University of Toronto
Faculty of Applied Science and Engineering

Midterm Test
October 28, 2016

ECE253 – Digital and Computer Systems

Examiner – Prof. Stephen Brown

Print:
First Name ..... Last Name .....

Student Number Solutions .....

1. There are **6** questions and **16** pages. Do **all** questions. The duration of the test is 1.5 hours.
2. **ALL WORK IS TO BE DONE ON THESE SHEETS.** THERE IS EXTRA SPACE ON PAGE 13 FOR ANY QUESTION IF YOU NEED TO USE IT.

   THERE ARE BLANK PAGES AT THE END OF THE EXAM THAT YOU CAN TEAR OFF TO USE FOR ROUGH WORK. YOU DON'T NEED TO HAND IN THESE EXTRA PAGES.
3. Closed book. No aids are permitted.
4. No calculators are permitted.

1 [14]	
2 [6]	
3 [8]	
4 [8]	
5 [10]	
6 [4]	
Total [50]	

[14 marks] 1. Short answers:

[2 marks] (a) Perform the following number conversions. 0.5 marks each

i. $(7777)_{16}$ to octal

Answer 16'o73567

ii. $(10100011101)_2$ to hexadecimal

Answer 15'h51D

iii. $(1100001)_2$ to decimal

Answer 7'd97

iv. $(6400)_{10}$ to binary

Answer 13'd1100100000000

[1 marks] (b) Perform the following addition of octal numbers.

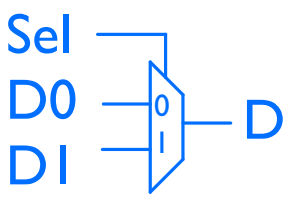
$$\begin{array}{r} 7\,2\,2\,2 \\ 7\,6\,5\,4 \\ 7\,1\,2\,3 \\ +\,\underline{7\,1\,2\,3} \end{array}$$

35344

[1 mark] (c) What circuit is produced by the following Verilog statement? Draw the circuit in the space below:

```
assign D = Sel ? D1 : D0;
```

Answer: A Multiplexer 0.5 marks if inputs swapped



[2 marks] (d) Write a Verilog always block in the space below that specifies a negative-edge-triggered 8-bit register that has an enable input *E*, data input *D*, clock input *Clock*, and data output *R*.

Answer:

```
always@(negedge Clock)
    if (E) R[7:0] <= D[7:0]
```

-0.5 if missing enable
-0.5 if using the wrong edge
-1 for not using the correct edge or other errors in sensitivity list
-0.5 for errors in R and D

[1 mark] (e) On the DE1-SoC board what would be shown on HEX0 if you made the assignment:

```
HEX0[0:6] = 7'b0010010;
```

Answer: 2 0.5 marks for "5"

[1 mark] (f) Assume that you set the SW switches on the lab board so that SW_0 is on, SW_1 is off, SW_2 is on, SW_3 is off, and so on alternating between on and off. What would the red lights LEDR look like after you made the following assignment:

```
LEDR[9:0] = SW[0:9];
```

Answer: LEDR [9:0] = 1010101010

[1 mark] (g) How many selector bits are needed for a 7-to-1 multiplexer?

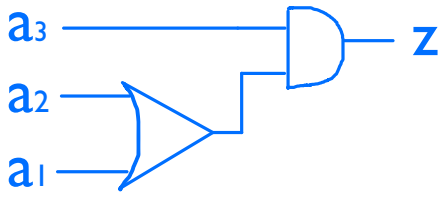
Answer: 3

[1 mark] (h) In the signal name *Resetn*, what is the conventional meaning of the letter *n*?

Answer: Specifies the reset as "active-low"

[2 marks] (i) Design and draw a circuit that has a four-bit input $A = a_3a_2a_1a_0$ and and that produces an output z. The value of z should be 1 if $A > 9$, otherwise z should be 0. Draw the circuit in the space below, using logic gates. Make your circuit as simple as possible.

Answer:



-1 for not being min-cost

[2 marks] (j) Design a circuit whose input is a BCD digit $B = b_3b_2b_1b_0$ and produces an output M = m_1m_0. The circuit should calculate $M = B \% 3$, where % is the modulo operator. Give minimum-cost sum-of-products expressions for m_1 and m_0.

Answer: Truth table on pg 13

$m_0 = b_2\bar{b}_1\bar{b}_0 + b_2b_1b_0 + \bar{b}_3\bar{b}_2\bar{b}_1b_0$

$m_1 = b_3\bar{b}_0 + b_2\bar{b}_1b_0 + \bar{b}_3b_1\bar{b}_0$

b_1b_0 \ b_3b_2	00	01	11	10
00	0	1	d	0
01	1	0	d	0
11	0	1	d	d
10	0	0	d	d

m_0

b_1b_0 \ b_3b_2	00	01	11	10
00	0	0	d	1
01	0	1	d	0
11	0	0	d	d
10	1	0	d	d

m_1

For each "m"
-0.5 for one extra or wrong term or not using "don't cares"
-1 for 2+ wrong terms

[6 marks] 2. Boolean Algebra:

[2 marks] (a) Use Boolean algebra to prove or disprove the following relation:

$\overline{x \oplus y} = \overline{x}\,\overline{y} + xy$

LS = $\overline{(x\overline{y} + \overline{x}y)}$
= $\overline{(x\overline{y})}\,\overline{(\overline{x}y)}$
= $(\overline{x} + y)(x + \overline{y})$
= $x\overline{x} + \overline{x}\overline{y} + xy + y\overline{y}$
= $\overline{x}\overline{y} + xy$
= RS

For each part:
0.5 marks for prove/disprove only
1 mark for partial proof
1.5 marks for only minor errors
2 marks for correct solution

No marks awarded for Truth Tables (must be algebra)

[2 marks] (b) Use Boolean algebra to prove or disprove:

$x(y \oplus z) = xy \oplus xz$

LS = $x(y\overline{z} + \overline{y}z)$
= $xy\overline{z} + x\overline{y}z$

RS = $\overline{(xy)}(xz) + (xy)\overline{(xz)}$
= $(\overline{x}+\overline{y})(xz) + (xy)(\overline{x}+\overline{z})$
= $\overline{x}xz + x\overline{y}z + xy\overline{x} + xy\overline{z}$
= $x\overline{y}z + xy\overline{z}$

LS = RS

[2 marks] (c) Use Boolean algebra to prove or disprove (note: $\wedge$ is the NAND operator):

$x \wedge (y + z) = x \wedge y + x \wedge z$

LS = $\overline{x(y + z)}$
= $\overline{x} + \overline{(y + z)}$
= $\overline{x} + \overline{y}\overline{z}$

RS = $\overline{xy} + \overline{xz}$
= $(\overline{x} + \overline{y}) + (\overline{x} + \overline{z})$
= $\overline{x} + \overline{y} + \overline{z}$

LS != RS

3. Karnaugh Maps:

[8 marks] Consider the function f shown in the Karnaugh map below.

x_3x_4 \ x_1x_2	00	01	11	10
00	1	0	d	1
01	0	d	d	0
11	0	d	1	1
10	1	0	1	1

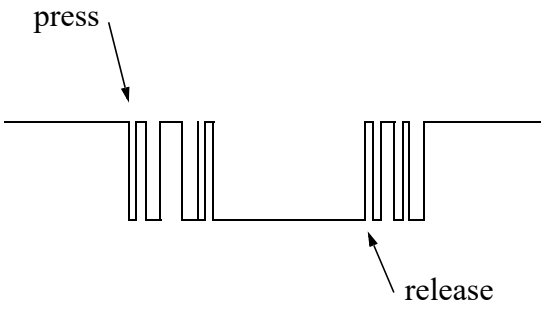
For each of the expressions below, place a ✓check mark in the box on the left if the expression represents a valid cover for f.

	$\overline{x}_2\overline{x}_1 + x_1x_3$
	$x_1x_2 + \overline{x}_2\overline{x}_4$
	$(\overline{x}_2 + x_3) \cdot (x_3 + \overline{x}_4) \cdot (\overline{x}_1 + \overline{x}_2)$
✓	$\overline{x}_1\overline{x}_2\overline{x}_4 + x_1\overline{x}_4 + x_1x_3$
✓	$(x_1 + \overline{x}_4) \cdot (x_1 + \overline{x}_2) \cdot (x_3 + \overline{x}_4)$
	$\overline{x}_2x_3 + x_2x_4 + \overline{x}_2\overline{x}_3\overline{x}_4$
	$x_2x_4 + x_1\overline{x}_3$
✓	$(x_1 + x_2 + \overline{x}_4) \cdot (x_1 + \overline{x}_2 + x_4) \cdot (\overline{x}_1 + x_2 + x_3 + \overline{x}_4)$

1 mark each (Y/N)

[8 marks] 4. Sequential circuit:

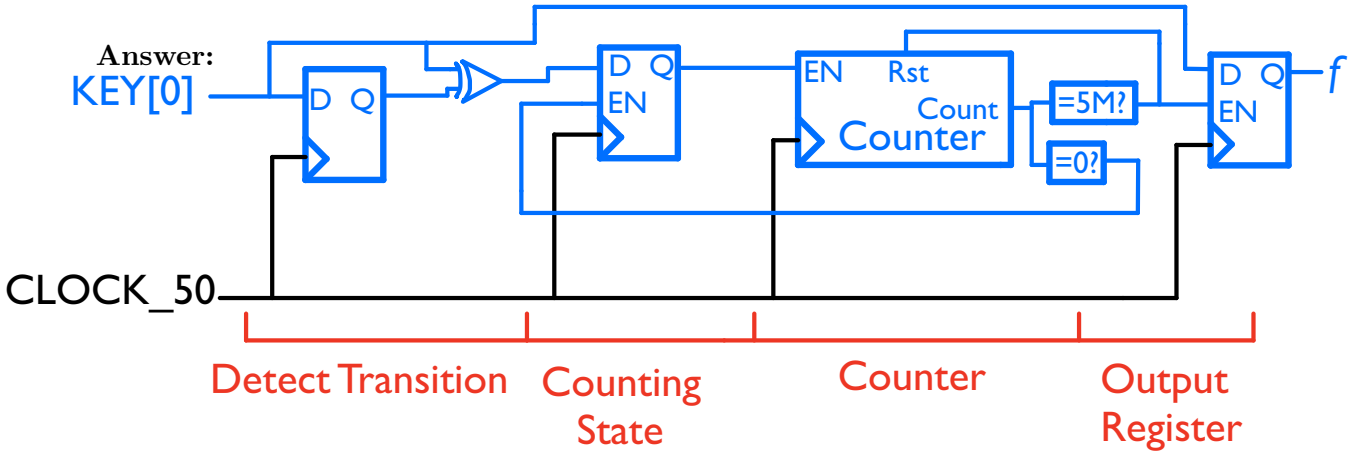
Consider a simple pushbutton switch, such as KEY_0 on the DE1-SoC board. When pressed, or released, this pushbutton *bounces* up/down for a while before settling to 0 (when pressed) or 1 (when released). You can assume that bouncing never persists for more than 100 ms. An example of switch bouncing is shown below.



On the DE1-SoC a *debouncing* circuit is used outside the FPGA chip to eliminate this problem, so that when the KEY is pressed the FPGA sees only a single change from 1 to 0 and then from 0 to 1. For this question, assume that debouncing has not been done at all, and that you have to implement a circuit in the FPGA that performs debouncing. Your circuit has the input KEY_0 and the output f, which is a debounced version of KEY_0. You will likely want to use flip-flops, or registers, or counters in your design, so assume that you have CLOCK_50 as another input to your debouncing circuit. *Hint*: a possible solution is to use a counter to wait for an appropriate amount of time for bouncing to stop.

Draw your circuit in the space below, and/or on the next page.

Answer:



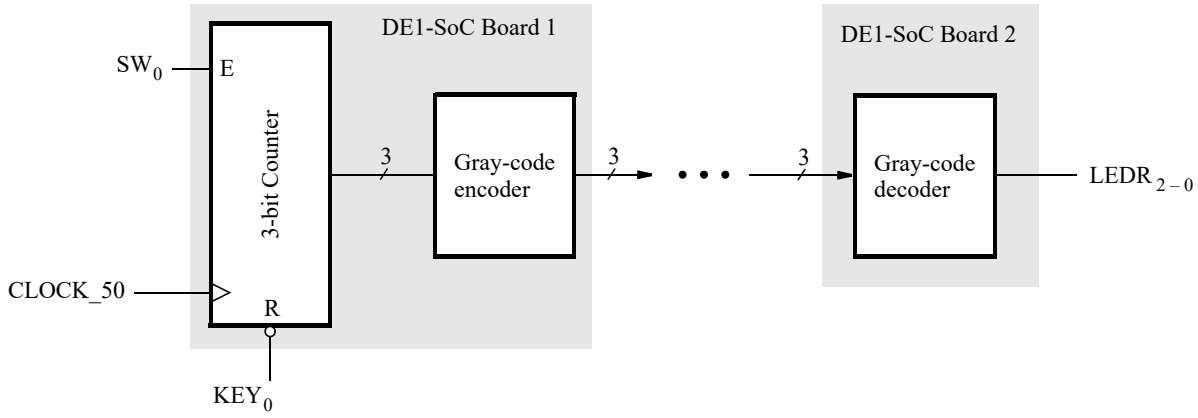
2 marks / part:
1 mark for the idea
1 mark for correctness

Verilog solutions (instead of schematics) only awarded idea marks

Answer space for Question 4 continued:

[10 marks] 5. Verilog Question:

Consider the circuits shown below, implemented in two separate DE1-SoC boards:



The circuit on the left has a three-bit counter that uses the CLOCK_50 clock signal, an enable input connected to SW_0, and a synchronous reset connected to KEY_0. The counter is required to increment every 1/4 second, in the sequence $000, 001, \ldots, 111$. The counter is connected to a *gray-code encoder*, which converts the counter output into a gray code. Recall that in a gray code only one bit changes in each clock cycle. This gray code is then transmitted using three wires, as indicated in the picture, to another DE1-SoC board where it is converted back to a sequential code and then displayed on the red lights $LEDR_{2-0}$.

Assume that you want to implement hierarchical Verilog code for the circuit in DE1-SoC Board 1, and separate Verilog code for the circuit in DE1-SoC Board 2. You can use any types of Verilog statements, including if-else, case, etc.

[4 marks] (a) Write a Verilog module for the counter subcircuit, which increments every 1/4 second. More space is provided at the top of the following page.

Answer:

```
module counter (input clk, input rstn,
input en, output reg [2:0] count);

    reg [23:0] quart;

    always@(posedge clk)
        if (!rstn) begin
            count <= 0;
            quart <= 0;
        end
```

Per counter (fast and slow):
-0.5 for not using CLOCK_50
-0.5 for too few bits
-0.5 for no enable signal
-0.5 for no resetn
-0.5 if only reset when enabled

Additional space for the counter Verilog code ...

```
        else if (en) begin
            if (quart == 12499999) begin
                quart <= 0;
                count <= count + 1;
            end
            else
                quart <= quart + 1;
        end
endmodule
```

[2 marks] (b) Write a Verilog module for the gray-code encoder subcircuit.

Answer:

```
module gray_encode (input [2:0] count,
output reg [2:0] graycount);

always@(*)
    case (count)
        3'b000: graycount = 3'b000;
        3'b001: graycount = 3'b001;
        3'b010: graycount = 3'b011;
        3'b011: graycount = 3'b010;
        3'b100: graycount = 3'b110;
        3'b101: graycount = 3'b111;
        3'b110: graycount = 3'b101;
        3'b111: graycount = 3'b100;
    endcase
```

0.5 marks for only module declaration
1 mark for 1-3 terms correct
1.5 marks for 4-7 terms correct
2 marks for 8 terms correct

[2 marks] (c) Write a top-level Verilog module for the circuit in DE1-SoC Board 1.

Answer:

```
module boardone (input CLOCK_50,
input [0:0] KEY, input [0:0] SW, output
[2:0] graycount);
    wire [2:0] count;

    counter c (CLOCK_50, KEY[0],
SW[0], count);

    gray_encode g (count, graycount);
endmodule
```

0.5 marks for inputs+outputs
0.5 marks for connecting wires
0.5 marks for each submodule instantiation

[2 marks] Write a top-level Verilog module for the circuit in DE1-SoC Board 2.

Answer:

```
module boardtwo (input [2:0] graycount,
output [2:0] LEDR);

    gray_decode (graycount, LEDR);
endmodule
```

1 mark inputs + outputs
1 mark submodule instantiation

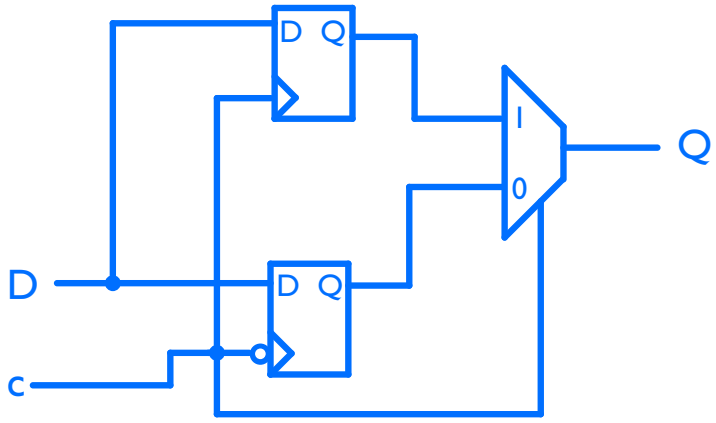
[4 marks] 6. Verilog Code Analysis:

Consider the Verilog code shown below.

```
module DDR (input c, input D, output Q);
    reg p, n;
    always @ (posedge c)
        p <= D;
    always @ (negedge c)
        n <= D;
    assign Q = c ? p : n;
endmodule
```

[3 marks] (a) Draw a circuit that corresponds to this code.

Answer:



0.5 marks for each register
1 mark for 2:1 mux & c select
1 mark for correct reg-to-mux connections

[1 mark] (b) Explain briefly what this circuit "does"?

Answer:

Dual-edge triggered Flip Flop
or
D goes to Q at both clock edges

Extra answer space for any question on the test, if needed:

b_3	b_2	b_1	b_0	m_1	m_0	b_3	b_2	b_1	b_0	m_1	m_0
0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	1	1	0	0	1	0	0
0	0	1	0	1	0	1	0	1	0	d	d
0	0	1	1	0	0	1	0	1	1	d	d
0	1	0	0	0	1	1	1	0	0	d	d
0	1	0	1	1	0	1	1	0	1	d	d
0	1	1	0	0	0	1	1	1	0	d	d
0	1	1	1	0	1	1	1	1	1	d	d

EXTRA PAGE: TEAR THIS PAGE OFF TO USE FOR ROUGH WORK. YOU DO NOT NEED TO HAND IN THIS SHEET.

EXTRA PAGE: TEAR THIS PAGE OFF TO USE FOR ROUGH WORK. YOU DO NOT NEED TO HAND IN THIS SHEET.

EXTRA PAGE: TEAR THIS PAGE OFF TO USE FOR ROUGH WORK. YOU DO NOT NEED TO HAND IN THIS SHEET.