CSEE 3827: Homework 3 — Due: 3/5/10, 5PM

- *You may submit this homework by bringing it by CSB 469, bringing it to a TA's office hours, or electronically on courseworks.*
- *You may specify your designs using whatever combination of schematics and boolean expressions makes sense to you.*

1. **Sequence recognizer:** Implement a finite state machine that has the following inputs:

   - CLK, a clock (1 bit)
   - $\overline{RST}$, a reset signal (1 bit, active low)
   - I_3, I_2, I_1, I_0, a hexadecimal digit (4 bits)

   and produces a single bit of output (F). The input bits encode digits, so $I_3I_2I_1I_0 = 0101$ encodes the value 5. The output F should be high if and only if the machine has seen the sequence of inputs "3", "8", "2", "7" on four successive cycles. After matching the string "3827", the machine should set F back to 0 and resume searching for another instance of the string.

   You should implement a Moore machine, using any type of flip flop you like. You may specify your machine via schematic or boolean expressions.

2. **Serial priority encoder:** In this exercise, you will implement a serial priority encoder. The top level interface of the encoder is shown in Figure 1.

Figure 1: Block diagram of serial priority encoder (SPRIORITY).

The two-bit output P indicates the position of the most significant 1 in each 4-bit serial input word. You should assume the 4-bit word is arriving least-significant bit first and most-significant bit last. The output V is true if and only if a 1 occurred in the preceding input word. These outputs are to hold their values for four cycles until a new input word has been read. The table below shows the desired behavior.

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}
$\overline{RST}$:	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
IN:	X	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	X	X
V:	X	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	X	X
P:	X	X	X	X	X	2	2	2	2	X	X	X	X	3	3	3	3	X	X

You should implement this encoder piece by piece, by implementing the four components shown Figure 2. In all cases you should use Moore machines.

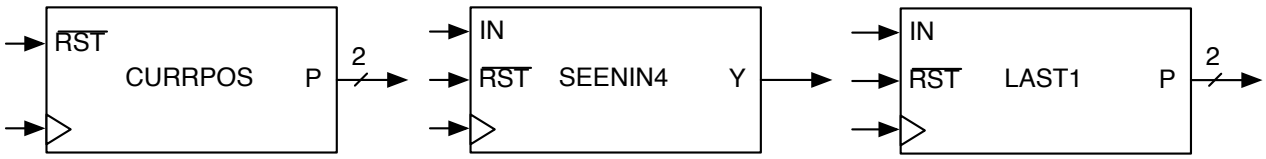

Figure 2: Block diagram of four sub-components of the serial priority encoder.

(a) Start by implementing the CURRPOS machine. The two-bit output of this machine should indicate the current bit position of the input word, as shown in this table:

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}
$\overline{RST}$:	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
P:	X	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	X	X

(b) Implement the SEENIN4 machine whose output, Y, is true if the machine has seen a 1 on IN in the last four cycles. It is 0 otherwise, as shown in this table:

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}
$\overline{RST}$:	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
IN:	X	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	X	X
Y:	X	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	X	X

(c) Implement the LAST1 machine. The output of LAST1, P, should show the position mod 4 of the last 1 seen on the input.

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}
$\overline{RST}$:	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
IN:	X	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	X	X
P:	X	X	0	0	2	2	X	X	X	X	X	X	X	3	0	0	0	X	X

(d) Finally, use these three components to implement the full SPRIORITY encoder such that it behaves as specified at the start of this problem.

3. **Tic Tac Toe:** In this exercise you will implement a tic-tac-toe game.

   (a) Start by implementing a Moore state machine for a *single square*. This state machine takes the following inputs:

   - CLK, a clock
   - $\overline{RST}$, an active-low reset signal
   - $MARK$, indicates that a player is marking the square
   - $PLAYER$, indicates which player is marking the square

   The machine outputs the following three bits

   - $MARKED$, indicates whether or not the square has been marked
   - $OWNER$, indicates which player has marked the square
   - ERR, indicates that there has been an error

   An error occurs when a player attempts to mark an already-marked square. Once an error has occurred, the square stays in an error state until it is reset.

   (b) We will now use the above state machines to implement a game board. The game board circuit should include 9 squares, as well as some additional circuitry to implement the rest of the game's functionality. In each cycle, the board takes the following inputs:

   - CLK, a clock (1 bit)
   - $\overline{RST}$, a reset signal (1 bit, active low)
   - A move which consists of the following five bits of information:
     - $PLAYER$, a single bit indicating which player is making the move
     - ROW, two bits indicating which row the player wants to mark
     - COL, two bits indicating which column the player wants to mark

   The board outputs the following information:

   - $GAMEOVER$, goes high when the game is over either because a player has won or because there has been an error
   - ERR, indicates an error
   - $DRAW$, indicates the game concluded with a draw (i.e., all squares are marked and there is no winner)
   - $WINNER$, a single bit indicating which player won

   The board enters an error state when a player attempts two consecutive moves or when any individual square is in an error state.