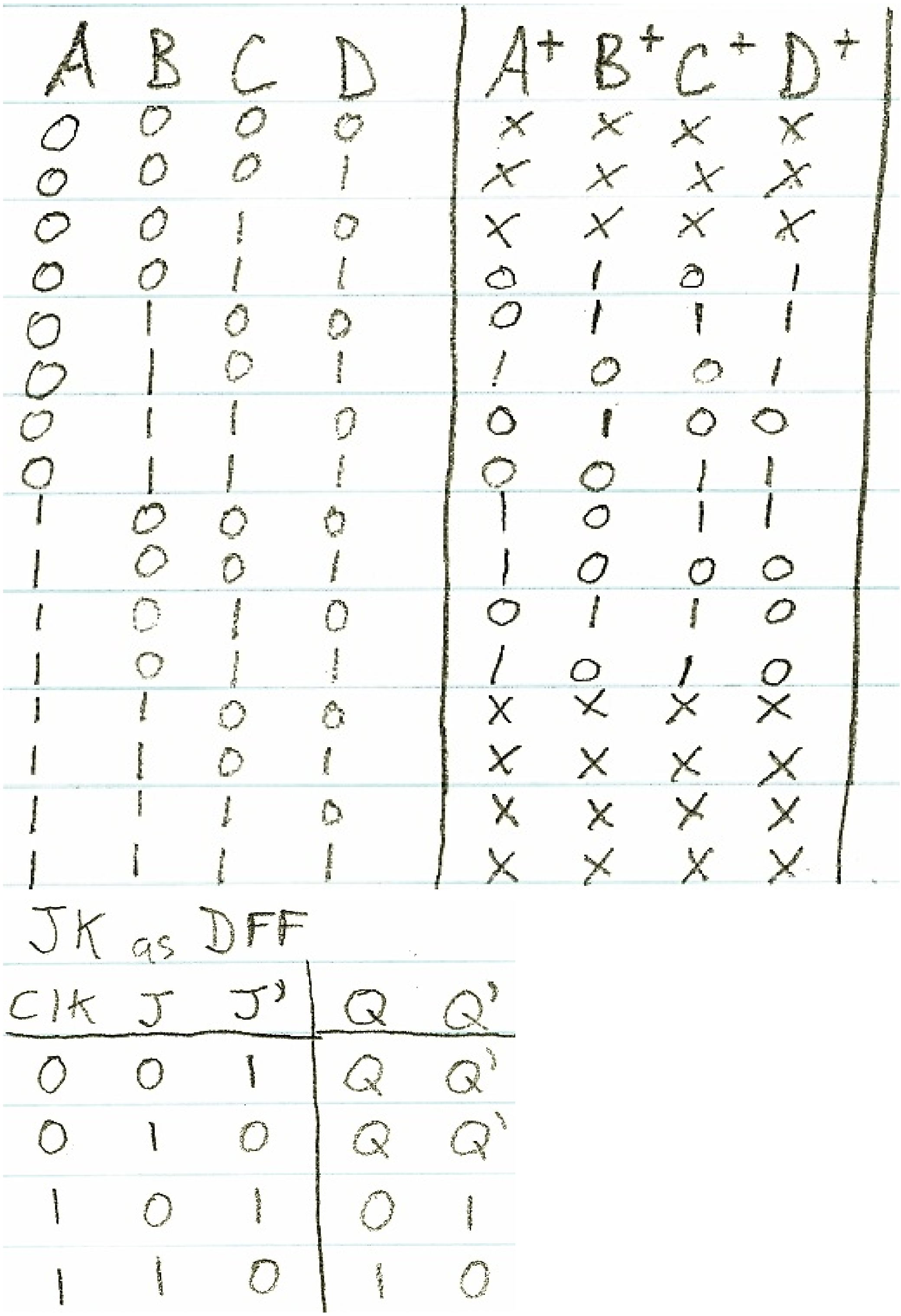
Diagram



State Diagram:



Truth Table

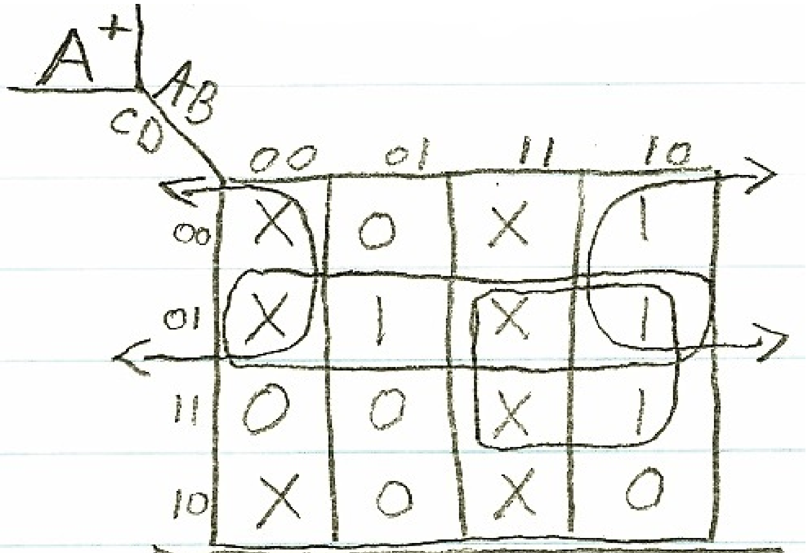


A	B	C	D	A^+	B^+	C^+	D^+
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	0	1	0	1
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	1
0	1	1	0	0	1	0	0
0	1	1	1	0	0	1	1
1	0	0	0	1	0	1	1
1	0	0	1	1	0	0	0
1	0	1	0	0	1	1	0
1	0	1	1	1	0	1	0
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

JK as DFF

Clk	J	J'	Q	Q'
0	0	1	Q	Q'
0	1	0	Q	Q'
1	0	1	0	1
1	1	0	1	0

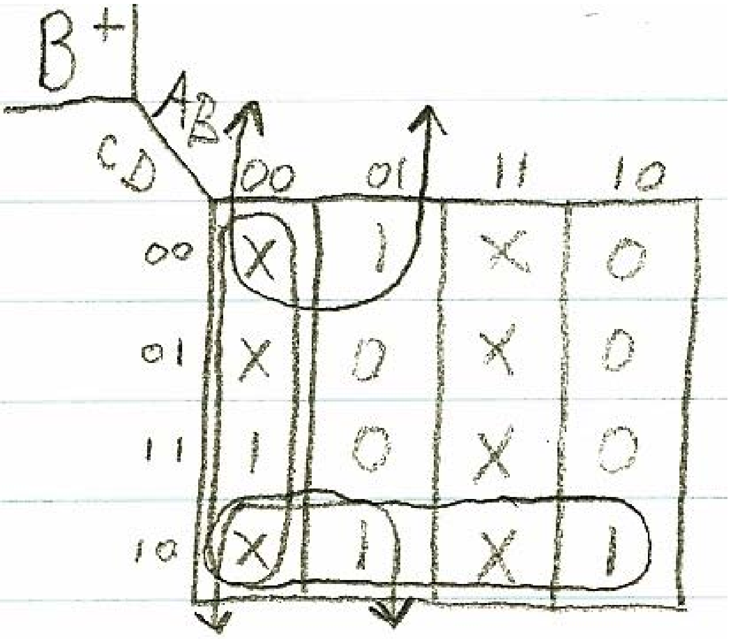
KMAPS & Expressions:



A^+

CD \ AB	00	01	11	10
00	X	0	X	1
01	X	1	X	1
11	0	0	X	1
10	X	0	X	0

$$\boxed{A^+ = \overline{B}\,\overline{C} + \overline{C}D + AD}$$



B^+

CD \ AB	00	01	11	10
00	X	1	X	0
01	X	0	X	0
11	1	0	X	0
10	X	1	X	1

$$\boxed{B^+ = \overline{A}\,\overline{B} + C\overline{D} + \overline{A}\,\overline{D}}$$

C^+

CD \ AB	00	01	11	10
00	X	1	X	1
01	X	0	X	0
11	0	1	X	1
10	X	0	X	1

$$\boxed{C^+ = \bar{C}\bar{D} + AC + BCD}$$

D^+

CD \ AB	00	01	11	10
00	X	1	X	1
01	X	1	X	0
11	1	1	X	0
10	X	0	X	0

$$\boxed{D^+ = \bar{C}\bar{D} + \bar{A}D}$$

Discussion & Conclusion:

We chose a design approach that differed from most of the other students by implementing this counter using JK Flip-Flops (JKFF) that were modified to function as D Flip-Flops (DFF). We chose this approach as it seemed more natural to create a counter with DFF logic rather than relying on JKFF functionality. The task proved easier to develop but harder to implement. The design approach allowed us to very simply develop a small state table that easily related the current output with the next desired state input. This approach only required 4 simplified expressions in our design phase. The implementation phase however, proved more difficult than expected. The simplicity of the design phase led us to believe that the implementation phase would follow suit. Implementation of our design led to the use of more gates than expected. The 9-step counter we’ve designed are state machines. With any state machine, the current state it is in can be a point of entry for new functionality. A vending machine for example, at different states can either be waiting for payment, waiting for selection or performing a fetch operation. The states of our counter could easily be implemented as states of a machine where each state is a new state with brand new functionality.