CDA 3201L Thursday Section
Lab number 07 -- Sequential Logic Circuits III
07/05/14, John Gangemi

PURPOSE AND OBJECTIVES

Lab assignment number 7 requires that a nine-step counter be designed to count in a predefined sequence using D flip-flops, specifically the TTL 74LS74 IC. The design guidelines also call for a means of resetting the counter to the first state of the sequence (0011). It is also required that the counter be implemented in Verilog.

This assignment is meant to introduce a fundamental sequential logic circuit (counter) using the basic design procedure for finite state machines (state transition diagram, state table, and next-state logic).

A finite state machine is a sequential logic circuit that exists in one of a fixed number of states determined by design requirements.

COMPONENTS USED

- Integrated Circuits For Demonstration
    - 74LS74 Dual D Flip-Flop with Preset and Clear
    - 74LS02 Quad NOR gate
    - 74LS32 Quad OR gate
    - 74LS08 Quad AND gate
    - 74LS04 HEX Inverter
    - TLC555 LinCMOS TIMER (astable operation @ 1Hz)
        - 1K ohm Resistor
        - 4.7K ohm Resistor
        - 10K ohm Resistor
        - 47 microFarad Electrolytic Capacitor
        - .01 microFarad Metal Film Capacitor
- 4 x 5mm Red LEDs
- 4 x 470 ohm Resistors
- 5 Volt Regulated DC Power Supply
- Assortment of 22 AWG Jumper Wires
- 2250 point Breadboard
- Logisim Program version 2.7.1
- Icarus Verilog version 0.9.7

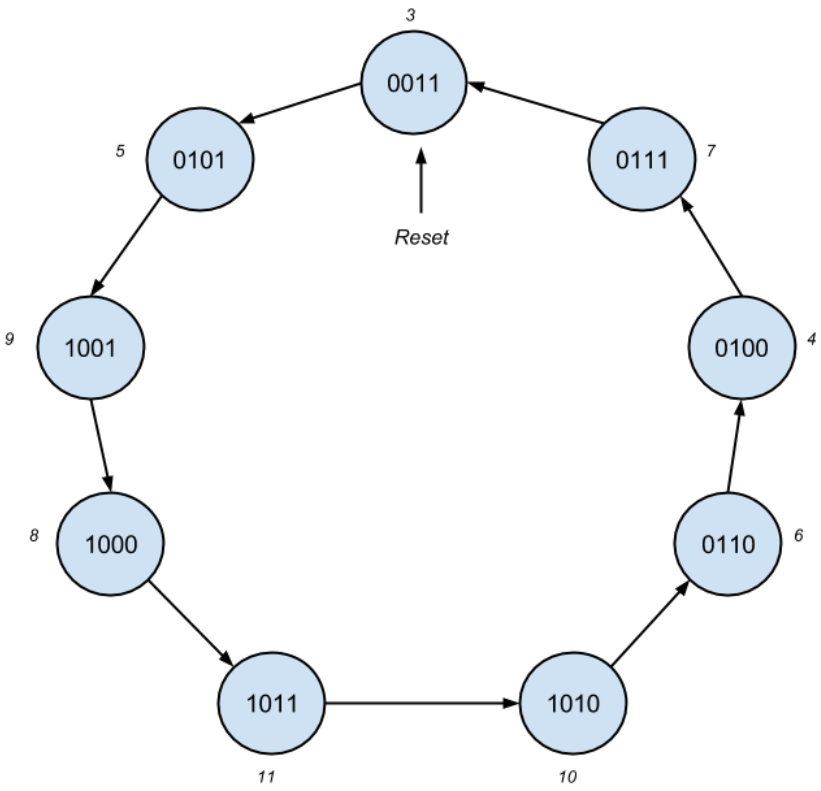
DESIGN DESCRIPTION

Implementing a counter is the simplest possible finite state machine as there is normally a single input to trigger state transitions and that being the leading edge of a clock.

For the nine-step counter presented in this lab assignment a strict adherence to the design procedure of finite state machines will lead to a simplified logic realization of the circuit. Per guidelines, a maximum of four D flip-flops are necessary to "store" a state of the circuit. Resetting the nine-step counter does not depend on synchronous logic and therefore can be ignored till after the basic counter design procedure.

State Transition Diagram

As shown below a circle with a 4-bit binary value represents a state of the counter and its corresponding output. The arrows after each circle point to the next state and will transition when a leading edge of a clock is present at the clock input for each of the D flip-flops.



State Transition Table

From the state transition diagram shown above it is possible to create a table correlating the present state to the next state of the circuit. Let A be the most significant bit and D be the least significant bit where A, B, C, and D are the storage elements of the circuit (D flip-flops).

	Previous				Next			
	A	**B**	**C**	**D**	**A+**	**B+**	**C+**	**D+**
3	0	0	1	1	0	1	0	1
5	0	1	0	1	1	0	0	1
9	1	0	0	1	1	0	0	0
8	1	0	0	0	1	0	1	1
11	1	0	1	1	1	0	1	0
10	1	0	1	0	0	1	1	0
6	0	1	1	0	0	1	0	0
4	0	1	0	0	0	1	1	1
7	0	1	1	1	0	0	1	1

Next-State Logic

Expressing each next-state bit as a combinational logic function of the present state bits will give the relevant logic at the input to each flip-flop. Examining each next-state bit shows the minterms and maxterms for the combinational logic but it is also possible to consider the "don't care" conditions given by unused states represented in 4-bits (0, 1, 2, 12, 13, 14, & 15).

Plotting the minterms, maxterms, and don't care values for each next-state bit yields the following Karnaugh maps and their equations as a function of the minterms in sum-of-products form.

A+

CD \ AB	0 0	0 1	1 1	1 0
0 0	x	0	x	1
0 1	x	1	x	1
1 1	0	0	x	1
1 0	x	0	x	0

=> B'C' + C'D + AD

B+

CD \ AB	0 0	0 1	1 1	1 0
0 0	x	1	x	0
0 1	x	0	x	0
1 1	1	0	x	0
1 0	x	1	x	1

=> A’D’ + CD’ + A’B’

C+

CD \ AB	0 0	0 1	1 1	1 0
0 0	x	1	x	1
0 1	x	0	x	0
1 1	0	1	x	1
1 0	x	0	x	1

=> C’D’ + BCD + AC

D+

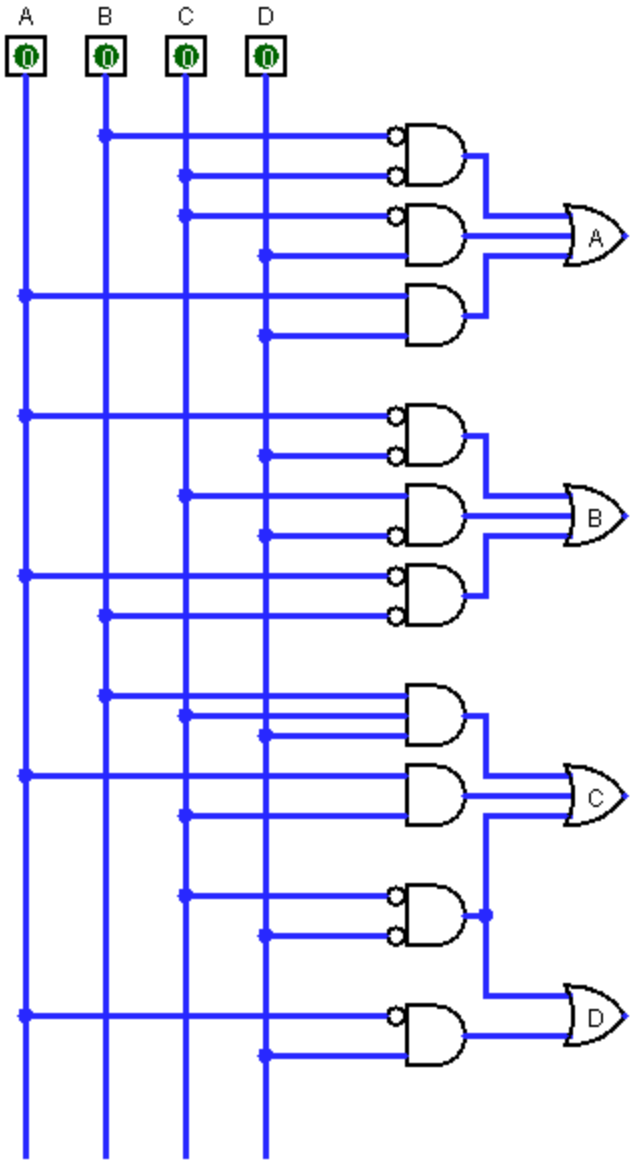
CD \ AB	0 0	0 1	1 1	1 0
0 0	x	1	x	1
0 1	x	1	x	0
1 1	1	1	x	0
1 0	x	0	x	0

=> C’D’ + A’D

Applying a combinational logic simplification technique of extraction reduces the number of gates essential for a functional circuit, thus let X = C’D’ then the next-state equations become…

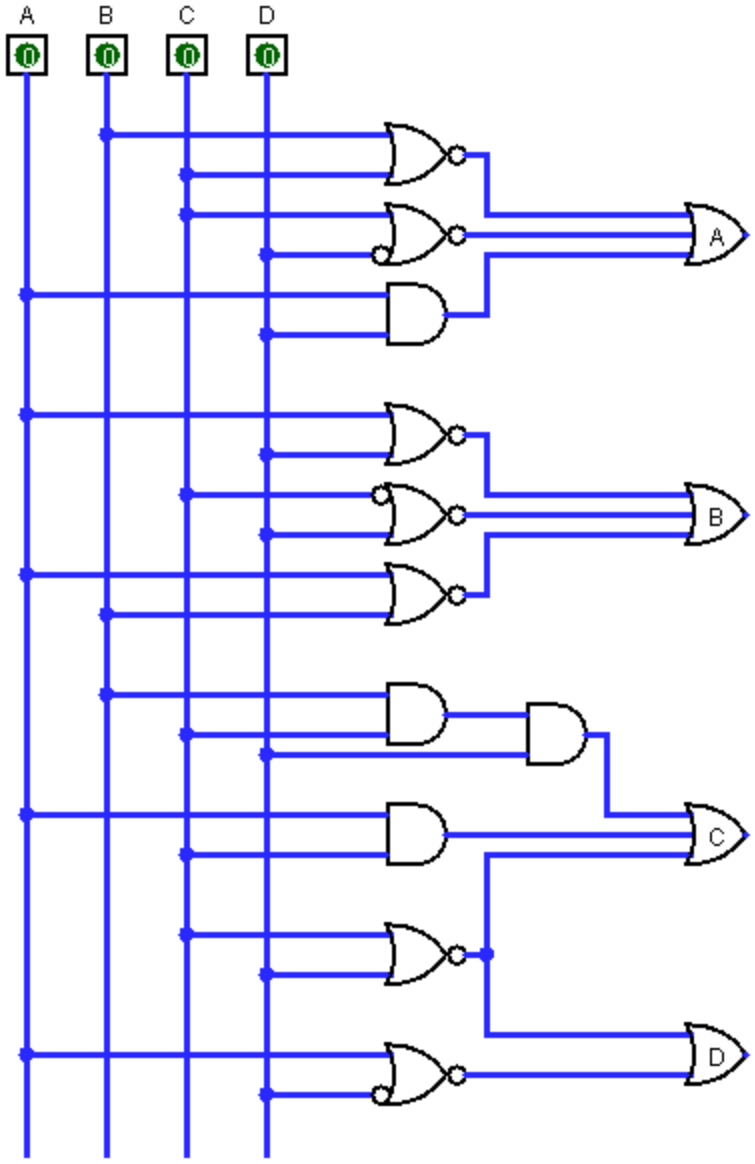
A+ = B’C’ + C’D + AD
B+ = A’D’ + CD’ + A’B’
C+ = **X** + BCD + AC
D+ = **X** + A’D

Implementing a two-level combinational network of logic gates results in the following logic circuit. Note that factoring out common terms for each next-state bit Boolean expression will lead to a more complicated multi-level combinational network that requires a greater amount of physical TTL ICs on the breadboard, which is very disadvantageous to the simplification process. It was a design choice to then forgo the factorization of the next-state Boolean expressions.



The two-level combinational network above would require a total of seven TTL ICs. At this point in the design it became apparent that there could be further ways to reduce the number of IC packages required. For instance converting the negated input AND gates to NOR gates would

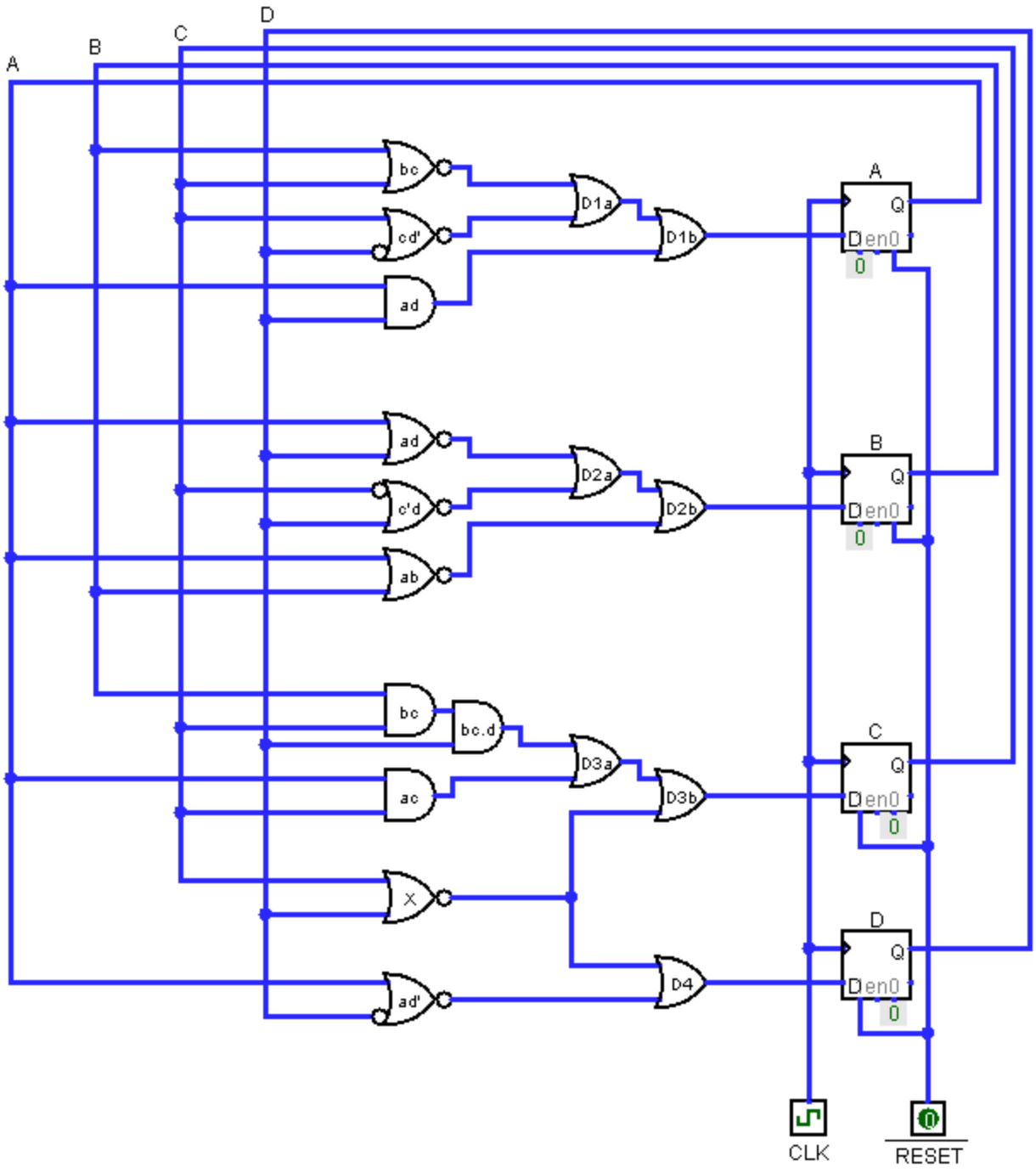
result in the following circuit where a total of six TTL ICs for the combinational network can be realized.



Thus the multilevel and multi-gate combinational implementation uses a reduced number of TTL ICs. However if you are to assume that there are only 2-input OR gates in the lab then the number of TTL ICs will still remain at a total of six not including the two D flip-flop ICs.

The last step in designing the circuit was combine the combinational logic network with the sequential logic network of D flip-flops and to address the reset requirement.

Resetting the counter to its original state is an asynchronous event that can be implemented using the asynchronous inputs on the D flip-flops (Preset and Clear). Whenever reset is asserted "hi" the next state of the counter will immediately become the initial state of 0011.
The complete circuit is shown below…



OBSERVATIONS AND DATA ANALYSIS

While designing the nine-step counter it was necessary to reduce the physical number of TTL ICs needed to implement the functionality set forth by the lab assignment guidelines and this was accomplished using combinational simplification techniques learned from chapter three of the accompanying textbook for the course. It was possible to use more techniques than what has been shown in the report but this only led to more wires and gates in Logisim.

Apparently there is a tradeoff between realized minimization of a circuit and the physical reduction in package count as each package contains a set number of logic gates and inputs complicating the real-world implementation.

Upon further investigation into the finalized circuit and the Boolean expressions for the next-state bits it was observed that static-hazards had not been accounted for in the combinational logic. Reviewing the Boolean expressions reveals that the Boolean expression for next-state bit C contains a static-1 hazard. This can be remedy by adding AD' to the expression thus C+ becomes…

$C+ = X + BCD + AC + AD'$

DISCUSSION AND CONCLUSION

Designing the nine-step counter was a moderately simple process when following the design procedure for a finite state machine outlined in chapter 7 of the textbook for the course. I was a bit nervous when beginning this assignment as finite state machines were still somewhat of a mystery after not having thoroughly grasped the concept in the preceding Computer Organization course. However I now feel confident in my ability to decipher somewhat more complex finite state machine problems.

The struggle with this weeks lab assignment was the design of the circuit in Verilog. Having little to no experience using the hardware description language made it difficult to comprehend the few examples of random circuits found online. This is an area of hardware design that I look forward to learning more of, and possibly implementing some of my own projects using VHDL or Verilog in the future.