University of Toronto
Faculty of Applied Science and Engineering

Midterm Test, October 16, 2025

ECE241 – Digital Systems

Examiners – Profs. Jason Anderson, Stephen Brown, and Shuran Wang

SOLUTIONS

1. There are **9** questions, worth a total of **80** marks, and 24 pages. Do **all** questions.
2. **ALL WORK IS TO BE DONE ON THESE SHEETS.** You can use the blank pages included on Pages 20 to 23 if you need more space for any question. Be sure to indicate clearly if your work continues elsewhere. An **Aid Sheet** is included on Page 24.
3. The test is **Closed book**. No calculators are permitted.
4. The duration of the test is **2 hours**.

[10 marks] 1. Number Representation

[2 marks] (a) Convert the following decimal numbers to 12-bit binary:

Answer:

i) 197 = 0000 1100 0101

ii) 2049 = 1000 0000 0001

[2 marks] (b) Convert the following binary to hexadecimal

Answer:

i) 101100101100 = B2C

ii) 010110010111 = 597

[2 marks] (c) Convert the following hexadecimal to 12-bit binary

Answer:

i) 241 = 001001000001

ii) ECF = 111011001111

[2 marks] (d) Convert the following binary to decimal

Answer:

i) 00101101 = 45

ii) 01000001 = 65

[2 marks] (e) What is the minimum number of bits needed to represent the result from the addition of one 8-bit number and four 6-bit numbers?

Answer:

11111111
+ 111111
+ 111111
+ 111111
+ 111111
111111011
9 bits

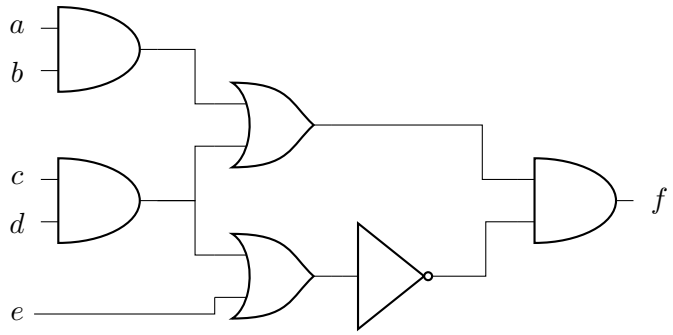
[6 marks] 2. Boolean Algebra

[2 marks] (a) Minimize the following expression using Boolean algebra. The minimized result should be in the sum-of-products (SOP) form. Show all your work and label the rules/identities used. There is an Aid Sheet at the end of this exam.

$$ab + a\bar{b}c + \bar{a}bc + \bar{a}\bar{b}c$$

Answer:
$f = a(b + \bar{b}c) + \bar{a}c(b + \bar{b})$ - distributive
$f = a(b + c) + \bar{a}c \cdot 1$ - absorption + #8
$f = ab + ac + \bar{a}c$ - distributive + #6
$f = ab + c(a + \bar{a})$ - distributive
$f = ab + c$ - #8

[4 marks] (b) Derive the minimal sum-of-products (SOP) expression for the logic circuit shown below using Boolean algebra. Show all steps and label rules/identities used.



Answer:
$f = (ab + cd)(\overline{cd + e})$
$f = (ab + cd)(\overline{cd} \cdot \bar{e})$ - DeMorgan's
$f = (ab + cd)((\bar{c} + \bar{d})\bar{e})$ - DeMorgan's
$f = (ab + cd)(\bar{c} + \bar{d})\bar{e}$ - associative
$f = (ab\bar{c} + ab\bar{d} + cd\bar{d} + cd\bar{c})\bar{e}$ - distributive
$f = (ab\bar{c} + ab\bar{d})\bar{e})$ - #8 + #6
$f = ab\bar{c}\bar{e} + ab\bar{d}\bar{e}$ - distributive

[6 marks] 3. Karnaugh Map

[4 marks] (a) Given the following logic function f, find the minimal sum-of-products (SOP) form using a Karnaugh Map (K-Map). Fill in the provided K-Map. Be sure to mind the ordering of variables.

$$f(x_3, x_2, x_1, x_0) = \sum m(0, 1, 4, 6, 7, 9, 12, 14, 15)$$

x_1x_0 \ x_3x_2	00	01	11	10
00	1	1	1	0
01	1	0	0	1
11	0	1	1	0
10	0	1	1	0

Answer:

$f = x_2x_1 + x_2\bar{x_0} + \bar{x_3}\bar{x_2}\bar{x_1} + \bar{x_2}\bar{x_1}x_0$

[2 marks] (b) Given the following K-Map, find the minimal product-of-sums (POS) form of the logic function $f(x_2, x_1, x_0)$. Note: you are required to provide POS form (not SOP form).

x_0 \ x_2x_1	00	01	11	10
0	0	1	d	0
1	d	1	0	1

Answer:

$f = (\overline{x_2} + \overline{x_1})(x_1 + x_0)$

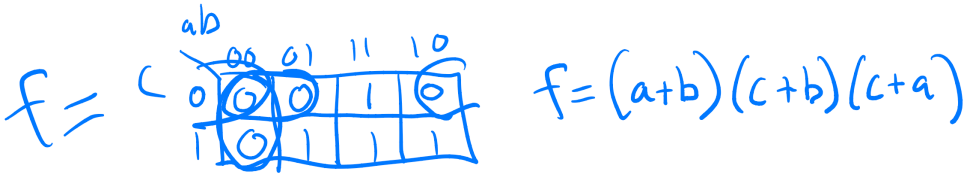
[8 marks] 4. Product-of-Sums / NOR Gates

The following truth table is for the majority function. a, b, c are inputs; f is the output. This is the function for C_{out} in a full adder.

c	a	b	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

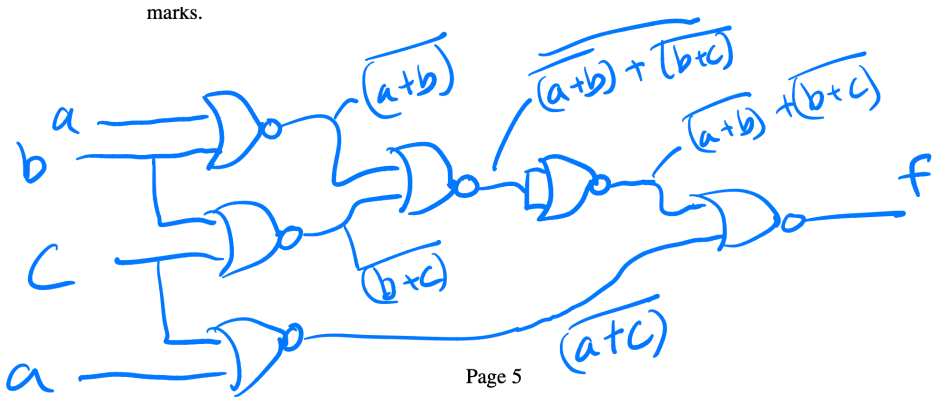
[4 marks] (a) Show the function in minimized product-of-sums (POS) form. Use any approach to minimize the function.

Answer:



[4 marks] (b) Provide a gate-level schematic for the function using only 2-input NOR gates. Label each wire in your schematic with its corresponding logic function. Unclear schematics will received reduced marks.

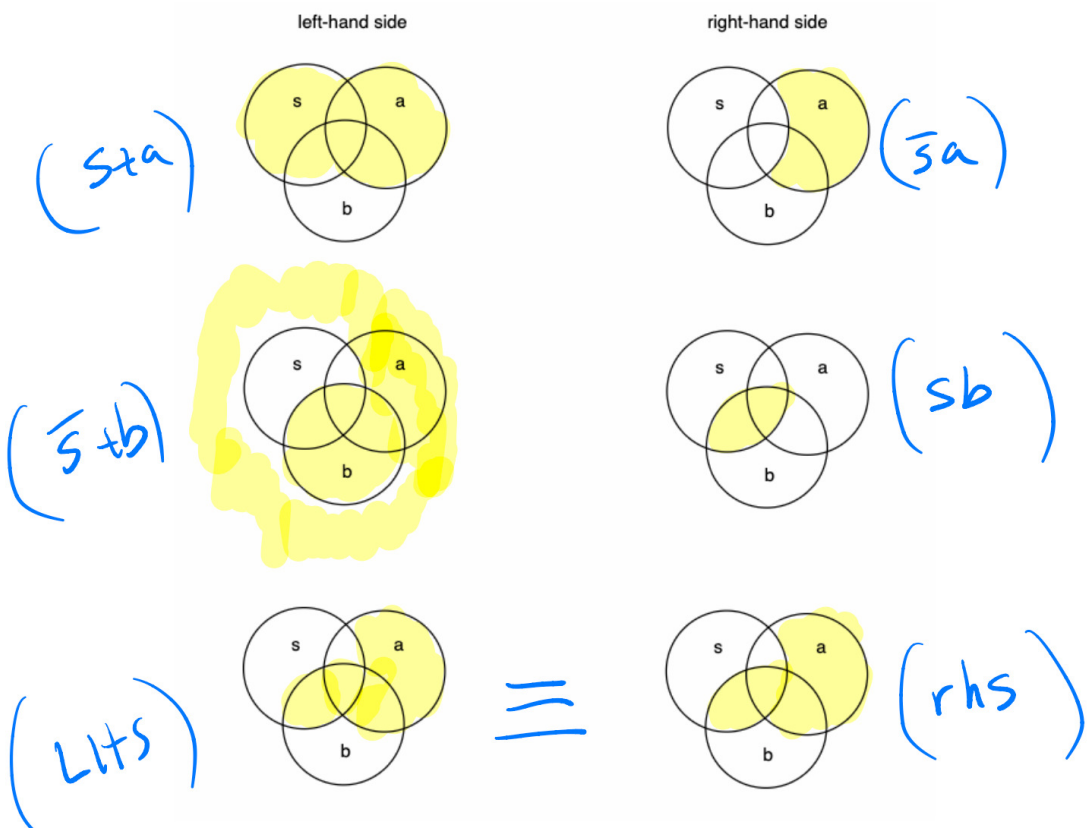
Answer:



[6 marks] 5. Venn Diagram Proof

Use Venn diagrams to prove or disprove that $(s + a)(\overline{s} + b) = \overline{s}a + sb$. Show your work using the figures below for full marks.

Answer:

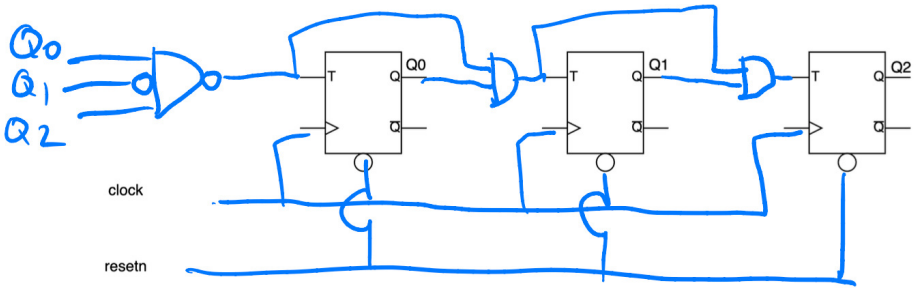


[5 marks] 6. Counter Design Question

Consider the following partial circuit with three T flip-flops. You must complete the circuit by adding additional logic gates and wires so that it implements an up-counter that counts from 0 to 5, i.e. $Q_2Q_1Q_0 = 000, \ldots, 101$, and then stops counting (remains at $Q_2Q_1Q_0 = 101$). The *resetn* input is an active-low asynchronous reset that causes $Q_2Q_1Q_0$ to reset to 000.

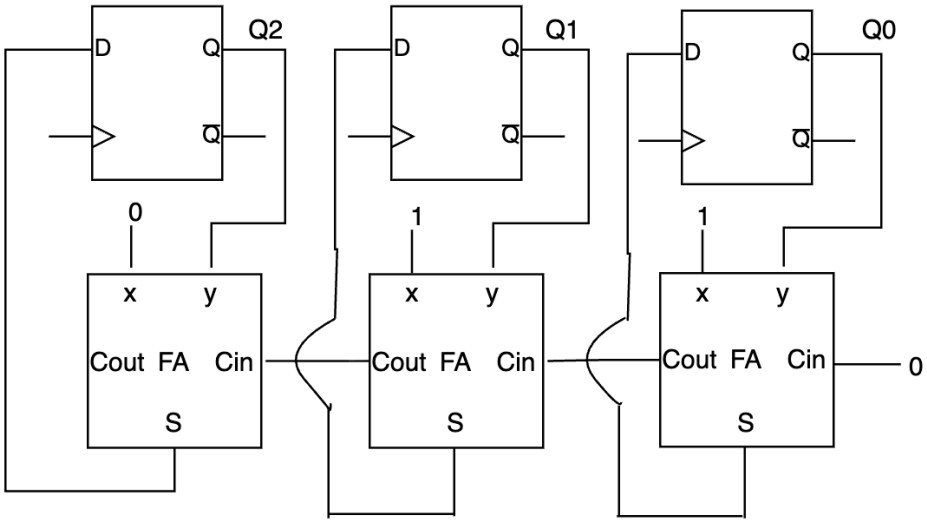
You may use any logic gates you have learned about in this course. Unclear schematics may receive reduced marks.

Answer:



[12 marks] 7. Storage Elements

[5 marks] (a) Consider the following circuit with three D-type flip-flops and three full adders (FAs). Assuming that the flip-flop outputs are initially all zero. Give the values of $Q2, Q1, Q0$ for the next 8 clock cycles. Just use text for your answer, as in $000, \ldots,$



Answer:

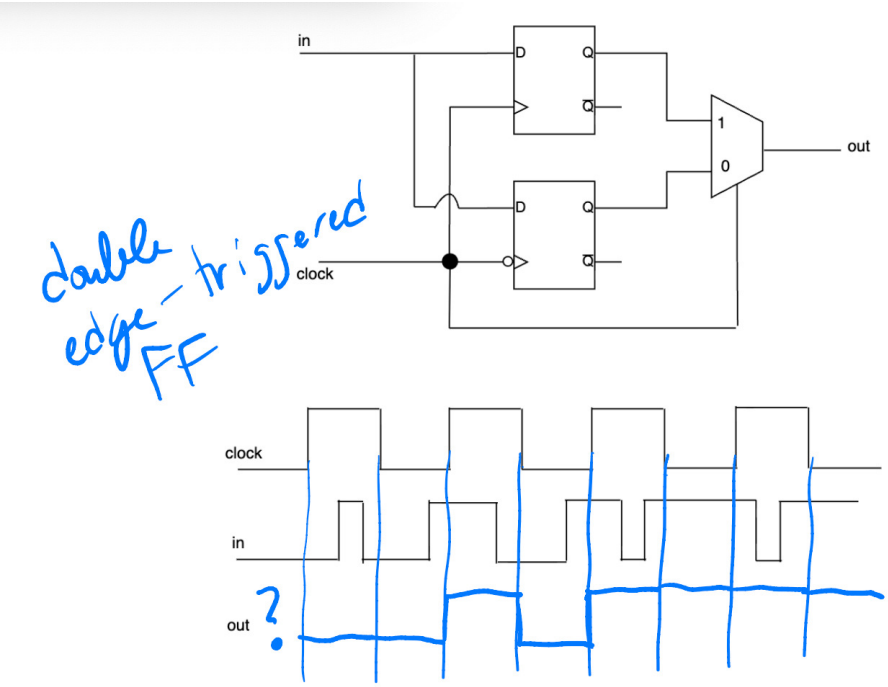
000, 011, 110, 001, 100, 111, 010, 101, 000, ...

[1 mark] (b) In one sentence, describe the output sequence produced by the above circuit.

Answer: Counter that counts up by 3 every clock cycle.

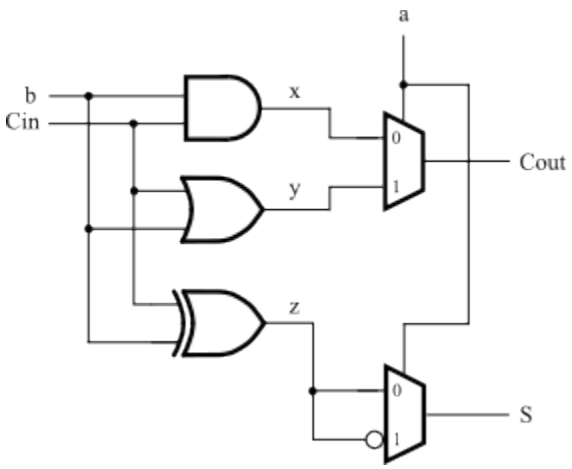
[6 marks] (c) Complete the timing diagram for the circuit below. Assume the values stored in both flip-flops are unknown before the first rising clock edge.

Answer:



[12 marks] 8. Adder question

(a) The figure below gives a circuit that implements a full adder using logic gates and multiplexers.



[3 marks]

i. In the space provided on the following page complete Verilog code for the full adder that corresponds to this circuit structure. The module and signal definitions are given below as a starting point. Use **if-else** statements to specify the multiplexers in your code.

. . . put your answer on the next page.

Answer:

Answer:

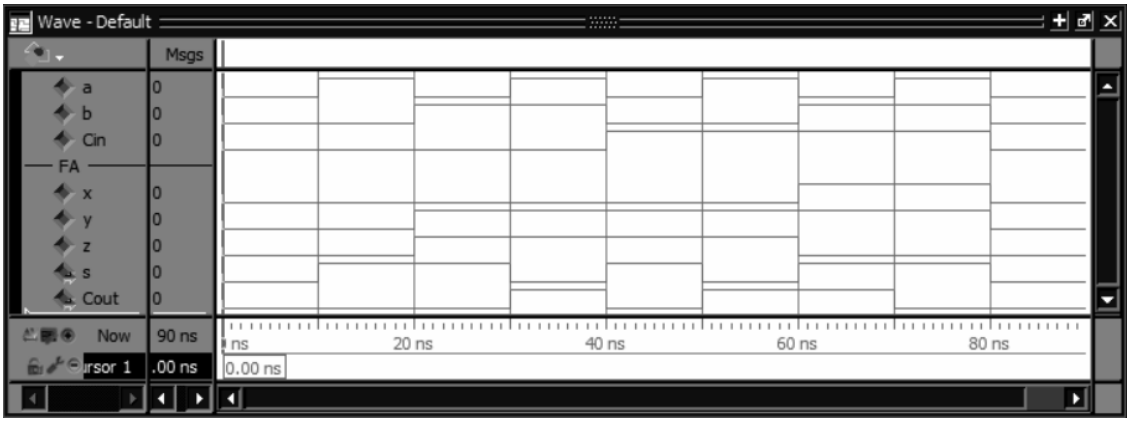
```
module FA (a, b, Cin, s, Cout);
    input a, b, Cin;
    output reg s, Cout;

    wire x, y, z;
    assign x = b & Cin;
    assign y = b | Cin;
    assign z = b ^ Cin;

    always @(*)
        if (!a) begin
            Cout = x;
            s = y;       // wrong, should be z
        end
        else begin
            Cout = z;    // wrong: should be y
            s = ~z;
        end
endmodule
```

[3 marks]

ii. If you performed a ModelSim simulation of your Verilog code, it would work perfectly (for sure!). But assume now that this question is part of a lab exercise and your lab partner, who is not as prone to perfection as yourself, has simulated their Verilog code using ModelSim and shared the simulation results with you. Your task is to determine if your partner's Verilog code produces a correct result in all cases. Those simulation results, for your lab partner's Verilog code, are given below.



Answer the following two questions.

Does your partner's simulation result show a correct implementation of the full adder circuit for all input valuations? If not, which input valuations are incorrect?

Answer:

No. The simulation result is incorrect when b and Cin are both 1.

Being as specific as possible, what do you think is wrong in your partner's Verilog code?

Answer:

There is some connection of the OR and XOR gates for b and Cin that is incorrect, because the result is wrong only when these signals are both 1.

(b) In Lab 2, you implemented a circuit that could add two BCD digits and display the two-digit result on the 7-segment displays HEX1 and HEX0.

For this question you are to solve a similar problem, but for a circuit that can add two BCO digits, where BCO means *Binary Code Octal*. Thus, your circuit produces the result:

$S = X + Y + C_{in}$,

where S is a two-digit base-8 result, X and Y are base-8 numbers and C_{in} is a carry-in. The maximum size of the result is $S = 7 + 7 + 1 = (17)_8$.

[2 marks]

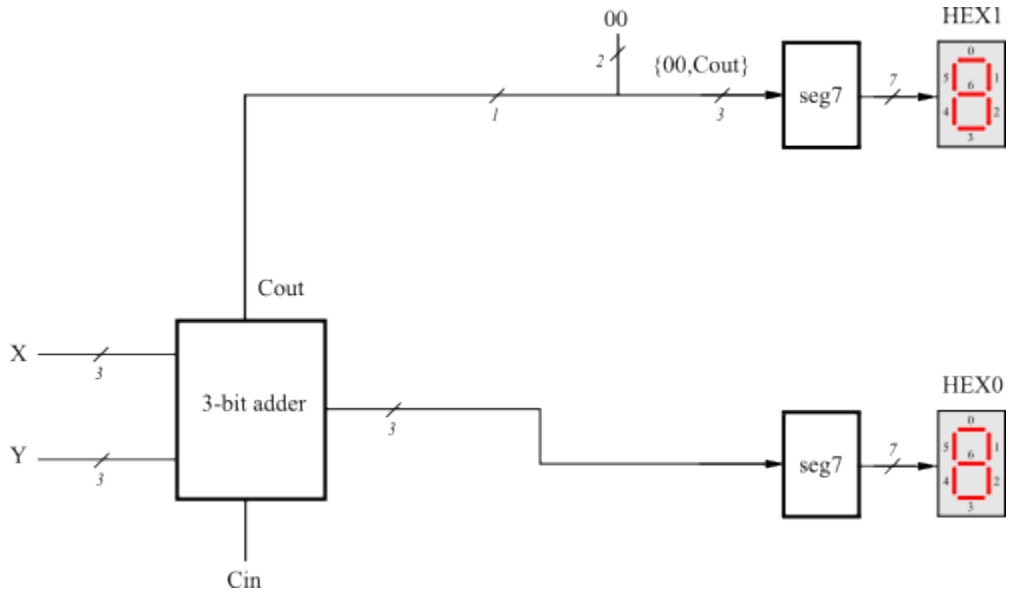
i. Fill in the table below to show what digits should be shown on HEX1 and HEX0 for each possible result S.
Answer:

Result	**HEX1**	**HEX0**
0000	0	0
0001	0	1
0010	0	2
0011	0	3
0100	0	4
0101	0	5
0110	0	6
0111	0	7
1000	1	0
1001	1	1
1010	1	2
1011	1	3
1100	1	4
1101	1	5
1110	1	6
1111	1	7

[4 marks]

ii. In the space below complete the design of your circuit that produces the 2-digit octal sum $S = X + Y + C_{in}$. You can use any needed logic gates or multiplexers, an n-bit adder module (you do not need to show the design of the adder; just draw it as a symbol with inputs and outputs). You do not need to design the 7-segment decoder that is shown in the partial circuit diagram below. In your solution, draw all lines and symbols clearly, and label all key features; marks will be deducted for unclear drawings.

Answer:



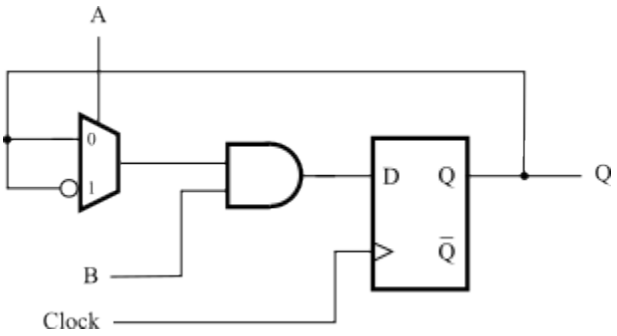
[15 marks] 9. Verilog question

[2 marks] (a) Consider the Verilog module given below. Complete the logic circuit below so that it correctly implements this Verilog code. You should make use of 2-to-1 multiplexers where appropriate, and can also include other logic gates as needed. Marks will deducted for messy circuits, so keep your circuits as simple as possible and draw carefully.

```
module TFF (A, B, C, Q);
    input A, B, C;
    output reg Q;

    always @(posedge C)
        if (B == 0)
            Q <= 1'b0;
        else if (A == 1)
            Q <= ~Q;
endmodule
```

Answer:

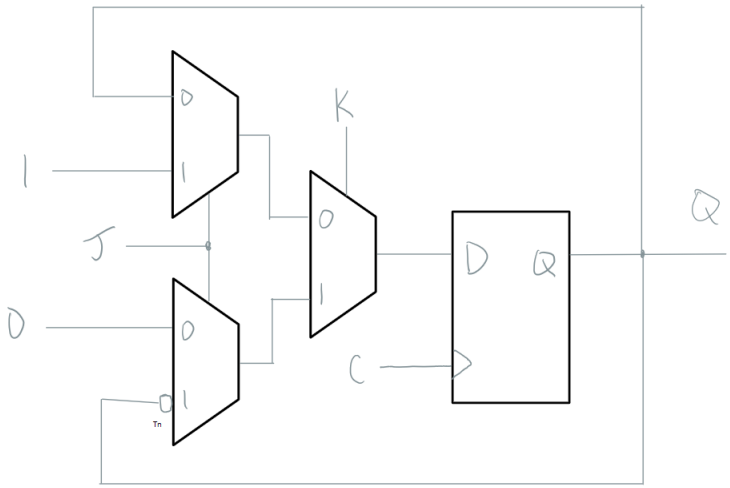


[4 marks] (b) Consider the Verilog module given below. Complete the logic circuit below so that it correctly implements this Verilog code. You should make use of 2-to-1 multiplexers where appropriate, and can also include other logic gates as needed. Marks will deducted for messy circuits, so keep your circuits as simple as possible and draw carefully.

```
module JKFF (J, K, C, Q);
    input J, K, C;
    output reg Q;

    always @(posedge C)
        if (K)
            if (J)
                Q <= ~Q;
            else
                Q <= 1'b0;
        else
            if (J)
                Q <= 1'b1;
endmodule
```

Answer:

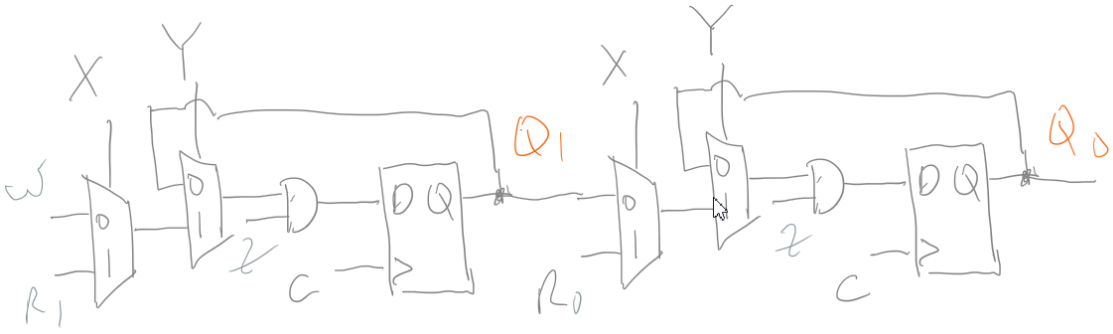


[4 marks] (c) Consider the Verilog module given below. Complete the logic circuit below so that it correctly implements this Verilog code. You should make use of 2-to-1 multiplexers where appropriate, and can also include other logic gates as needed. Marks will deducted for messy circuits, so keep your circuits as simple as possible and draw carefully.

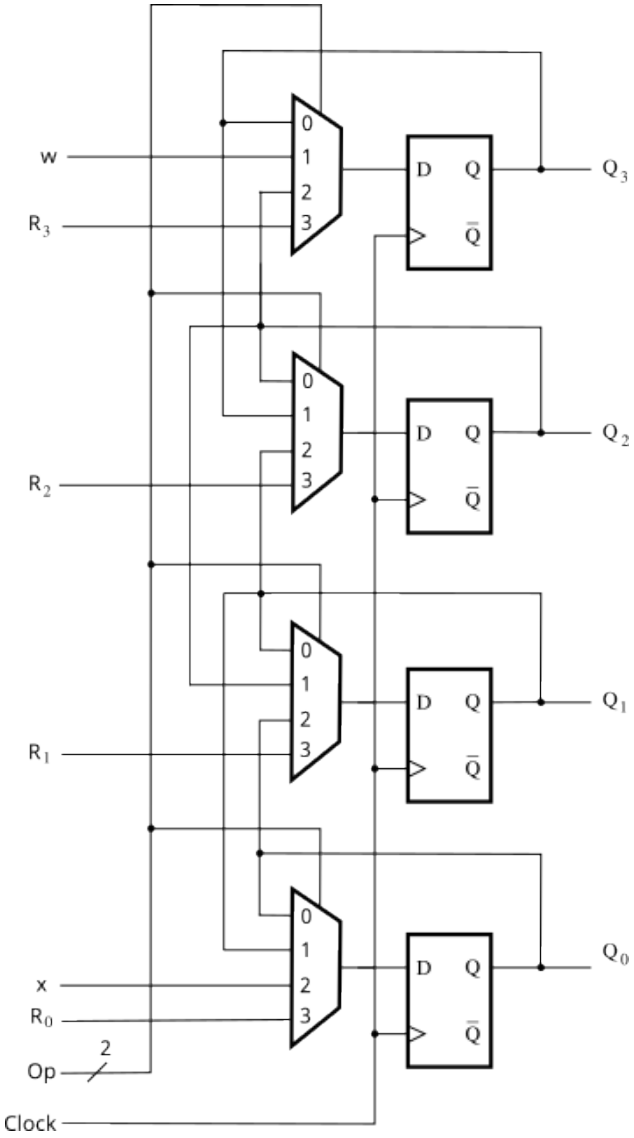
```
module SR (W, X, Y, Z, R, C, Q);
    input W, X, Y, Z, C;
    input [1:0] R;
    output reg [1:0] Q;

    always @(posedge C)
        if (!Z)
            Q <= 2'b0;
        else if (Y)
            if (X)
                Q <= R;
            else
                Q <= {W,Q[1]};
endmodule
```

Answer:



(d) Consider the logic circuit shown below. In this circuit any two wires that cross each other are connected together only if there is a **dot** at their intersection.



Answer the questions about this circuit on the following page.

[2 marks]

i. Discuss briefly what this circuit "does", by describing what happens in the circuit during each clock cycle for the different possible valuations of the 2-bit *Op* input.

Answer:

This circuit is a type of shift register. If Op == 00, the shift register contents do not change. If Op == 01, the circuit shift in the top-to-bottom direction. If Op == 10, the circuit shifts in the bottom-to-top direction. If Op == 11 the FFs are loaded in parallel from the R input.

[3 marks]

ii. In the space below write complete Verilog code that can be used to specify this circuit. The module definition is provided for you as a starting point.

Answer:

```
module Univ (w, x, R, Op, Clock, Q);
    input w, x, Clock;
    input [1:0] Op;
    input [3:0] R;
    output reg [3:0] Q;

    always @(posedge Clock)
        if (Op == 2'b01)
            Q <= {w,Q[3:1]};
        else if (Op == 2'b10)
            Q <= {Q[2:0],x};
        else if (Op == 2'b11)
            Q <= R;
endmodule
```

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

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

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

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

Aid Sheet

Boolean Algebra Axioms

1*a*.	$0 \cdot 0 = 0$	1*b*.	$1 + 1 = 1$
2*a*.	$1 \cdot 1 = 1$	2*b*.	$0 + 0 = 0$
3*a*.	$0 \cdot 1 = 1 \cdot 0 = 0$	3*b*.	$1 + 0 = 0 + 1 = 1$
4*a*.	If $x = 0$, then $\overline{x} = 1$	4*b*.	If $x = 1$, then $\overline{x} = 0$

Boolean Algebra Rules

5*a*.	$x \cdot 0 = 0$	5*b*.	$x + 1 = 1$
6*a*.	$x \cdot 1 = x$	6*b*.	$x + 0 = x$
7*a*.	$x \cdot x = x$	7*b*.	$x + x = x$
8*a*.	$x \cdot \overline{x} = 0$	8*b*.	$x + \overline{x} = 1$
9.	$\overline{\overline{x}} = x$		

Boolean Algebra Identities

10*a*.	$x \cdot y = y \cdot x$	10*b*.	$x + y = y + x$
11*a*.	$x \cdot (\, y \cdot z) = (x \cdot y) \cdot z$	11*b*.	$x + (\, y + z) = (x + y) + z$
12*a*.	$x \cdot (\, y + z) = x \cdot y + x \cdot z$	12*b*.	$x + y \cdot z = (x + y) \cdot (x + z)$
13*a*.	$x + x \cdot y = x$	13*b*.	$x \cdot (x + y) = x$
14*a*.	$x \cdot y + x \cdot \overline{y} = x$	14*b*.	$(x + y) \cdot (x + \overline{y}) = x$
15*a*.	$\overline{x \cdot y} = \overline{x} + \overline{y}$	15*b*.	$\overline{x + y} = \overline{x} \cdot \overline{y}$
16*a*.	$x + \overline{x} \cdot y = x + y$	16*b*.	$x \cdot (\overline{x} + y) = x \cdot y$
17*a*.	$x \cdot y + y \cdot z + \overline{x} \cdot z = x \cdot y + \overline{x} \cdot z$	17*b*.	$(x + y) \cdot (y + z) \cdot (\overline{x} + z) = (x + y) \cdot (\overline{x} + z)$