# CSEE W3827

Fundamentals of Computer Systems

Homework Assignment 3

Solutions

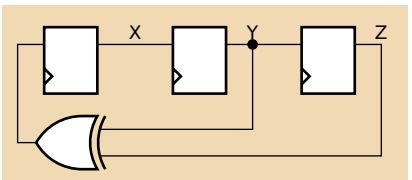
Prof. Stephen A. Edwards

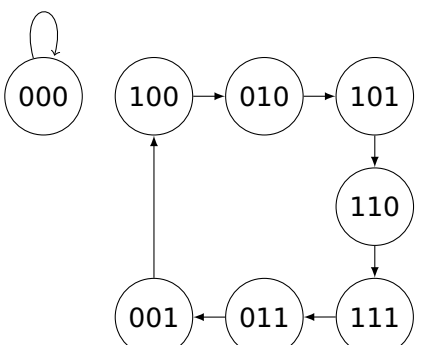
Columbia University

Due October 18th, 2011 at 10:35 AM

Show your work for each problem; we are more interested in how you get the answer than whether you get the right answer.

1. (10 pts.) The circuit below is called a linear-feedback shift register. Draw a bubble-and-arc diagram representing its behavior. Start from both the state $X = 1$, $Y = 0$, $Z = 0$ and the all-zeros state.



2. (15 pts.) The 1965 Ford Thunderbird had sequential turn signals that would light sequentially. Design an FSM that implements such signals.

Your machine has three inputs: LEFT, RIGHT, and HAZARD, and six light outputs, LA, LB, LC, RA, RB, and RC.

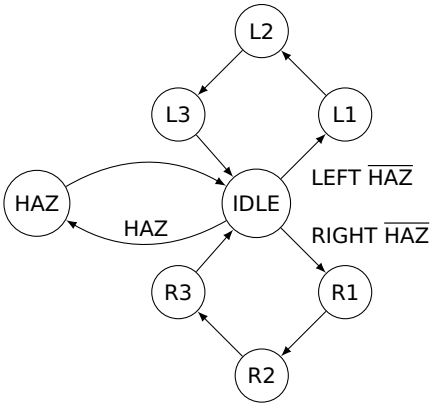
When HAZARD is active, your machine should alternate between all lights on and all lights off.

Otherwise, when LEFT is active, LA, then LA and LB, then all the L's should light, then all turn off, then repeat. RIGHT should do the same thing for the R outputs. Assume LEFT and RIGHT are mutually exclusive.

(a) Draw a Moore-style bubble-and-arc diagram for your machine.

(b) Draw a transition table for your machine with symbolic state names.

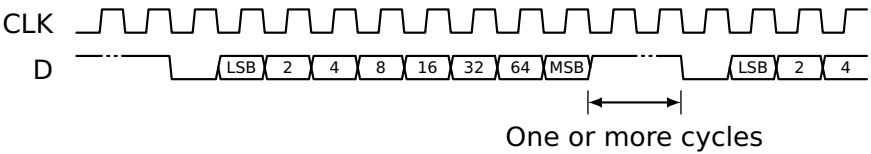
(c) Draw an output table for your machine expressing the L's and the R's in terms of your states.




| S | H | L | R | S' |
|---|---|---|---|---|
| IDLE | 0 | 0 | 0 | IDLE |
| IDLE | 1 | X | X | HAZ |
| IDLE | 0 | 1 | X | L1 |
| IDLE | 0 | X | 1 | R1 |
| HAZ | X | X | X | IDLE |
| L1 | X | X | X | L2 |
| L2 | X | X | X | L3 |
| L3 | X | X | X | IDLE |
| R1 | X | X | X | R2 |
| R2 | X | X | X | R3 |
| R3 | X | X | X | IDLE |

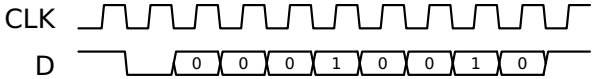
| S | LC | LB | LA | RA | RB | RC |
|---|---|---|---|---|---|---|
| IDLE | 0 | 0 | 0 | 0 | 0 | 0 |
| HAZ | 1 | 1 | 1 | 1 | 1 | 1 |
| L1 | 0 | 0 | 1 | 0 | 0 | 0 |
| L2 | 0 | 1 | 1 | 0 | 0 | 0 |
| L3 | 1 | 1 | 1 | 0 | 0 | 0 |
| R1 | 0 | 0 | 0 | 1 | 0 | 0 |
| R2 | 0 | 0 | 0 | 1 | 1 | 0 |
| R3 | 0 | 0 | 0 | 1 | 1 | 1 |

3. (20 pts.) Design a circuit for part of an HTML parser that signals when an ASCII character “H” (01001000 in binary) has arrived on a serial link. The rising edge of the clock indicates when the next bit is present on the input. Between bytes, the input stays high and the clock continues running. A byte starts with a single 0 start bit followed by the bits, LSB first, followed by a single 1 stop bit. The next byte could start immediately after the stop bit, or after an arbitrary number of 1’s.



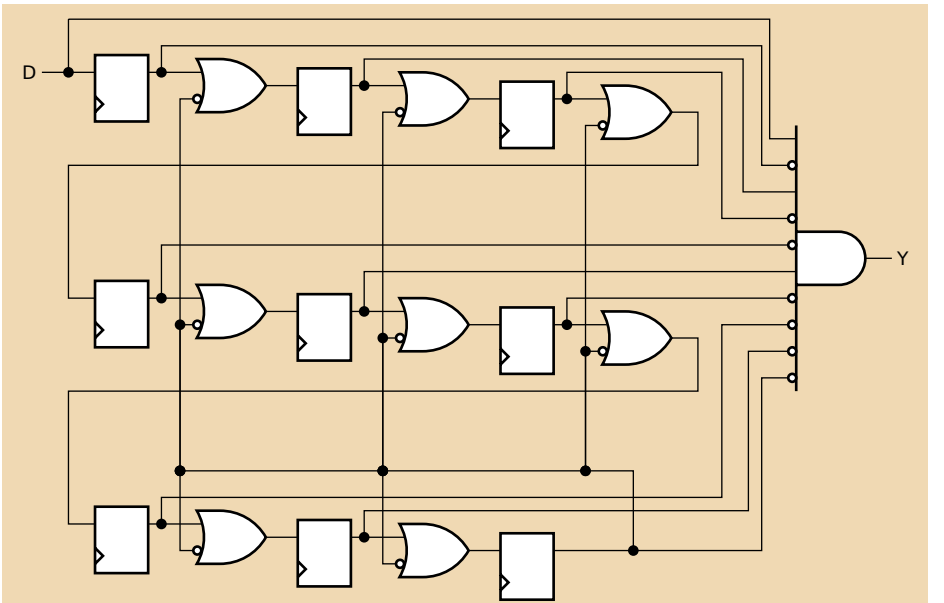

Assume the data stream starts in the idle state (i.e., not in the middle of a byte), then start looking for a start bit followed by the bit pattern, followed by a stop bit. Make sure your machine stays synchronized, i.e., so it will not erroneously report an H that crosses a stop/start bit boundary.

Consider modifying a shift register for part of your circuit.



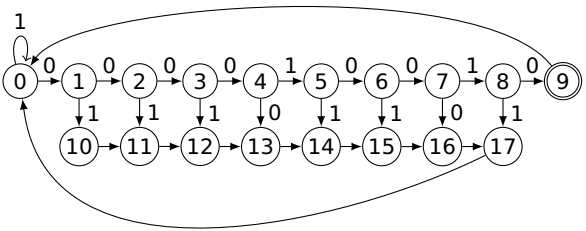

Trick: use the start bit to indicate when a complete byte has passed

| State | Comment |
|---|---|
| 1 11111111 1 | Reset state |
| 0 11111111 1 | Start bit received |
| 0 01111111 1 | 0 (LSB) |
| 0 00111111 1 | 0 |
| 0 00011111 1 | 0 |
| 1 00001111 1 | 1 |
| 0 10000111 1 | 0 |
| 0 01000011 1 | 0 |
| 1 00100001 1 | 1 |
| 0 10010000 1 | 0 (MSB) |
| 1 01001000 0 | Stop bit: check inner 8 and reset |
| 0 11111111 1 | Start bit received |




Needs to reset in the all-1’s state.

Moore Machine



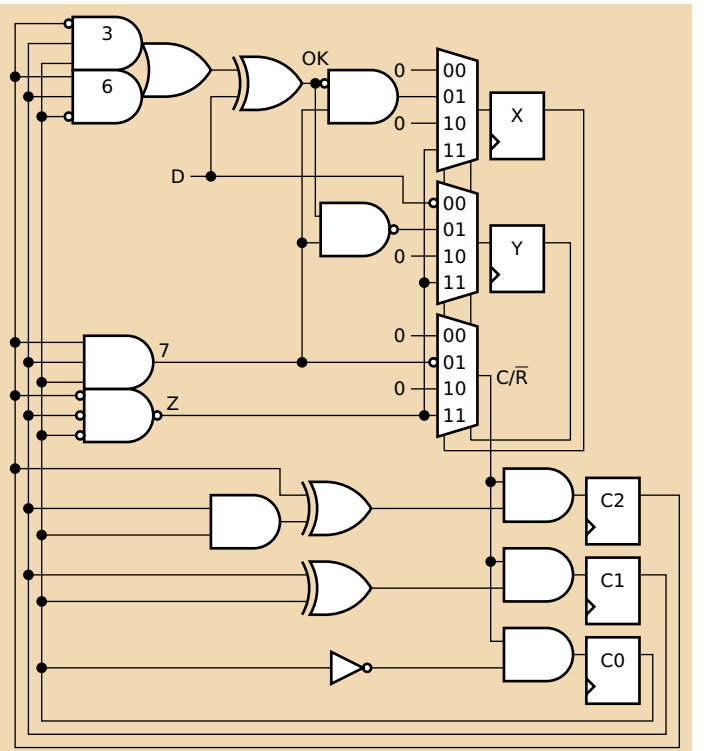

| S | XY ccc |
|---|---|
| 0 | 00 −−− |
| 1 | 01 000 |
| 2 | 01 001 |
| 3 | 01 010 |
| ⋮ | ⋮ |
| 8 | 01 111 |
| 9 | 10 −−− |
| 10 | 11 001 |
| 11 | 11 010 |
| ⋮ | ⋮ |
| 16 | 11 111 |
| 17 | 11 000 |

| XY | D | OK | Z | 7 | X'Y' | C/$\overline{R}$ |
|---|---|---|---|---|---|---|
| 00 | 0 | - | - | - | 01 | 0 |
| 00 | 1 | - | - | - | 00 | - |
| 01 | - | 0 | - | 0 | 11 | 1 |
| 01 | - | 1 | - | 0 | 01 | 1 |
| 01 | - | 0 | - | 1 | 11 | 0 |
| 01 | - | 1 | - | 1 | 10 | - |
| 10 | - | - | - | - | 00 | - |
| 11 | - | - | 0 | - | 11 | 1 |
| 11 | - | - | 1 | - | 00 | - |

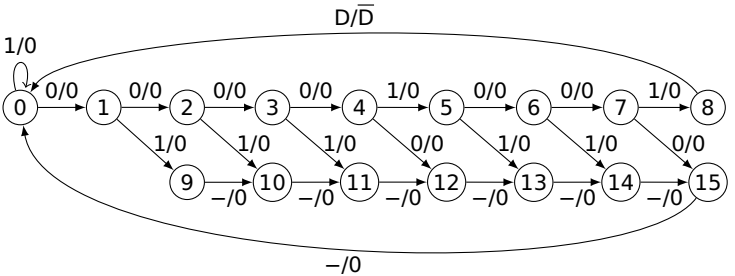
| S | D | S' |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 2 |
| 1 | 1 | 10 |
| 2 | 0 | 3 |
| 2 | 1 | 11 |
| 3 | 0 | 4 |
| 3 | 1 | 12 |
| 4 | 0 | 13 |
| 4 | 1 | 5 |
| 5 | 0 | 6 |
| 5 | 1 | 14 |
| 6 | 0 | 7 |
| 6 | 1 | 15 |
| 7 | 0 | 16 |
| 7 | 1 | 8 |
| 8 | 0 | 9 |
| 8 | 1 | 17 |
| 9 | - | 0 |
| 10 | - | 11 |
| 11 | - | 12 |
| 12 | - | 13 |
| 13 | - | 14 |
| 14 | - | 15 |
| 15 | - | 16 |
| 16 | - | 17 |
| 17 | - | 0 |

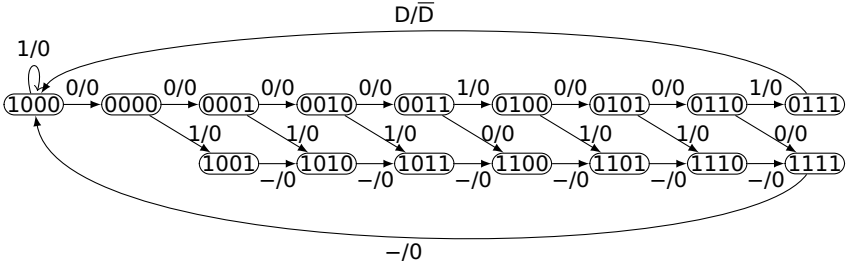
| XY | D | OK | Z | 7 | X'Y' | C/$\overline{R}$ |
|---|---|---|---|---|---|---|
| 00 | 0 | - | - | - | 01 | 0 |
| 00 | 1 | - | - | - | 00 | - |
| 01 | - | 0 | - | 0 | 11 | 1 |
| 01 | - | 1 | - | 0 | 01 | 1 |
| 01 | - | 0 | - | 1 | 11 | 0 |
| 01 | - | 1 | - | 1 | 10 | - |
| 10 | - | - | - | - | 00 | - |
| 11 | - | - | 0 | - | 11 | 1 |
| 11 | - | - | 1 | - | 00 | - |

| XY | X' | Y' | C/$\overline{R}$ |
|---|---|---|---|
| 00 | 0 | $\overline{D}$ | 0 |
| 01 | 7 $\overline{OK}$ | $\overline{OK}$ 7 | $\overline{7}$ |
| 10 | 0 | 0 | 0 |
| 11 | $\overline{Z}$ | $\overline{Z}$ | $\overline{Z}$ |



Mealy Machine






$$
\begin{aligned}
Y &= \overline{D}\,\overline{Q_3}Q_2Q_1Q_0 \\
D_3 &= Q_3(D+Q_2+Q_1+Q_0)+\overline{Q_3}Q_2Q_1Q_0+ \\
&\quad D(\overline{Q_3}\,\overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}+\overline{Q_3}\,\overline{Q_2}\,\overline{Q_1}\,Q_0+\overline{Q_3}\,\overline{Q_2}\,Q_1\,\overline{Q_0}+\overline{Q_3}\,Q_2\,\overline{Q_1}\,\overline{Q_0}+\overline{Q_3}\,Q_2\,\overline{Q_1}\,Q_0)+ \\
&\quad \overline{D}(\overline{Q_3}\,\overline{Q_2}\,Q_1\,Q_0+\overline{Q_3}\,Q_2\,Q_1\,\overline{Q_0}) \\
&= Q_3(D+Q_2+Q_1+Q_0)+Q_2Q_1Q_0+ \\
&\quad D(\overline{Q_2}\,\overline{Q_1}+\overline{Q_2}\,Q_1\,\overline{Q_0}+Q_2\,\overline{Q_1})+\overline{D}Q_1(\overline{Q_2}\,Q_0+Q_2\,\overline{Q_0}) \\
D_2 &= Q_2 \oplus (Q_1Q_0) \\
D_1 &= Q_1 \oplus Q_0 \\
D_0 &= \overline{Q_0}(\overline{Q_3}+Q_2+Q_1)
\end{aligned}
$$

4.

(30 pts.) Design a synchronous 7-bit counter that counts 0-59 in BCD, i.e., the seconds digits of a "binary clock." There are seven outputs, $Q_1$, $Q_2$, $Q_4$, $Q_8$, $Q_{10}$, $Q_{20}$, and $Q_{40}$, each driven by a D flip-flop.

(a) Write Boolean expressions of the form $D_i = Q_i \oplus (\cdots)$ for each flip-flop's input. ($\oplus$ is XOR)

$$
\begin{aligned}
D_1 &= \overline{Q_1} \\
D_2 &= Q_2 \oplus (\overline{Q_8} Q_1) \\
D_4 &= Q_4 \oplus (Q_2 Q_1) \\
D_8 &= Q_8 \oplus (Q_4 Q_2 Q_1 + Q_8 Q_1) \\
D_{10} &= Q_{10} \oplus (Q_8 Q_1) \\
D_{20} &= Q_{20} \oplus (\overline{Q_{40}} Q_{10} Q_8 Q_1) \\
D_{40} &= Q_{40} \oplus ((Q_{40} + Q_{20}) Q_{10} Q_8 Q_1)
\end{aligned}
$$

(b) Write a small program to test your expressions and attach both it and its output. I wrote a C program like the one below, but you may use any language.

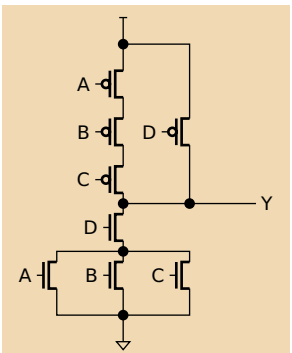
```
#include <stdio.h>

int main()
{
  int i, q1, q2, q4, q8, q10, q20, q40,
      d1, d2, d4, d8, d10, d20, d40;
  q1 = q2 = q4 = q8 = q10 = q20 = q40 = 0;
  for (i = 0 ; i < 121 ; ++i) {
    printf("%2d %c%c%c %c%c%c%c\n", i, ... );
    d1 = ~q1;
    d2 = q2 ^ (q1 & ~q8);
    d4 = q4 ^ (q1 & q2);
    d8 = q8 ^ (q4 & q2 & q1 | q8 & q1);
    d10 = q10 ^ (q8 & q1);
    d20 = q20 ^ (q10 & ~q40 & q8 & q1);
    d40 = q40 ^ ((q20 | q40) & q10 & q8 & q1);
    q1  = d1; q2  = d2;  q4  = d4;
    q8 = d8;  q10 = d10; q20 = d20;
    q40 = d40;
  }
  return 0;
}
```

The output:

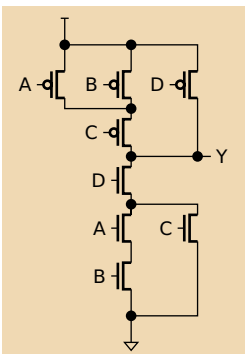
```
 0 000 0000
 1 000 0001
 2 000 0010
 3 000 0011
 4 000 0100
 5 000 0101
 6 000 0110
 7 000 0111
 8 000 1000
 9 000 1001
10 001 0000
 ...
19 001 1001
20 010 0000
 ...
59 101 1001
60 000 0000
```

5. (10 pts.)

(a) Write a Boolean expression for the function of the following static CMOS gate.

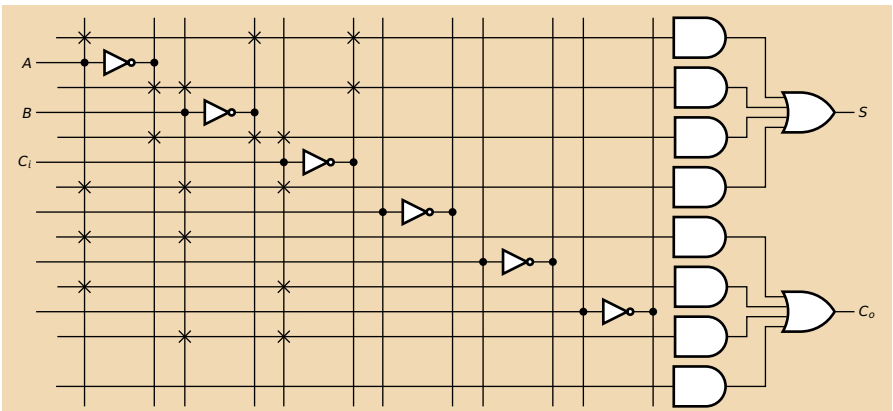



$Y = \overline{(A + B + C)D}$




(b) Draw the schematic for a static CMOS gate that implements $Y = \overline{(AB + C)D}$

6. (15 pts.) Show how to implement a full adder using the following PLA. The inputs should be $A$, $B$, and $C_i$ and the outputs $S$ and $C_o$ Hint: write the functions for the sum and carry in sum-of-products form then draw crosses to indicate connections on the AND plane.




$S = A \oplus B \oplus C_i = A\overline{B}\,\overline{C_i} + \overline{A}B\overline{C_i} + \overline{A}\,\overline{B}C_i + ABC_i$

$C_o = AB + AC_i + BC_i$