# University of Toronto
# Faculty of Applied Science and Engineering
# Edward S. Rogers Sr. Department of Electrical and Computer Engineering

## Final Examination
## ECE 241F - Digital Systems

## Examiners: S. Brown, J. Rose, K. Truong and B. Wang
## December 9, 2004

## Duration: 2.5 Hours

ANSWER ALL QUESTIONS ON THESE SHEETS, USING THE BACKS IF NECESSARY.

- Exam Type D, these specific aids allowed:
  i. Original Versions (no photocopies) of the course text, **Fundamentals of Digital Logic with Verilog Design,** by Brown & Vranesic, ISBN 0-07-282315-1.
  ii. One 8.5 x 11" two-sided aid sheet.

- The amount of marks available for each question is given in square brackets [].
- No calculators of any type are allowed. Cellular phones are also prohibited.

LAST NAME: SOLUTIONS

FIRST NAME: _____

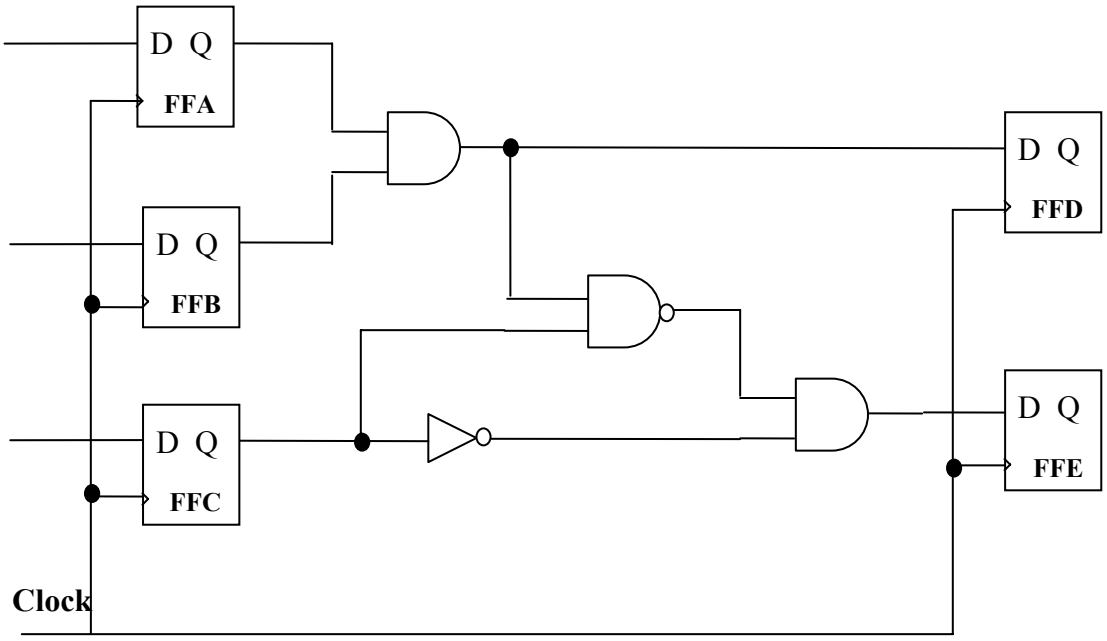
STUDENT NUMBER: _____

Lecture Section: Section 01 (Rose) [ ]
Section 02 (Wang) [ ]
Section 03 (Brown) [ ]
Section 04 (Truong) [ ]

| Question | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| Maximum Mark | 8 | 10 | 6 | 9 | 6 | 6 | 8 | 8 | 10 | 71 |
| Mark Obtained | | | | | | | | | | |

[8] Q1. Consider the circuit below and the table giving various maximum propagation delays and other timing parameters for the circuit elements.

| **Timing Parameter** | **Maximum (ns)** |
|---|---|
| $T_{INV}$ – Delay of Inverter | 2.0 |
| $T_{AND}$ – Delay of AND | 5.0 |
| $T_{NAND}$– Delay of NAND | 4.0 |
| Flip-Flop $T_{Clock\text{-}to\text{-}Q}$ | 3.0 |
| Flip-Flop $T_{Set\text{-}up}$ | 2.0 |
| Flip-Flop $T_{Hold}$ | 1.5 |




(a) [4] Determine the minimum clock period for which this circuit is guaranteed to operate correctly. Show how you arrived at your answer.

Answer: Period = $T_{Clock\text{-}to\text{-}Q} + T_{AND} + T_{NAND} + T_{AND} + T_{Set\text{-}up} = 3 + 5 + 4 + 5 + 2 = 19$ ns

1 mark off for each missing or erroneous part of the equation, up to max of 4.

Question 1, continued

(b) [4] Assume that the *minimum* propagation delay for all gates and the clock-to-Q of the flip-flops is 0.5ns. Will the hold time for the Flip-Flops labeled **FFD** and **FFE** be violated? Give an answer for *both* flip-flops and explain each answer.

Answer: FFD has a hold time violation because Th is 1.5ns, and the minimum time for a transition on the D input to FFD is a min of Tclock-to-q (a->d) + min of Tand (d->g) = 0.5 + 0.5 = 1ns < 1.5ns -> violation.

2 marks – 0.5 for answer (violation), 1.5 for explanation.

FFE does not have a hold time violation. The quickest change that could occur would come through the path cfhj or cfij, both of which have a clock-to-q + min gate + min gate = 0.5 +0.5 +_0.5 = 1.5ns, just meeting the hold time, so no violation.

2 marks – 0.5 for answer (no violation), 1.5 for explanation

[10] Q2. Consider the one-input (**X**) one-output (**Out**) 3-state Moore-type Finite State Machine described in the State Transition and Assignment Table given below. Notice that this is a fully-encoded state machine for which the codes for States A and C have already been assigned, but the code for State B has not yet been determined. The code for state B is listed as $\mathbf{b_1\ b_0}$ in the table.

You are to choose the code for State B (without changing the given code for A and C) such that the total cost of the sum-of-products form of the resulting **Next State** logic and the **Output** logic is minimized. To measure cost, use the (# gates) + (# inputs) metric, in which you **should include** the cost of inverters.

You must show all of your work in the area provided, including how you decided which encoding resulted in the best cost. At the end, you must give the circuit schematic (diagram) of the state machine.

In the state transition and assignment table below, the *current state* (the flip-flop outputs) of the corresponding finite state machine are denoted $Q_1Q_0$, and the next state (flip-flop inputs) are denoted as $D_1D_0$.

| **Current State** | | **Next State** | | | | | | **Out** |
|---|---|---|---|---|---|---|---|---|
| | | **X = 0** | | | **X = 1** | | | |
| | **$Q_1Q_0$** | | **$D_1$** | **$D_0$** | | **$D_1$** | **$D_0$** | |
| A | 00 | A | 0 | 0 | B | $b_1$ | $b_0$ | 1 |
| B | $b_1b_0$ | C | 0 | 1 | B | $b_1$ | $b_0$ | 0 |
| C | 01 | A | 0 | 0 | B | $b_1$ | $b_0$ | 1 |

Below are some extra state tables that will help you determine your answer:

Table for State Encoding B 10

| **Current State** | | **Next State** | | | | **Out** |
|---|---|---|---|---|---|---|
| | | **X = 0** | | **X = 1** | | |
| | **$Q_1Q_0$** | **$D_1$** | **$D_0$** | **$D_1$** | **$D_0$** | |
| A | 00 | 0 | 0 | **1** | **0** | 1 |
| B | **10** | 0 | 1 | **1** | **0** | 0 |
| C | 01 | 0 | 0 | **1** | **0** | 1 |

**D1 = X - cost =0**

| X/Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | d | 0 |
| 1 | 1 | 1 | d | 1 |

**D0 = X'Q1 - cost = 2 gates + 3 inputs = 5**

| X/Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | d | 1 |
| 1 | 0 | 0 | d | 0 |

**Out = Q1' – cost = 1 gate + 1 input = 2**

| Q0/Q1 | 0 | 1 |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | d |

**Total cost = 0 + 5 + 2 = 7**

.Table for State Encoding B 11

| **Current State** | | **Next State** | | | | **Out** |
|---|---|---|---|---|---|---|
| | | **X = 0** | | **X = 1** | | |
| | **$Q_1Q_0$** | **$D_1$** | **$D_0$** | **$D_1$** | **$D_0$** | |
| A | 00 | 0 | 0 | **1** | **1** | 1 |
| B | **11** | 0 | 1 | **1** | **1** | 0 |
| C | 01 | 0 | 0 | **1** | **1** | 1 |

Work area:

**D1 = X - cost = 0**

| X/Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | d |
| 1 | 1 | 1 | 1 | d |

**D0 = = X + Q1 – cost = 1 gate + 2 inputs = 3**

| X/Q1Q0 | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | d |
| 1 | 1 | 1 | 1 | d |

**Out = S1' – cost = 1 gate + 1 input = 2**

| Q0/Q1 | 0 | 1 |
|---|---|---|
| 0 | 1 | d |
| 1 | 1 | 0 |

**Total Cost = 0 + 3 + 2 = 5**

**Cost of Code 10 = 7 > Cost of Code 11 = 5 -> Code 11 is better**

Marking: 0.5 for each state table filled out correctly (1)
1 Mark each for D1 calculated correctly (2)
1 Mark each for each D0 calculated correctly (2)
0.5 for each Out calculated correctly (1)
1 for correct final answer – total 7

(a) [7] Answer: The Best Code for state B is:_____
Be sure to show all of your work above, including how you determined the cost of all of the alternative codes for B.

Q2, continued.

(b) [3] Give the full state machine design, (using D-type flip-flops, AND, OR and NOT gates,) of the state machine that uses the encoding that results in the minimum-cost logic.

Two flip-flops, one OR gate. (-1 for each error). If final answer in part (a) is wrong, check and see if circuit diagram is correct from what the state answer is. 1 mark off for each error, up to max of 3.

[6] Q3. Consider the following logic function:

$$\mathbf{f\,(x_3,x_2,x_1,x_0) = \Sigma\ m\ (0,\ 4,\ 7,\ 8,\ 9,\ 12,\ 13,\ 15)}$$

(a) [3] Determine the minimal sum-of-products form for the logic function. Make use of the following Karnaugh map:

| $x_3x_2$ / $x_1x_0$ | **00** | **01** | **11** | **10** |
|---|---|---|---|---|
| **00** | 1 | 1 | 1 | 1 |
| **01** | 0 | 0 | 1 | 1 |
| **11** | 0 | 1 | 1 | 0 |
| **10** | 0 | 0 | 0 | 0 |

Answer:

$$f = x_1'x_0' + x_3x_1' + x_2x_1x_0$$

Marking scheme: 1 marking for filling in the K-map correctly. 2 marks for deriving the correct function from the K-map.

Q3, continued

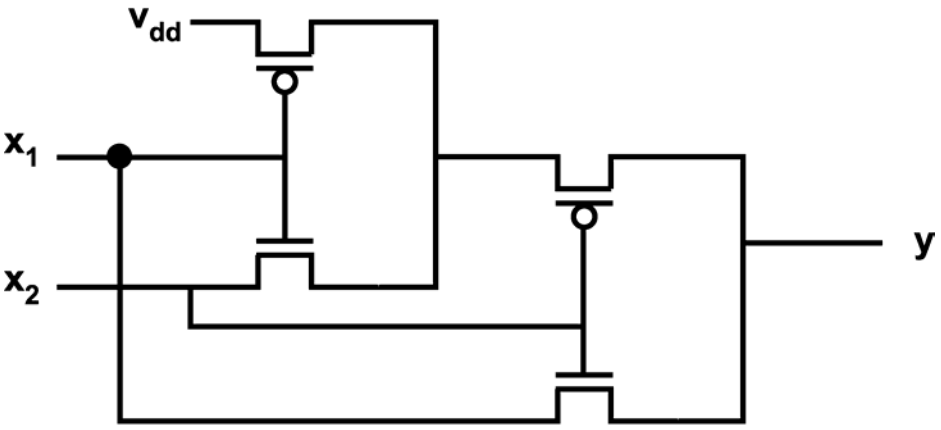
(b) [3] Implement the above function using only one 4:1 multiplexor and as many basic gates (NOT, AND, OR) as necessary. Note: full marks will be given to a solution that uses no basic gates.

Solution: A 4:1 multiplexor with two selection bits – ($x_1$,$x_0$). When (0,0), f=1; when (0,1), f=$x_3$; when (1,0), f=0; when (0,1), f=$x_2$.

Marking scheme: 3 marks for the correct answer; 2 marks for correct answer that uses gates; 1 mark for any correct answer.

[9] Q4.

(a) [3] Consider the following transistor-level design of a 2-input logic gate. Derive the truth table of the output **y** in terms of the inputs $\mathbf{x_1}$ and $\mathbf{x_2}$. Place your answer in the truth table beside the circuit



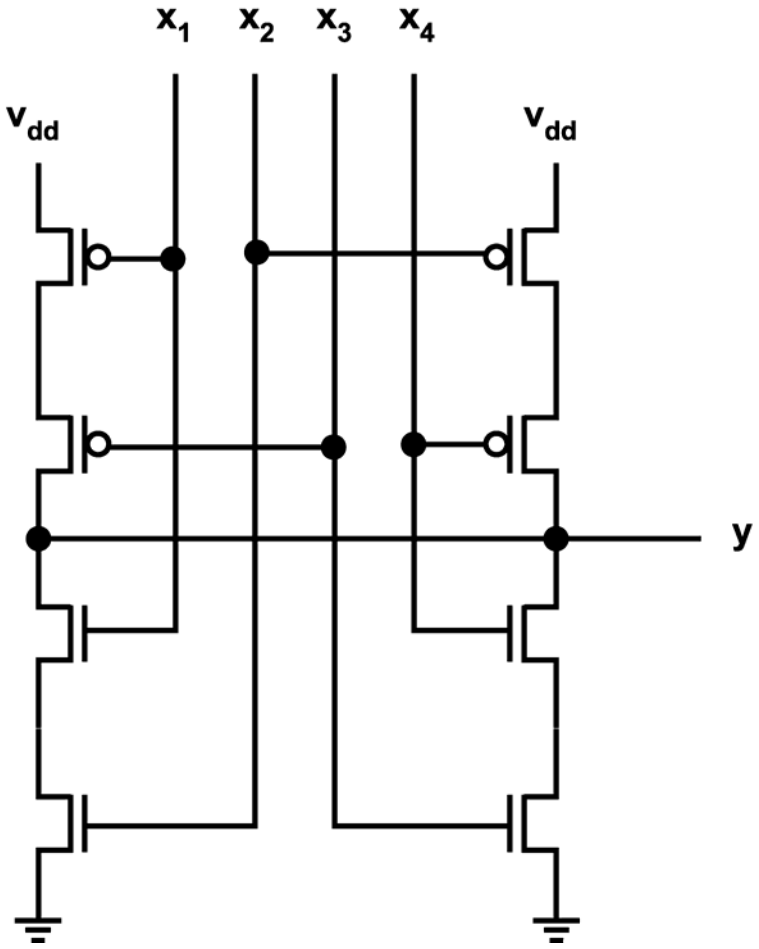

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | **1** |
| 0 | 1 | **0** |
| 1 | 0 | **0** |
| 1 | 1 | **1** |

Put your rough work here:

Solution: y = XNOR (x1,x2)
Marking scheme : -1 mark for each incorrect row in the truth table, up to maximum of 3.

Q4, continued

(b) [6] Consider the transistor level design of the 4-input gate below. Determine the truth table of the output **y** in terms of the inputs $x_1$, $x_2$, $x_3$ and $x_4$. **For the cases where y is not driven to 0 or 1, give the answer 'Z'**. Place your answer in the truth table give at the right.




| $x_1$ | $x_2$ | $x_3$ | $x_4$ | y |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | Z |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | Z |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Marking scheme : -0.5 mark for each incorrect row, up to maximum of 6.

[6] Q5. Consider the two-input, two-output State Machine described by the state transition table below.

| Present State | Next State | | | | Outputs | |
|---|---|---|---|---|---|---|
| | $x_1x_2$=00 | $x_1x_2$=01 | $x_1x_2$=10 | $x_1x_2$=11 | $z_1$ | $z_2$ |
| $S_0$ | $S_0$ | $S_6$ | $S_2$ | $S_4$ | 1 | 0 |
| $S_1$ | $S_0$ | $S_3$ | $S_1$ | $S_5$ | 0 | 1 |
| $S_2$ | $S_1$ | $S_6$ | $S_2$ | $S_4$ | 1 | 1 |
| $S_3$ | $S_1$ | $S_0$ | $S_4$ | $S_6$ | 0 | 1 |
| $S_4$ | $S_0$ | $S_3$ | $S_2$ | $S_0$ | 1 | 0 |
| $S_5$ | $S_1$ | $S_4$ | $S_0$ | $S_1$ | 0 | 1 |
| $S_6$ | $S_1$ | $S_0$ | $S_4$ | $S_3$ | 0 | 1 |

(a) [4] Determine the minimum number of states that could be used to implement this state machine, showing your work

Solution:

$P_0 = (S_0S_1S_2S_3S_4S_5)$
$P_1 = (S_0S_4)(S_2)(S_1S_3S_5S_6)$
$P_2 = (S_0S_4)(S_2)(S_1)(S_3S_5S_6)$
$P_3 = (S_0S_4)(S_2)(S_1)(S_3S_6)(S_5)$
$P_4 = (S_0S_4)(S_2)(S_1)(S_3S_6)(S_5)$

Marking scheme: 1 mark for $P_0$, $P_1$, $P_2$, and $P_3$

Q5, continued

(b) [2] Give the state transition table for the minimized state machine:

| Present State | Next State | | | | Outputs | |
|---|---|---|---|---|---|---|
| | $x_1x_2$=00 | $x_1x_2$=01 | $x_1x_2$=10 | $x_1x_2$=11 | $z_1$ | $z_2$ |
| $S_0/S_4$ | $S_0/S_4$ | $S_3/S_6$ | $S_2$ | $S_0/S_4$ | 1 | 0 |
| $S_1$ | $S_0/S_4$ | $S_3/S_6$ | $S_1$ | $S_5$ | 0 | 1 |
| $S_2$ | $S_1$ | $S_3/S_6$ | $S_2$ | $S_0/S_4$ | 1 | 1 |
| $S_3/S_6$ | $S_1$ | $S_0/S_4$ | $S_0/S_4$ | $S_3/S_6$ | 0 | 1 |
| $S_5$ | $S_1$ | $S_0/S_4$ | $S_0/S_4$ | $S_1$ | 0 | 1 |

Marking scheme: -0.5 for each incorrect field based on the minimization in the first part, up to maximum of 2..

[6] Q6. Consider the following two logic functions:

$$f = x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6$$
$$g = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4 \cdot \bar{x}_5 \cdot \bar{x}_6$$

You are to implement these two functions using a total of three 4-input lookup tables. Show what logic function has to be implemented in each of the three lookup tables by giving the logic expression as a function of its labeled inputs (in the case where there isn't a label, you should create one).

Use the simplest logic expressions possible for each lookup table, using AND, OR, and NOT operators. Show your logic expression inside each lookup table, and show how the lookup tables are connected together.

Note that some part marks will be given if you use four or more lookup tables, but to get full marks you have to use exactly three lookup tables.

**ANSWER**:

h = x1 x2 x3 x4

f = h x1 x4 x5 x6

g = h !x1 !x4 !x5 !x6

note that there is nothing special about the group x1, x2, x3, x4 in the expression for h; any four inputs can be used. Also, in the expressions for f and g, there is nothing special about x1; any of the four inputs used for f can be used instead of x1. Note that the expression for g could also be written as
g = !(!h + x1 + x4 + x5 + x6).

The solution can be given by using logic expressions as shown here, or by drawing a circuit that has three LUTs.

Marks: 2 for the expression for h, 2 for f, and 2 for g.

Marks: 3 marks for a correct circuit that has four LUTs.

Marks: 2 marks for a correct circuit with more than four LUTs.

[8] Q7. Consider the Verilog code for a finite state machine shown below:

```
module whatsthis (Clock, Resetn, In3, Out);
    input Clock, Resetn, In3;
    output [1:3] Out;
    reg [1:3] y, Y;
    parameter [1:3] A = 3'b000, B = 3'b001, C = 3'b010,
        D = 3'b011, E = 3'b100, F = 3'b101, G = 3'b110, H = 3'b111;

    // Define the next state combinational circuit
    always @(In3 or y)
        case (y)
                A:    if (In3) Y = B;
                      else     Y = A;
                B:    if (In3) Y = D;
                      else     Y = C;
                C:    if (In3) Y = F;
                      else     Y = E;
                D:    if (In3) Y = H;
                      else     Y = G;
                E:    if (In3) Y = B;
                      else     Y = A;
                F:    if (In3) Y = D;
                      else     Y = C;
                G:    if (In3) Y = F;
                      else     Y = E;
                H:    if (In3) Y = H;
                      else     Y = G;
        endcase

    // Define the sequential block
    always @(negedge Resetn or posedge Clock)
        if (Resetn == 0) y <= A;
        else y <= Y;

    // Define output
    assign Out = y;

endmodule
```

(a) [4] In the table at the top of the next page give the state-assigned table (the table that shows the state codes and not just the letters A, B, etc for the state names) for this FSM.

Q7, continued

**ANSWER for State-Assigned Table**:

| | PS y | In3 = 0 Y | In3 = 1 Y | Out |
|---|---|---|---|---|
| A | 000 | 000 | 001 | 000 |
| B | 001 | 010 | 011 | 001 |
| C | 010 | 100 | 101 | 010 |
| D | 011 | 110 | 111 | 011 |
| E | 100 | 000 | 001 | 100 |
| F | 101 | 010 | 011 | 101 |
| G | 110 | 100 | 101 | 110 |
| H | 111 | 110 | 111 | 111 |

Marks: 3. If Out column is missing, give 2. For every mistake, -1 to a minimum of 0/3.

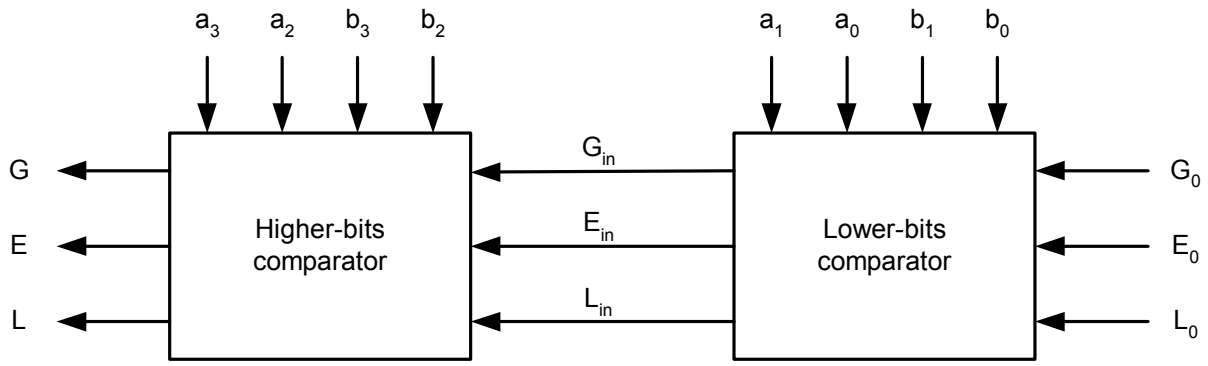
(b) [4] State in words, as simply as possible, what function is performed by this FSM.

**ANSWER**: This FSM is a shift register with the serial input In3 and the parallel output Out. If we arrange the flip-flops in the order y1 y2 y3, then it is a right-to-left shift register. If we put the flip-flops in the arrangement y3 y2 y1, then the shift direction is left-to-right.

Marks: 5 for fully correct answer. -1 for missing the name of serial input, -1 for missing name of parallel output, -1 for missing shift direction.

[8] Q8. A digital comparator is a logic circuit that takes two **n**-bit numbers **A** and **B** as input and compares the magnitude of A and B. It has three outputs, **G, E, and L**, where G=1 only if A>B, E=1 only if A=B, and L =1 only if A<B.

A 4-bit magnitude comparator (for unsigned numbers) is to be constructed from two 2-bit comparators as shown in the diagram below. One of these will compare the high-order 2-bits ($a_3a_2$ and $b_3b_2$) of each input and the other will compare the lower 2-bits ($a_1a_0$ and $b_1b_0$). The outputs, greater ($G_{in}$), equal ($E_{in}$), and less ($L_{in}$) from the lower-bits comparator become additional inputs to the higher-bits comparator. The inputs to the lower-bits comparator are labeled $G_0$, $E_0$, and $L_0$.




(a) [2] For what values of the inputs A and B will the result of the lower-bits comparator ($G_{in}$ , $E_{in}$ and $L_{in}$) influence the overall outputs *G*, *E* and *L*? Explain. Your answer *should not* give every possible input combination – you must express your answer by stating relationships between the input values of the $a_i$, $b_i$, $G_{in}$ , $E_{in}$ and $L_{in}$.

Marking Scheme:

*Solution:* *for E =1 to happen, $E_{in}$ must be 1;* 1 mark
*If $a_3 = b_3$ and $a_2 = b_2$, $G_{in}$ or $L_{in}$ will affect G or L.* 1 mark

Q8, continued

(b) [2] If both 2-bit comparator units are to contain identical logic circuits, specify, for the lower-bits comparator, what values the inputs $G_0$, $E_0$, and $L_0$ should have in order for the complete 4-bit comparator to operate properly.

Marking Scheme:

*Solution:* $G_0 = 0$, ½ mark
$E_0 = 1$, 1 mark
$L_0 = 0$. ½ mark

(c) [3] Give a logic expression for $G$ (recall that G = 1 if A > B) in terms of $a_3$, $a_2$, $b_3$, $b_2$ and $G_{in}$. You are allowed to use the AND, OR, NOT and XOR logic operators as needed.

*Solution:* $$G = a_3\overline{b_3} + \overline{(a_3 \oplus b_3)} \bullet a_2\overline{b_2} + \overline{(a_3 \oplus b_3)} \bullet \overline{(a_2 \oplus b_2)} \bullet G_{in}$$

Marking Scheme: ½ mark for the 1st product term
1 ½ marks for the 2nd product term
2 marks for the 3rd product term

[10] Q9. A finite-state machine has one input (**w**) and one output (**z**). The input **w** is a serial input synchronized to a clock in a similar manner to the sequence detectors you built in lab #6.

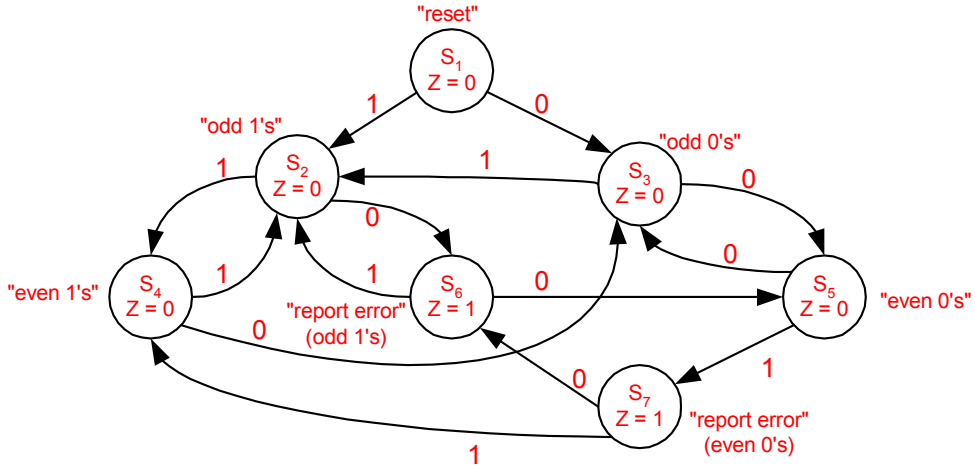
The state machine is part of a communication system that is using the following rules for transmission of data through the input w: If a *0* occurs in the input stream, then there should be an odd numbers of 0's; if a *1* occurs, then there should be an even numbers of 1's.

The purpose of the state machine is to detect errors in the sequence that doesn't obey these rules. The output, z, should be set to 1 for one clock cycle whenever an even number of *zeroes* occur in the input sequence or an odd number of *ones*. The following pattern shows the corresponding values of w and z.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **w** | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| **z** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Give a Moore-type state diagram for this FSM. Use as few states as possible:

*Solution:*




Marking Scheme: 10 marks for seven states w/ all correct state transitions
9 or 8 marks for seven states w/ some wrong state transitions (depending on how many were wrong)
7 marks for more than seven states w/ all correct state transitions
6 or 5 marks for more than seven states w/ some wrong state transitions