UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

ECE253F – Digital and Computer Systems
Final Examination

December 12, 2022 2:00pm - 4:30pm
Duration: 150 minutes

Examiners: Profs. N. Enright Jerger and M. Jeffrey

Please enter your name and student number in the spaces provided above as it appears on Quercus. It is important that your name exactly match the Quercus gradebook.

Exam Type D: Examiner specified aids: One single sheet of letter size paper (8.5 x 11 inch), both sides may be used.

Calculator Type 4: No calculators or other electronic devices are allowed.

All questions are to be answered on the examination paper. Your answer **MUST** be fully contained on the same page as the question. **Any material written on the back of each page will be ignored.** Exams will be scanned – please write clearly in **pen or dark pencil**.

Please state any assumptions you make when answering a question.

The number of marks for each question are indicated. The exam has **24 pages**, including this one.

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Total
5	6	12	3	8	13	8	13	10	12	16	4	110

Question 1 [5 Marks]

[2 marks] (a) Fill in the following table with the appropriate number conversions. If there is no possible answer, explain why.

Decimal	8-bit 2's complement
-57	1) 11000111
2) -6	**1111 1010**

You may use the space below for your calculations. Please indicate which part you are solving by writing the corresponding number from the boxes above.

[3 marks] (b) In class, you learned how to convert between decimal (base 10), binary (base 2) and hexadecimal (base 16) for signed numbers. Using the methods taught to you, this question asks you to convert to/from octal (base 8).

(a) Convert $(17)_{10}$ to Octal

$(21)_8$

(b) Convert $(1100111)_2$ to Octal

$(147)_8$

(c) Convert $(23)_8$ to decimal

$(19)_{10}$

Question 2 [6 Marks]

[3 marks] (a) For the K-map given below, derive the minimum-cost cover as a **sum-of-products expression**. 'X' in the K-map indicates a "don't care".

$f(a,b,c,d) = \bar{a}\bar{b} + \bar{a}\bar{c} + bd + \bar{c}d$

cd \ ab	00	01	11	10
00	1	1	0	0
01	X	X	1	1
11	1	1	1	0
10	1	0	X	0

[3 marks] (b) For the K-map given below, derive the minimum-cost cover as a **product-of-sums expression**. 'X' in the K-map indicates a "don't care".

$f(a,b,c,d) = (\bar{a} + \bar{b} + \bar{c})d$

cd \ ab	00	01	11	10
00	X	0	X	0
01	1	X	1	X
11	1	1	0	1
10	0	X	0	X

Question 3 [12 Marks]

Consider the following RISC-V assembly language code with memory addresses shown for every instruction or data word.

```
                        .text
                        .global _start
0x00000100    _start:   la s0, U
0x00000104              lw a0, 0(s0)
0x00000108              jal HELLO
0x0000010C              sw a0, 0(s0)
0x00000110              ebreak
0x00000114    HELLO:    mv t0, zero # t0 = 0
0x00000118              mv t1, a0
0x0000011C              mv t2, zero
0x00000120    LOOP:     beqz t1, BYE
0x00000124              lw t2, 4(t1)
0x00000128              sw t0, 4(t1)
0x0000012C              mv t0, t1
0x00000130              mv t1, t2
0x00000134              j LOOP
0x00000138    BYE:      mv a0, t0
0x0000013C              jr ra
...
                        .data
0x0000020C    U:        .word V
0x00000210    V:        .word 2, W

0x00000218    W:        .word 4, X

0x00000220    X:        .word 5, 0
```

[3 marks] (a) Suppose this program is executed on a RISC-V processor. What are the values of the RISC-V registers shown below the first time the code reaches the instruction at address 0x134? Provide numeric values and specify the number format.

s0	0x20C	a0	0x210	ra	0x10C
t0	0x210	t1	0x218	t2	0x218

[7 marks] (b) What are the values in memory at the following addresses when the code reaches the instruction at 0x110? Provide numeric values and specify the number format.

Memory address	**Value of 4-byte word**
0x0000020C	0x220
0x00000210	2
0x00000214	0
0x00000218	4
0x0000021C	0x210
0x00000220	5
0x00000224	0x218

[2 marks] (c) What does this code "do" at a high level? That is, describe what is stored in U after running the program.

The code reverses the linked list whose head is initially held in U. The code then replaces the old head in U with the new head (former tail) in U.

Question 4 [3 Marks]

The following segment of RISC-V assembly code has an error. Rewrite the code, retaining its functionality, but fixing the error. You do not know `LIBROUTINE`'s implementation details, but it is implemented correctly.

```
...
addi t1, s6, 3
add  a0, zero, s2
jal  LIBROUTINE
add  a0, t1, a0
jal  putchar      # Prints the least significant byte of its argument
...               # to STDOUT as an ASCII character
```

(a) Briefly explain the error.

The `t1` value is produced before the call of LIBROUTINE and used after the call. But `t1` is not a preserved register, so could be trampled by LIBROUTINE.

(b) Rewrite the code, retaining its functionality, but fixing the error.

A few solutions follow:

This replaces `t1` with a saved register, assuming `s0` is not used by the following code.

```
...
addi s0, s6, 3
add  a0, zero, s2
jal  LIBROUTINE
add  a0, s0, a0
jal  putchar
...
```

Push t1 and pop t1 to/from the stack:

```
...
addi t1, s6, 3
addi sp, sp, -4
sw   t1, 0(sp)
add  a0, zero, s2
jal  LIBROUTINE
lw   t1, 0(sp)
addi sp, sp, 4
add  a0, t1, a0
jal  putchar
...
```

Since the call to `LIBROUTINE` does not depend on `t1`, move the `addi` after the call:

```
...
add  a0, zero, s2
jal  LIBROUTINE
addi t1, s6, 3
add  a0, t1, a0
jal  putchar
...
```

Question 5 [8 Marks]

On the next page, write a RISC-V assembly language subroutine that implements the recursive Euclidean algorithm to compute the greatest common divisor (GCD) of two integers. The mathematical representation is shown below on the left, while a `C` implementation is on the right.

$$\gcd(x, y) = \begin{cases} x & \text{if } y = 0 \\ \gcd(y, \text{remainder}(x, y)) & \text{if } y > 0 \end{cases}$$

```
int GCD(int X, int Y) {
  if (Y == 0)
    return X;
  else
    return GCD(Y, X % Y);
}
```

Your subroutine must be recursive.

You must also provide a main program that calls your `GCD` subroutine. The values of the arguments X and Y should be loaded from a memory locations with labels X and Y, and passed to your subroutine. Put the GCD of X and Y in `s3` after the call. Assume that X and Y ≥ 0. A skeleton has been provided for you to fill in on the next page. (Hint: the RISC-V assembly for `s3=s1%s2` is `rem s3,s1,s2`.)

Question 5 continued . . .

```
    .data
X:  .word 21
Y:  .word 12
    .text
.global _start


_start: la s0, X
        la s1, Y
        lw a0, 0(s0)
        lw a1, 0(s1)
        jal GCD
        mv s3, a0
        ebreak
GCD:    beqz a1, DONE # a0 already holds x
        addi sp, sp, -4
        sw ra, 0(sp)
        rem t0, a0, a1 # t0 = X % Y
        mv a0, a1
        mv a1, t0
        jal GCD
        lw ra, 0(sp)
        addi sp, sp, 4
DONE:   jr ra
```

Question 6 [13 Marks]

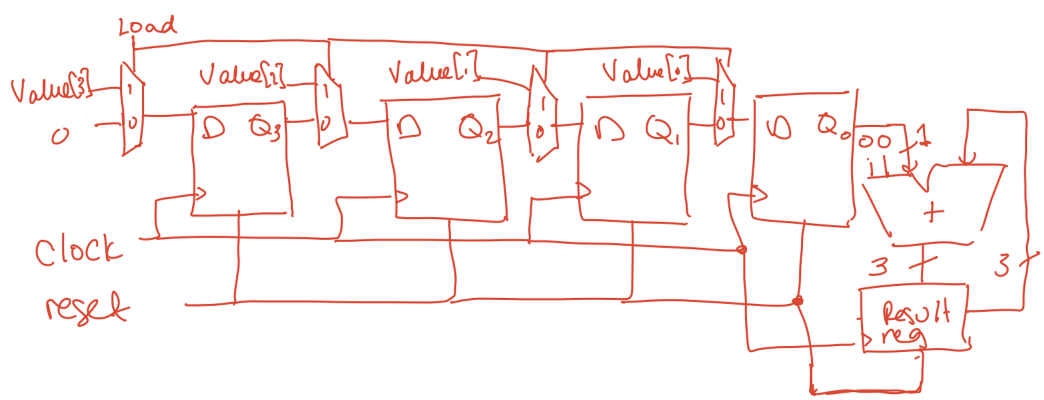
In this question, you must design a pop count circuit. This circuit takes a 4-bit binary number and counts the number of ones in it. You must do this using a 4-bit shift register and an adder. Each cycle, you must shift 1 bit out of your shift register to add to your current total. The current total must be stored in a result register. Your circuit must meet the following requirements:

- The shift register must have a parallel load feature for setting a new value
- Your circuit must use a synchronous, active high reset.
- You may assume that the circuit will be reset before every new input.

[1 mark] (a) How many bits should your result register be?

3 bits

[4 marks] (b) Draw a block diagram for your circuit. Your diagram should only use basic circuit elements (e.g., gates, muxes, flip-flops, adders). Use the input and output names provided in the module signature in part (c) on the next page. Be sure to indicate the bit widths of all the signals in your diagram.



[8 marks] (c) Complete the following System Verilog module to implement the described circuit. Be sure to specify the signal width of `result`.

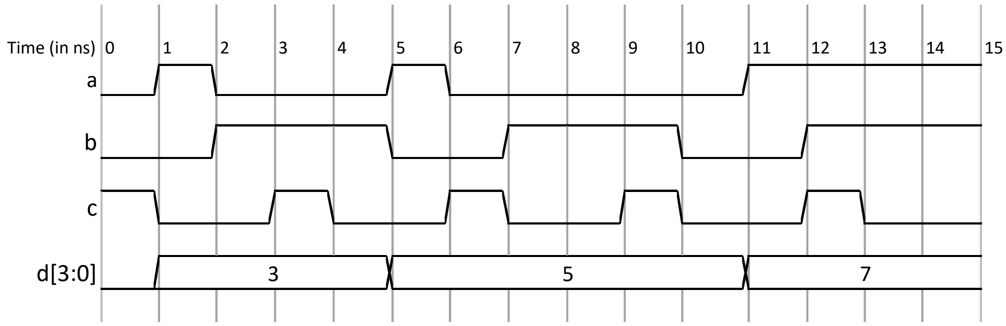
```
module popcount(input logic clock, reset, load,
                input logic [3:0] value,
                output logic [  :0] result, // enter the right value here
);
```

```
logic [3:0] Q; // shift register

always_ff @ (posedge clock) begin
  if (reset) begin
    result <= 3'b0;
    Q <= 4'b0;
  end
  else
  begin
    if (Load)
       Q <= value;
    else
    begin
       Q <= {1'b0,Q[3:1]};
       result <= result + Q[0];
    end
  end
endmodule
```

result reg should be 3 bits

Question 7 [8 Marks]

In this question, you must write the appropriate ModelSim commands to create the waveform shown below:



Complete the following *.do* file to generate the waveform from time $t = 0ns$ to $t = 15ns$.

```
vlib work
vlog question6.sv
vsim question6
log {/*}
add wave {/*}
```

[4 marks] (a) Write just **one line each** to create the signals *b* and *c*. Using more than one line for each signal will get 0 marks.

force b 0 0ns , 1 2ns -r 5ns or force {b} 0 {0ns} , 1 {2ns} -r {5ns}
force c 1 0ns , 0 1ns -r 3ns or force {c} 1 {0ns} , 0 {2ns} -r {3ns}

[4 marks] (b) Write the commands to create the signals *a* and *d*. You must use the minimum number of commands to get full marks. Unnecessary commands will be penalized.

force a 0
force d 0
run 1ns

force a 1
force d 11
run 1ns

force a 0
run 3ns

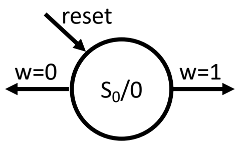
force a 1
force d 101

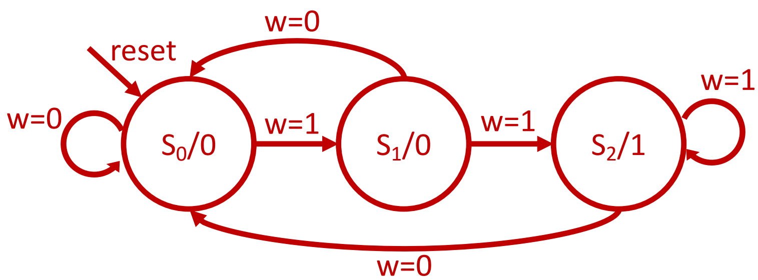
```
run 1ns

force a 0
run 5ns

force a 1
force d 111
run 4ns
```

Question 8 [13 Marks]

[3 marks] (a) Design the state transition diagram for a finite state machine with one input w and one output X. If w has been 1 for at least two consecutive cycles, X should be 1 the next cycle. Otherwise X should be 0 the next cycle.

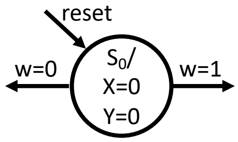
Use as few states as possible. Partial marks are awarded for correct solutions with more states than necessary. Consider drafting your diagram on a scratch page first.

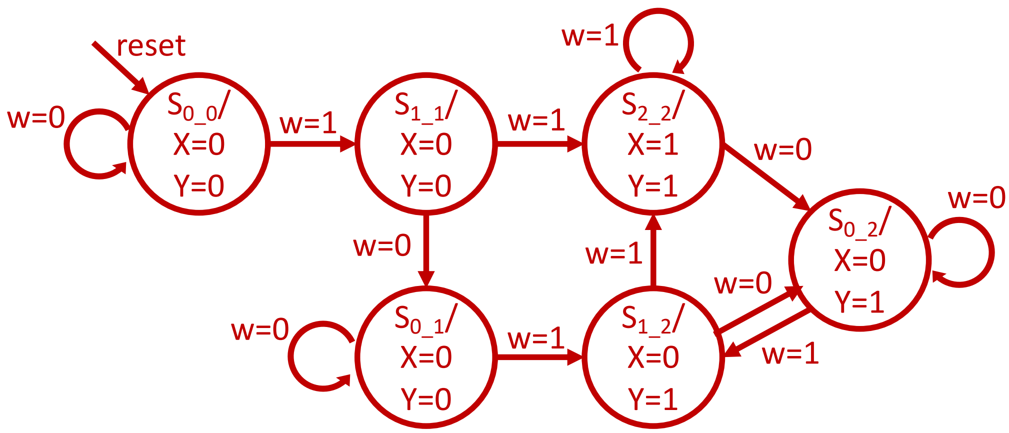




[10 marks] (b) Design the state transition diagram for a finite state machine with one input w and two outputs X and Y. If w has been 1 for at least two consecutive cycles, X should be 1 the next cycle. Otherwise X should be 0 the next cycle. If w has been 1 for at least two cycles altogether (not necessarily consecutively), Y should be 1 the next cycle. Otherwise Y should be 0 the next cycle. For example, Y should be 1 and stay 1 after sequences of w values such as 1,0,1 or 1,1 but not 0,1,0.

Use as few states as possible. Partial marks are awarded for correct solutions with more states than necessary. Consider drafting your diagram on a scratch page first.





Question 9 [10 Marks]

[6 marks] (a) Use Boolean algebra to express $f(x, y, z) = \overline{(x + y)(\bar{x} + z)}$ as a **product of maxterms**. For full marks, show your work and state the theorems used. If you do not know the theorem name, write the simplified form (e.g., $x + x = x$).

$$
\begin{aligned}
f(x,y,z) &= \overline{(x+y)(\bar{x}+z)} && \\
&= \overline{x\bar{x} + xz + \bar{x}y + yz} && \text{Distributive} \\
&= \overline{xz + \bar{x}y + yz} && \text{Complements} \\
&= \overline{xz} \cdot \overline{\bar{x}y} \cdot \overline{yz} && \text{Demorgan's} \\
&= (\bar{x}+\bar{z})(x+\bar{y})(\bar{y}+\bar{z}) && \text{Demorgan's} \\
&= (\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+\bar{z})(x+\bar{y}+z)(x+\bar{y}+\bar{z})(x+\bar{y}+\bar{z})(\bar{x}+\bar{y}+\bar{z}) && \text{Combining} \\
&= (x+\bar{y}+z)(x+\bar{y}+\bar{z})(\bar{x}+y+\bar{z})(\bar{x}+\bar{y}+\bar{z}) && \text{Idempotency}
\end{aligned}
$$

[4 marks] (b) Use Boolean algebra to derive a **minimal sum-of-products** expression for

$$f(w,x,y,z) = \overline{\overline{y\bar{z}}} + (w+\bar{x})(\bar{y}+z).$$

For full marks, show your work and state the theorems used. If you do not know the theorem name, write the simplified form (e.g., $x + x = x$).

$$\begin{aligned} f(x,y,z) &= y\bar{z} + (w+\bar{x})(\bar{y}+z) & \\ &= y\bar{z} + (w+\bar{x})(\overline{y\bar{z}}) & \text{DeMorgan's} \\ &= w + \bar{x} + y\bar{z} & \text{Redundancy/Absorption} \end{aligned}$$

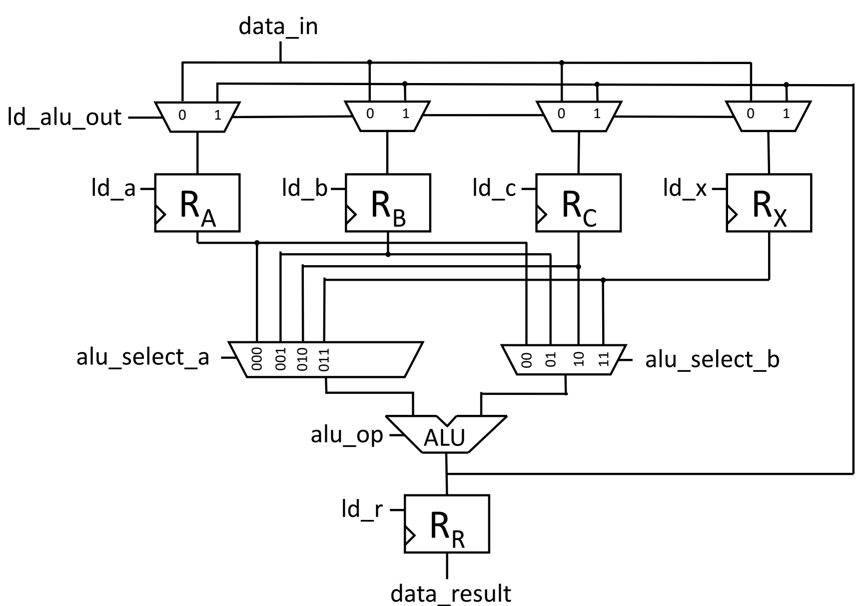
Question 10 [12 Marks]

The control and datapath from Lab 6 have been expanded to implement:

$$result = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

- The ALU can perform the following functions with alu_op codes in parenthesis: Add (000), Subtract (001), Multiplication (010), Square Root (011) and Division (100).
- The square root has only one input. That input should come come from the alu_select_a mux.
- The input to the square root will always be a positive number.
- Overflow will never occur.
- Inputs will be such that only integer results can be obtained.
- Each register holds an 8-bit signed value.
- Initially, registers a, b, and c hold the inputs to the function. Once those initial values have been used, all registers (a, b, c, x) can be used to store partial results. The result register (R_R) should only be updated with the final result.
- When the final result is ready in the register, the `result_valid` signal should be set.

If you would like to use any constants in your solution, you can add them to the extra inputs to the alu_select_a mux. Clearly label them. 100 – 4, 101 – 2



On the next page, fill in the table of control signals and register values. Assume that registers a, b, and c have already been loaded with their input values (shown in the column marked "0"), so the computation will start in Cycle 1. Add comments as necessary to help the marker understand your implementation. The table may contain more columns than needed.

[4 marks] (a) First, indicate what calculation is performed in each cycle.

Cycle #	Calculation performed
1	c = a x c
2	c = 4 x c (this is 4ac)
3	x = b x b (this is b^2)
4	x = x - c (this is $b^2 - 4ac$)
5	x = sqrt(x) (this is $\sqrt{b^2 - 4ac}$)
6	b = x - b (this is $\sqrt{b^2 - 4ac} - b$
7	a = a x 2 (this is $2a$)
8	result = b/a
9	
10	
11	

[8 marks] (b) Fill in the table of control signals and register values below. Each column represents a cycle.

	0	1	2	3	4	5	6	7	8	9	10	11
R_A	1	1	1	1	1	1	1	1	2	2		
R_B	4	4	4	4	4	4	4	-2	-2	-2		
R_C	3	3	3	12	12	12	12	12	12	12		
R_X	0	0	0	0	16	4	2	2	2	2		
R_R	0	0	0	0	0	0	0	0	0	-1		
ld_a		0	0	0	0	0	0	1	0	0		
ld_b		0	0	0	0	0	1	0	0	0		
ld_c		1	1	0	0	0	0	0	0	0		
ld_x		0	0	1	1	1	0	0	0	0		
ld_r		0	0	0	0	0	0	0	1	0		
ld_alu_out		1	1	1	1	1	1	1	X	X		
alu_select_a		000	100	001	011	011	011	101	001	XXX		
alu_select_b		10	10	01	10	XX	01	00	00	XX		
alu_op		010	010	010	001	011	001	010	100	XXX		
result_valid		0	0	0	0	0	0	0	0	1		

Question 11 [16 Marks]

You are to write assembly code that polls the keyboard and lights up LEDs based on the key that is pressed. Your code should start with 0 LEDs on and light LEDs starting from LED0 to LED9. Pressing a key 1-3 should turn on that many more LEDs. For example, if 1 is pressed, 1 more LED must turn on. If 2 is pressed, 2 more LEDs must turn on. If 3 is pressed, 3 more LEDs must turn on. If any other key is pressed, turn off all LEDs. Once all 10 LEDs are on, they should be turned off on the next key press regardless of what key is pressed.

The ASCII code for 1 is 0x50, 2 is 0x51, 3 is 0x52.

The I/Os have the following memory map information:

Keyboard

(a) The ASCII code for the key is written to the Receiver Data register. This register is located in the lowest 8-bits of memory location 0xffff0004).

(b) The Ready bit is set to 1 in the Receiver Control register, which is the least significant bit of memory location 0xffff0000. The Ready bit is automatically reset to 0 when you read the Receiver Data register using a lw instruction.

LEDs

(a) The system has 10 LEDs located at 0xffff8000. The lowest bit is LED0, the next bit is LED1, etc.

(b) A 1 is written to the corresponding LED to turn it on.

The following code is provided to get you started.

```
    .data
KEYBOARD: .word 0xFFFF0000
LEDS: .word 0xFFFF8000

    .text
    .global _start
_start:


  li s2, 0xFFFF0000
  li s4, 0 // Will hold current LED value
  li s9, 3 // compare for key3
  li s5, 0xFFFF8000
  sw s4, 0(s5) // All LEDS off
  addi s7, zero, 0 // count of LEDS
  li s8, 10
POLL: lw s3, 0(s2)
  andi s3, s3, 1
```

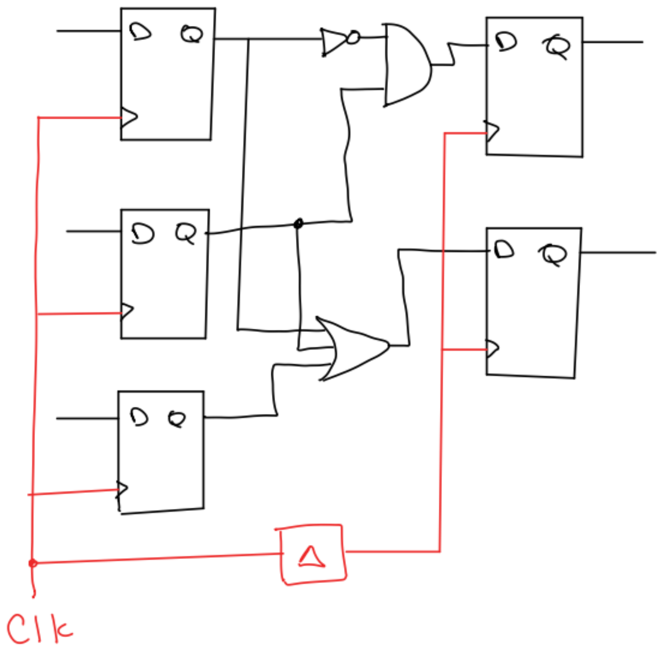
```
  beqz s3, POLL
  lw s3, 4(s2) // ascii value of key
  bne s7, s8, LOOP // are 10 LEDS ON
  addi s4, zero, 0 // value to turn off LEDs
  addi s3, s3, -0x49 // check s3 to 1, 2, 3
  bgtz s3, check_3 // check if key is not 1, 2, 3
  addi s4, zer0, 0
  j UPDATE
  blte s3, s9, LOOP
  addi s4, zer0, 0
  j UPDATE
LOOP: BEQZ s3, UPDATE
  bne s7, s8, LEDS // are 10 LEDS ON?
  j UPDATE
  slli s4, s4, 1
  ori s4, s4, 1
  addi s7, s7, 1
  addi s3, s3, -1
  j LOOP
UPDATE:
  sw s4, 0(s5) // update LED vaalue
  j POLL // resume POLLING
END: ebreak
```

Question 11 continued . . .

Question 11 continued . . .

Question 12 [4 Marks]

Given the following circuit:



With the following timing values:

t_{gate}	1 ns + 0.2 × number of inputs
t_{su}	0.5ns
t_{cq}	1 ns
t_{hold}	0.7ns

(a) Assuming there is no clock skew, what is the maximum operating frequency at which this circuit will operate correctly?

$T_{min} = t_{CQ} + t_{logic} + t_{su}$
$= 1 + 1.2 + 1.4 + 0.5 = 4.1ns$
$F_{max} = 1/4.1ns$

(b) Calculate the value for t_{skew} (shown as Δ in the figure), at which a hold time violation will occur.

$t_h + t_{skew} > t_{CQ} + t_{logic\_min}$
$t_{skew} > 1 + 1.4 - 0.7$
$t_{skew} > 1.7$

This page is intentionally left blank. You may use it as scratch to solve exam problems but please be sure to write your answers in the spaces provided for each question. You must submit this page with your exam.

This page is intentionally left blank. You may use it as scratch to solve exam problems but please be sure to write your answers in the spaces provided for each question. You must submit this page with your exam.