University of Toronto
Faculty of Applied Science and Engineering

Midterm Test
October 30, 2014

ECE253 – Digital and Computer Systems

Examiner – Prof. Stephen Brown

Print:

First Name . . . . . Last Name . . . . . .

Student Number . . . . . .

1. There are **6** questions and **14** 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 TWO 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 [7]	
2 [8]	
3 [8]	
4 [10]	
5 [9]	
6 [13]	
Total [55]	

[7 marks] 1. Number bases:

(a) Convert the following signed decimal numbers to 2's complement. Use the minimum number of bits necessary.

i. -511

Answer 1000000001

ii. -112

Answer 10010000

iii. 3200

Answer 0110010000000

(b) Give the equivalent signed decimal value of the following 2's complement binary numbers.

i. 111000111

Answer -57

ii. 10000

Answer -16

iii. 1

Answer -1

(c) Convert the following decimal number to base 7.

i. 126

Answer 240

[8 marks] 2. Boolean Algebra:

[3 marks] (a) You want to impress your friends with your knowledge of Boolean algebra. So, when they make the following statements, your response is to specify a Boolean identity they should have used to simplify their statement. For example, your response could be "12a". (The numbers of Boolean identities are listed at the end of this exam paper). Provide your responses to your friend's statements below:

Identity	
13a	This party would be more fun if Bob was here or if both Bob and Lisa were here.
16a	This party would be more fun if Jim was here or if Al was here and Jim was not here.
17a	This party would be more fun if Bob and Al were here, or if Al and Jim were here, or Jim was here and Bob was not here.

[3 marks] (b) Use Boolean algebra to derive a minimal sum-of-products expression for the function f, below. You do not need to specify which Boolean identities or rules you use, but perform as few steps as possible.

$$f = x_2x_4 + x_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2x_3x_4$$

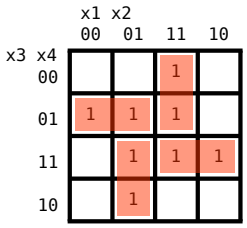
Answer:

$= x_2x_4x_1\overline{x}_3 + x_1x_2\overline{x}_3\overline{x}_4$
$+ x_2x_4\overline{x}_1\overline{x}_3 + \overline{x}_1x_2\overline{x}_3x_4$
$+ x_2x_4\overline{x}_1x_3 + \overline{x}_1x_2x_3\overline{x}_4$
$+ x_2x_4x_1x_3 + x_1\overline{x}_2x_3x_4$

expand x_2x_4 into four minterms

$= x_1x_2\overline{x}_3$
$+ \overline{x}_1 \overline{x}_3x_4$
$+ \overline{x}_1x_2x_3$
$+ x_1 x_3x_4$

Combining



[2 marks] (c) Consider the three-input functions $f = \overline{x}_2\overline{x}_3 + \overline{x}_1x_2 + x_1x_3$ and $g = \overline{x}_1\overline{x}_3 + x_1x_2 + x_1x_3$. In the space below, use Boolean algebra to prove that $f \neq g$. You are not allowed to use any tools other than Boolean algebra in your solution. Make your answer as concise as possible.

Answer:

$f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2\overline{x}_3 + \overline{x}_1x_2x_3 + x_1\overline{x}_2x_3 + x_1x_2x_3$

$= \Sigma\, m(0, 4, 2, 3, 5, 7)$

$g = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2\overline{x}_3 + x_1x_2\overline{x}_3 + x_1x_2x_3 + x_1\overline{x}_2x_3 + x_1x_2x_3$

$= \Sigma\, m(0, 2, 6, 7, 5, ~~7~~)$

One function contains at least one minterm not in the other one, so they're not equal.

f

x3 \ x1 x2	00	01	11	10
0	1	1		1
1		1	1	1

g

x3 \ x1 x2	00	01	11	10
0	1	1	1	
1			1	1

[8 marks] 3. Karnaugh Maps:

Consider the function f shown in the Karnaugh map below.

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

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. You may wish to use the blank space on the opposite page for rough work.

		$\overline{x}_2\overline{x}_4 + \overline{x}_1x_3$
✓		$(x_2 + x_3) \cdot (\overline{x}_1 + \overline{x}_3) \cdot (x_1 + x_3)$
		$(\overline{x}_2 + x_3) \cdot (x_3 + \overline{x}_4) \cdot (\overline{x}_1 + \overline{x}_2)$
✓		$\overline{x}_1x_3 + x_1x_2\overline{x}_3$
		$x_3x_4 + \overline{x}_2\overline{x}_4 + \overline{x}_1x_2x_3\overline{x}_4$
✓		$x_3x_4 + \overline{x}_1x_3\overline{x}_4 + x_1x_2\overline{x}_3x_4$
		$x_2x_4 + x_1\overline{x}_3$
✓		$(\overline{x}_1 + \overline{x}_2 + \overline{x}_3 + x_4) \cdot (x_3 + x_4) \cdot (x_2 + x_3)$

[10 marks] 4. Verilog question:

Match each of the 10 Verilog code snippet shown below with an equivalent circuit on the next page (A-L). The answer for the first code snippet is given (A).

Note that input and output names in the Verilog code may be different from those shown in the circuits. Each circuit may be used more than once, and logic simplification may be required.

4.1
```
always @(a, b, en)
  if (en)
    out = a | (~b & b);
```
Answer: A

4.2
```
assign z = (~x) & (~y);
```
Answer: E

4.3
```
assign out = a & b & c;
```
Answer: L

4.4
```
wire [4:0] out;
assign out = a[3:0] + b[3:0];
```
Answer: J

4.5
```
always @(D)
  Q = D
```
Answer: H

4.6
```
always @(posedge R, posedge clk)
  if (R)
    Q <= 0;
  else
    Q <= D;
```
Answer: B

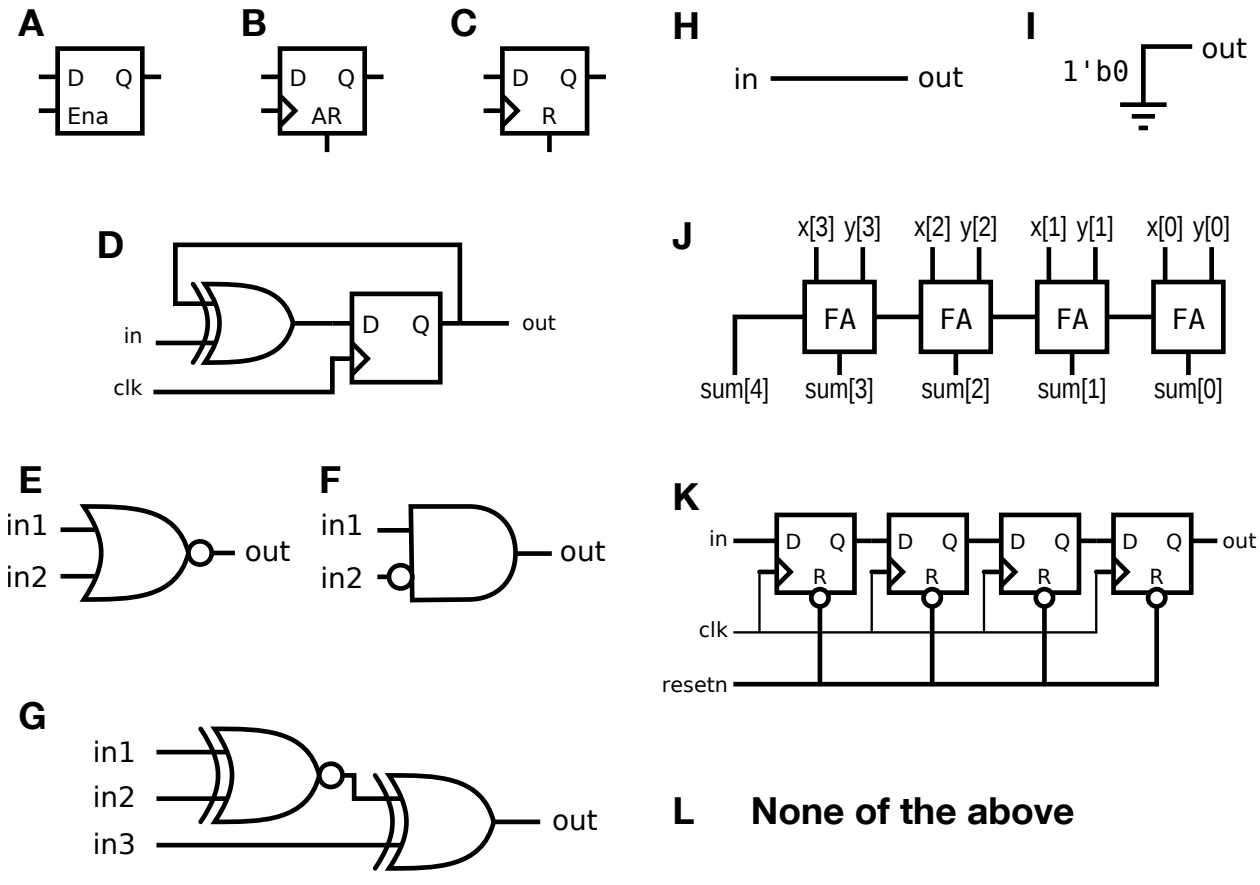
4.7
```
always @(D, R)
  if (R)
    Q = 0;
  else
    Q = D;
```
Answer: F

4.8
```
module code (input a, output z);
  reg b, c;
  assign z = ~b ^ c;
  always @(a)
    case (a)
      0: {b,c} = {1'h1, a};
      1: {b,c} = {a, 1'd0};
    endcase
endmodule
```
Answer: I

4.9
```
module M1 (input a);  // No outputs
  assign x = a;
endmodule

module top (input a, input b, output x);
  assign x = a & !b;
  M1 u(a);
endmodule
```
Answer: F

4.10
```
module code(input clk, input s, output out);
  reg state, next_state;
  parameter A=1'b0, B=1'b1;

  always@(state, s)
    case (state)
      A: if (s) next_state = B;
         else   next_state = A;
      B: if (s) next_state = A;
         else   next_state = B;
    endcase

  always @(posedge clk)
    state <= next_state;

  assign out = (state == B);
endmodule
```

Answer: D



[9 marks] 5. Verilog Question:

Three versions of a Verilog module are shown below. Each of these modules is intended to create two counters, $c1$ and $c2$, and comparators $r1$ and $r2$. The output o is meant to indicate when the counter $c2$ is finished counting. There are some differences in each of the modules shown, including errors in the code. Write the letter A-H of the timing diagram that matches the behavior of each module. The timing diagrams, which are on the following page, each show several cycles for the counters.

```
module m (input clk,
  output reg [1:0] c1,
  output reg [2:0] c2,
  output reg o
);
  wire r1 = (c1 == 2);
  wire r2 = (c2 == 5);
  always @(posedge clk, posedge r1)
    if (r1)
      c1 <= 0;
    else
      c1 <= c1 + 1;

  always @(posedge clk)
  begin
    if (r1 & r2) begin
      c2 <= 0;
      o <= 1;
    end
    else begin
      o <= 0;
      if (r1)
        c2 <= c2 + 1;
    end
  end

endmodule
```

Matches E

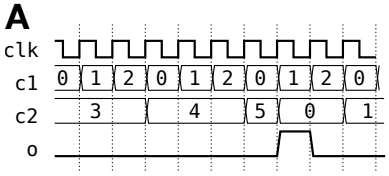
```
module m (input clk,
  output reg [1:0] c1,
  output reg [2:0] c2,
  output reg o
);
  wire r1 = (c1 == 2);
  wire r2 = (c2 == 5);
  always @(posedge clk)
    if (r1)
      c1 <= 0;
    else
      c1 <= c1 + 1;

  always @(posedge clk)
  begin
    if (r2) begin
      c2 <= 0;
      o <= 1;
    end
    else begin
      o <= 0;
      if (r1)
        c2 <= c2 + 1;
    end
  end

endmodule
```

Matches A

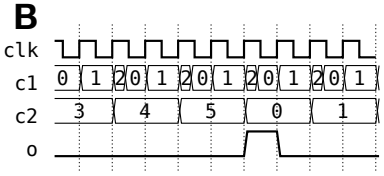
```
module m (input clk,
  output reg [1:0] c1,
  output reg [2:0] c2,
  output reg o
);
  wire r1 = (c1 == 2);
  wire r2 = (c2 == 5);
  always @(posedge clk)
    if (r1)
      c1 <= 0;
    else
      c1 <= c1 + 1;

  always @(posedge clk)
  begin
    if (r1 & r2) begin
      c2 <= 0;
      o <= 1;
    end
    else begin
      o <= 0;
      if (r1)
        c2 <= c2 + 1;
    end
  end

endmodule
```

Matches C

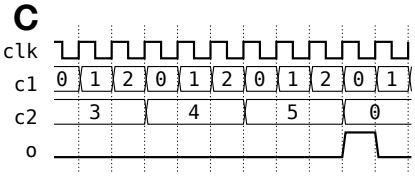
A



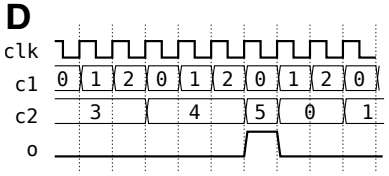
B



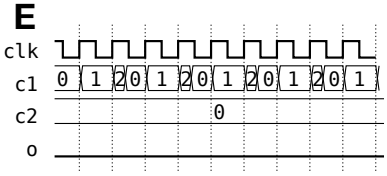
C



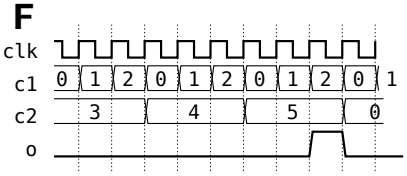
D



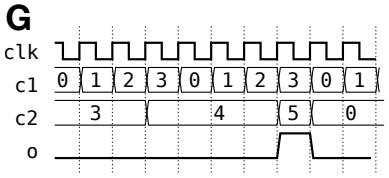
E



F



G



H

None of the above

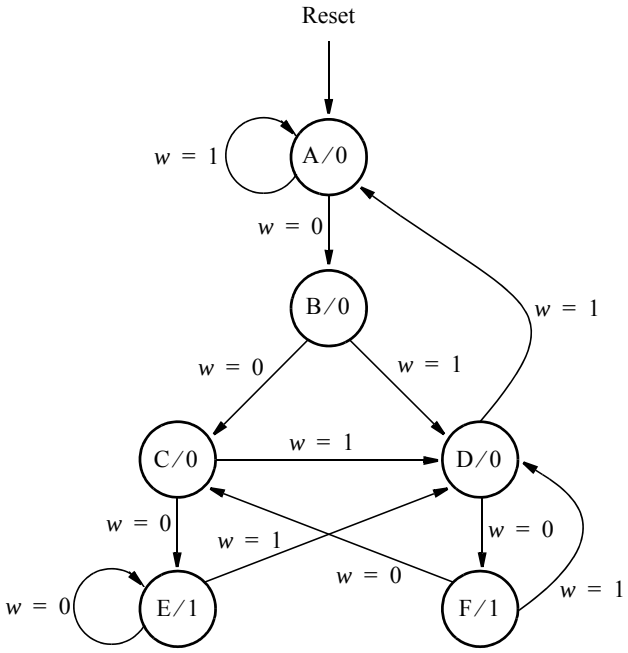
[13 marks] 6. Finite State Machine Question:

(a) Consider the state diagram shown below, which has the input w and output z. Answer this question as concisely as possible:

"For what values of w does this FSM produce an output $z = 1$?"

Answer:

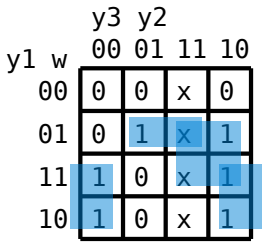
When a sequence of "010" or "000" is encountered.



(b) Assume that you wish to implement the FSM using three flip-flops and state codes $y_3y_2y_1 = 000, 001, \ldots, 101$ for states A, B, ..., F, respectively. Show a state-assigned table for this FSM. Derive a minimal sum-of-products next-state expression for the flip-flop y_2. Use a K-map and show your work. You do not need to derive any other expressions. Put your answer on the following page.

Answer for Question 6 Part (*b*):

State	Current state	Next state w=0	Next state w=1	Output
A	000	001	000	0
B	001	010	011	0
C	010	100	011	0
D	011	101	000	0
E	100	100	011	1
F	101	010	011	1



$Y2 = \overline{y_2}y_1 + y_3w + y_2\overline{y_1}w$

(c) For this part assume that a one-hot code is used with the state assignment $y_6y_5y_4y_3y_2y_1 = 000001, 000010, 000100, 001000, 010000, 100000$ for states A, B, ..., F, respectively.

i. Write a logic expression for the next-state signal Y_2.

Answer $Y_2 = y_1\overline{w}$

ii. Write a logic expression for the next-state signal Y_4.

Answer $Y_4 = w\ (y_2 + y_3 + y_5 + y_6)$ or $Y_4 = w(\overline{y}_1\overline{y}_4)$

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

Boolean Identities

12*a*.	$x \cdot (y + z) = x \cdot y + x \cdot z$	*Distributive*
13*a*.	$x + x \cdot y = x$	*Absorption*
14*a*.	$x \cdot y + x \cdot \overline{y} = x$	*Combining*
15*a*.	$\overline{x \cdot y} = \overline{x} + \overline{y}$	*DeMorgan's theorem*
16*a*.	$x + \overline{x} \cdot y = x + y$	
17*a*.	$x \cdot y + y \cdot z + \overline{x} \cdot z = x \cdot y + \overline{x} \cdot z$	*Consensus*