**UNIVERSITY OF TORONTO**

**Fall 2019 Midterm**

**CSC258: Computer Organization**

**Duration: 2 hours**

**October 24th, 2019**

**Last Name:** _____

**First Name:** _____

**Student Number:** _____

**Lecture section:** **L0101 – MWF 11am**

**L5101 – T 6pm**

**1 mark each for circling your lecture section and writing your name on the last page.**

## Instructions:

- **Write your name on the back of this exam paper.**
- **Do not open this exam until you hear the signal to start.**
- Have your student ID on your desk.
- No aids permitted other than writing tools. If you write in pencil, we reserve the right to deny any remark requests.
- Keep all bags and notes far from your desk before the exam begins.
- There are 5 parts on 12 pages. When you hear the signal to start, make sure that your exam is complete before you begin.
- Read over the entire exam before starting.
- If you use any space for rough work or have to use the overflow page, clearly indicate the section(s) that you want marked.

## Mark Breakdown

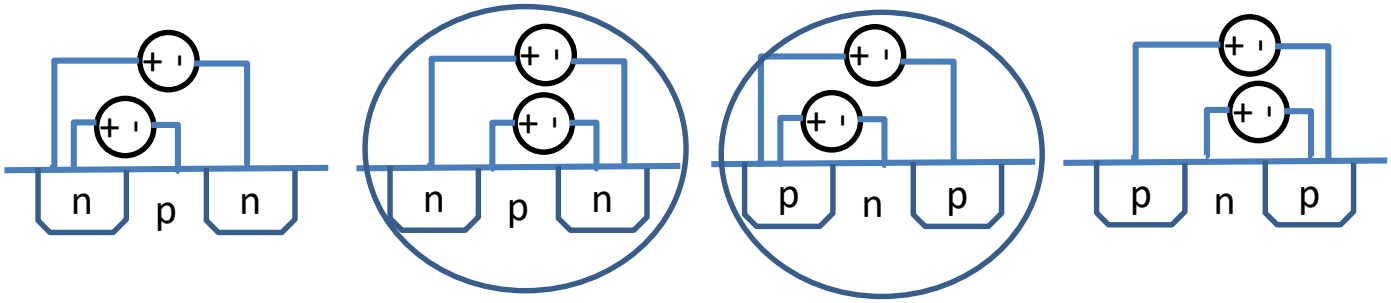
| | |
|---|---|
| **Front/Back:** | **/ 2** |
| **Part A:** | **/ 20** |
| **Part B:** | **/ 25** |
| **Part C:** | **/ 20** |
| **Part D:** | **/ 12** |
| **Part E:** | **/ 10** |
| **Bonus:** | **/ 1** |
| **Total:** | **/ 90** |

## Part A: Short Answer (20 marks)

Answer the following questions in the space provided. When providing a written answer, write **as clearly and legibly as possible**. Marks will not be awarded to unreadable answers.

1. Fill in the blanks:
   A reverse bias applied to a p-n junction causes the depletion region to expand and stop/block the current flowing. **(2 marks)**

2. Given the MOSFET diagrams below, circle the ones that are allowing current to flow. **(2 marks)**




3. Circle the Modelsim command(s) that could cause the waveform on the right. **(1 mark)**

   a. `force clk 0 0, 1 5 -repeat 30`
   b. `force clk 0 0, 1 10 -repeat 30`
   c. `force clk 0 0, 1 15 -repeat 30`
   d. `force clk 0 0, 1 20 -repeat 30`

4. Fill in the blank:
   We need at least 6 flip-flops to implement a state machine with 44 states. **(1 mark)**

5. Assuming 2's complement notation with 9 bits, give the binary representation of -120. **(1 marks)**

   **Binary representation:** 110001000

6. Fill in the blanks: For a logic function with 4 inputs A, B, C, D, the minterm $m_5$ is A'BC'D and maxterm $M_9$ is A'+B+C+D'. **(2 marks)**

7. Fill in the blank:
   If we change the order of inputs to C, B, A, D, the minterm $m_5$ becomes C'BA'D. **(1 mark)**

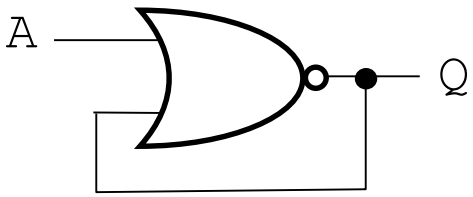
8. True or False? The number of different numbers that can be represented by n-bit 2's complement signed numbers is more than that of n-bit 1's complement signed numbers. **(1 mark)**

**TRUE**

9. True or False? The number of different numbers that can be represented by n-bit 2's complement signed numbers is more than that of n-bit unsigned numbers. **(1 mark)**

**FALSE**

10. A specific input causes an undesired behaviour in below feedback circuit. Give that input and show the corresponding undesired behaviour. **(2 marks)**




**If A is set to 0, Q keeps toggling.**

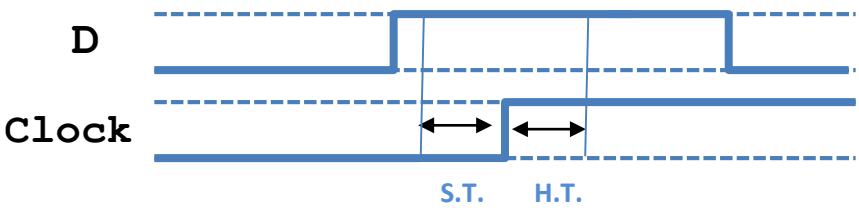
11. Fill in the blank:
By introducing impurities to semiconductors, their electrical resistance _ decreases ___. **(1 mark)**

12. If `HEX0[0:6]` is assigned the value `0111000`, what alphanumeric character is displayed on the 7-segment display? **(1 mark)**

**F**

13. What is the main difference between a Moore state machine and a Mealy state machine? **(1 mark)**
**Mealy machine**: Output depends on current state and current input
**Moore machine:** Output depends only on current state

14. The diagram below illustrates the input signals of a D flip-flop being assigned a value of 1. On the diagram, label the areas corresponding to setup time and hold time. **(2 marks)**




15. What is the main difference between synchronous reset and asynchronous reset? **(1 mark)**

**Asynchronous reset** does not require an active clock to bring flip-flops to a known state.
**Synchronous reset** works only when applied at the clock edge.

## Part B: Slightly Longer Answer! (25 marks)

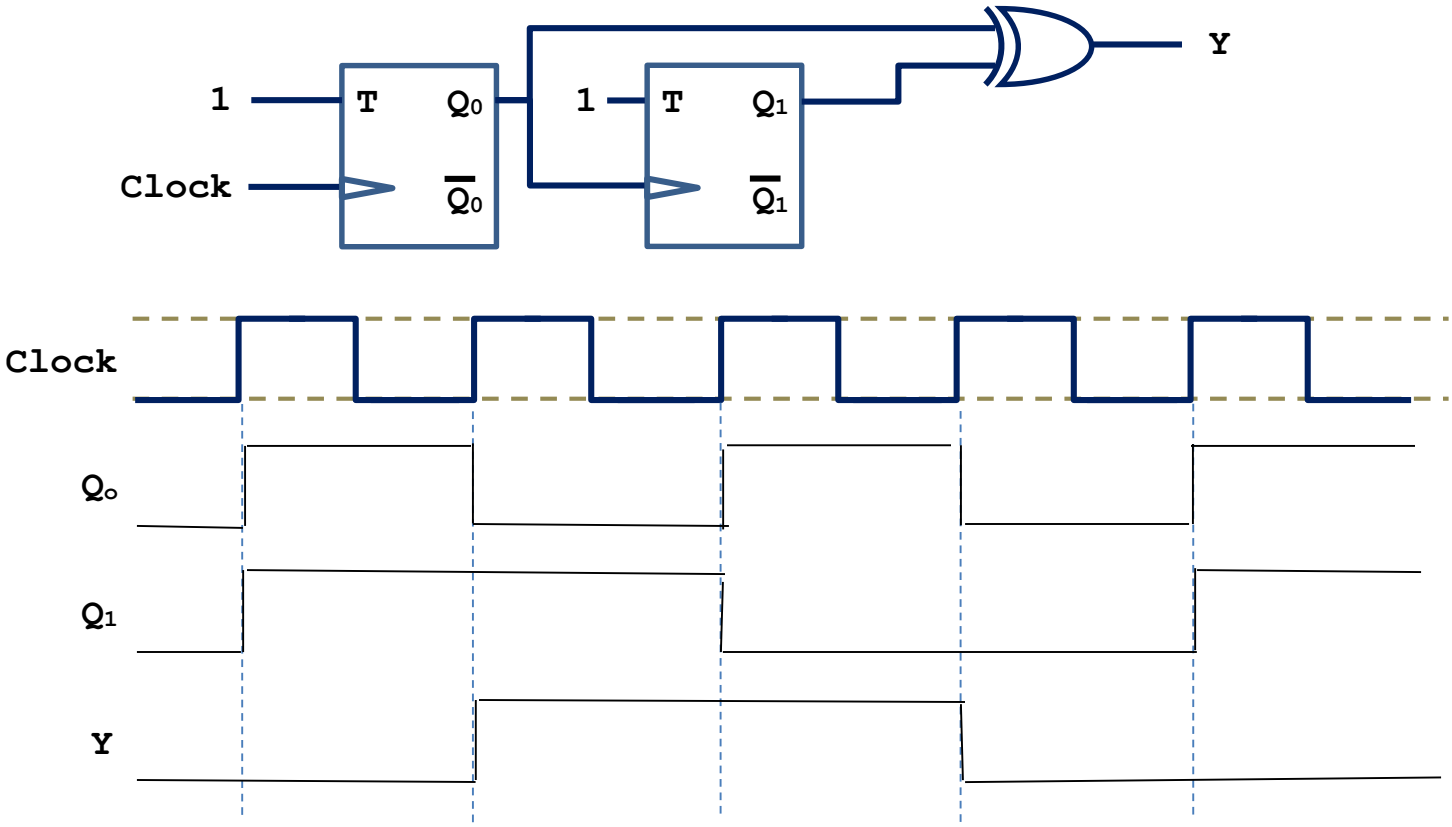
Answer the following questions in the space provided. The final answer is all that is necessary, but showing your work can help if your final answer isn't correct. Again, make sure to write legibly here.

**1.** Use Boolean identities to prove that below identity holds: **(2 marks)**

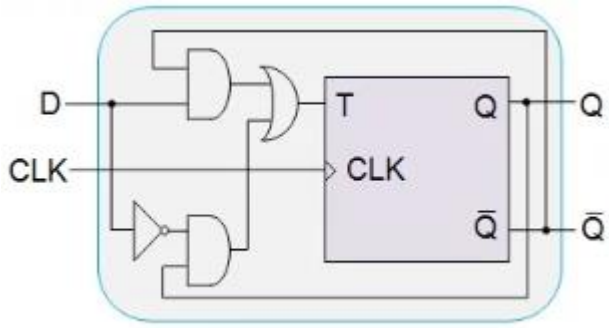
$$x + (x \cdot y) = x$$

**X+ (X.Y) = (X.1)+(X.Y) = X.(1+Y) = X.1 = X**

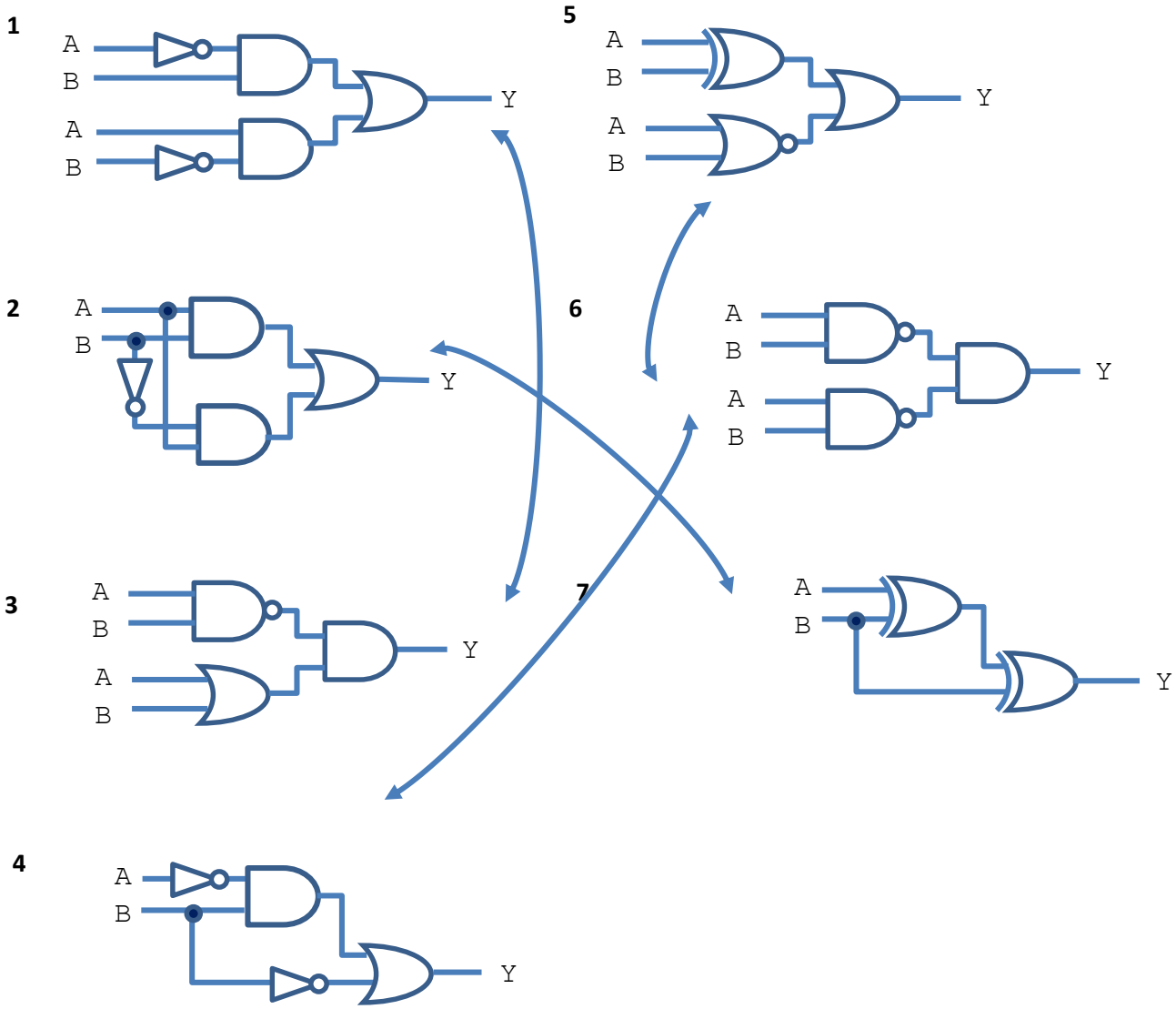
**2.** For the sequential circuit below, fill in the waveform behaviour for the outputs $Q_0$, $Q_1$ and Y. Assume the flip-flop output values are all low before the first positive clock edge. **(6 marks)**




**3.** Create a D flip-flop using a T flip-flop and the fewest number of AND, OR, and NOT gates. **(3 marks)**



**4.** Consider the combinational circuits below. Indicate which circuits have the same output behaviour by drawing lines to connect equivalent circuits. **(14 marks)**




1) A'B + AB'
2) AB+AB' = A
3) (AB)'(A+B) = (A'+B')(A+B) = A(A'+B') + B(A'+B') = AB'+A'B
4) A`B + B` = A'+B'
5) B` + A`B = A'+B'
6) A^B+(A+B)'=AB'+A'B+A'B'= AB'+A'=A`+B`
7) (A^B)^B=(AB'+A'B)^B=(AB'+A'B)B'+(AB'+A'B)'B=AB'+(A'+B)(A+B')B=AB'+(A'+B)AB=AB'+AB=A

Ans: 1 ⇔ 3, 2 ⇔ 7, 4 ⇔ 5⇔6

## Part C: Circuit Design and Analysis (20 marks)

1. Consider a three-bit state machine, whose state table is drawn on the right. In the Karnaugh maps shown below, fill in the next values for $F_2$, $F_1$ and $F_0$. **(6 marks)**
2. Make groupings on your Karnaugh maps that result in the lowest-cost implementation for $F_2$, $F_1$ and $F_0$, and provide the formulae for this implementation in the spaces on the bottom right. **(9 marks)**

| Previous State | | | | Next State | | |
|---|---|---|---|---|---|---|
| $F_2$ | $F_1$ | $F_0$ | EN | $F_2$ | $F_1$ | $F_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |

| $F_2$ | $\overline{F_0}\,\overline{EN}$ | $\overline{F_0}\,EN$ | $F_0\,EN$ | $F_0\,\overline{EN}$ |
|---|---|---|---|---|
| $\overline{F_2}\,\overline{F_1}$ | 0 | 0 | 0 | 0 |
| $\overline{F_2}\,F_1$ | 0 | 1 | 1 | 0 |
| $F_2\,F_1$ | X | X | X | X |
| $F_2\,\overline{F_1}$ | 1 | 0 | 0 | 1 |

| $F_1$ | $\overline{F_0}\,\overline{EN}$ | $\overline{F_0}\,EN$ | $F_0\,EN$ | $F_0\,\overline{EN}$ |
|---|---|---|---|---|
| $\overline{F_2}\,\overline{F_1}$ | 0 | 1 | 1 | 0 |
| $\overline{F_2}\,F_1$ | 1 | 0 | 0 | 1 |
| $F_2\,F_1$ | X | X | X | X |
| $F_2\,\overline{F_1}$ | 0 | 0 | 0 | 0 |

| $F_0$ | $\overline{F_0}\,\overline{EN}$ | $\overline{F_0}\,EN$ | $F_0\,EN$ | $F_0\,\overline{EN}$ |
|---|---|---|---|---|
| $\overline{F_2}\,\overline{F_1}$ | 0 | 0 | 1 | 1 |
| $\overline{F_2}\,F_1$ | 0 | 0 | 1 | 1 |
| $F_2\,F_1$ | X | X | X | X |
| $F_2\,\overline{F_1}$ | 0 | 1 | 0 | 1 |

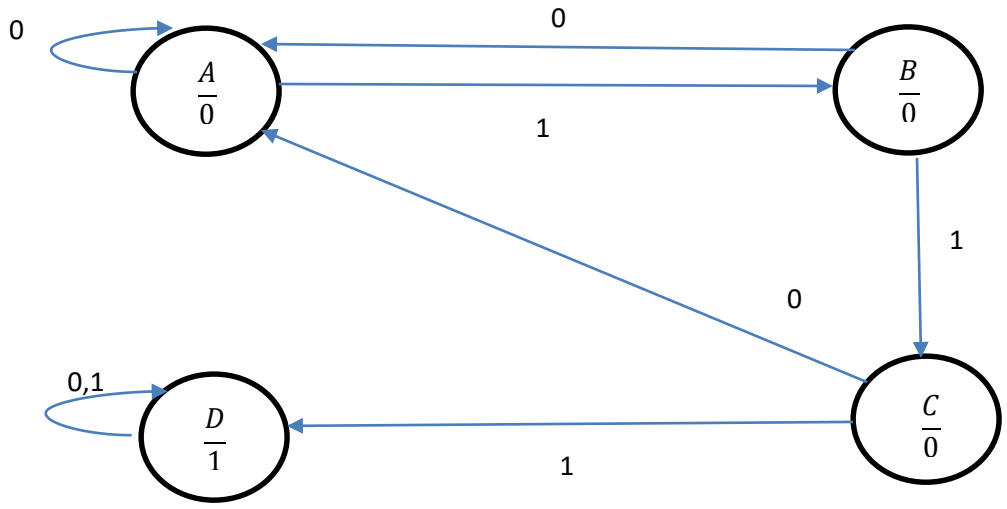
$$F_2 = F_2EN' + F_1EN$$

$$F_1 = F_1EN` + ENF_2`F_1`$$

$$F_0 = F_2'F_0 + F_0EN` + F_2F_0`EN$$

3. Apple is changing its passcode locking system, so that users who enter the incorrect password three times in a row permanently lock their phone. Entering the code correctly resets this count back to zero.

   In the space below, draw the state diagram for a Moore machine that takes in the 1-bit signal `ERROR` (which is `0` if the code was entered correctly and `1` if it was entered incorrectly), with an output that goes high after three incorrect attempts in a row. **(5 marks)**



## Part D: Devices (12 marks)

**1.** The Verilog modules below implement various devices, as indicated by the module names. Each one has a broken line in the middle (in bold) that needs to be fixed. In the space provided on the right, write the lines of replacement Verilog code to fix the module. **(4 marks)**

```
module ha (X, Y, B, C);
  output X, Y;
  input B, C;
  {X, Y} <= {B ^ C, B & C};
endmodule
```

**Either** assign {X, Y} = {B ^ C, B & C};
**OR** assign {X, Y} = { B & C, B ^ C};

```
module mystery (X, Y, B, C);
  output X, Y;
  input B, C;
  always @(*) { X = B && ~C;
    Y = ~B && C;
  }
endmodule
```

output **reg** X, Y;
always @(*) **begin**
X = B && ~C;
Y = ~B && C;
**end**

```
module dff (Y, B, C);
  output Y;
  input B, C;
  reg Y;
  always @ posedge(C)
    Y <= B;
endmodule
```

always @ **(posedge C)**

```
module my_latch (Y, B, C);
  output wire [3:0] Y;
  input B, C;
  always @ (C)
    if (C)
      Y = { 4{B} };
endmodule
```

output **reg** [3:0] Y;
input B, C;
always @ (C**, B)**
if (C)
Y = { 4{B} };

**2.** The Verilog modules below are for logical and sequential devices that we have seen in either lectures or labs. In the space below each module, indicate what device that module is implementing.

Full marks are given to more specific answers. For instance, for a clocked SR latch, writing "latch" will only earn you one mark. **(8 marks)**

```
module one (X, Y, B, C);
  output X, Y;
  input B, C;
  reg X, Y;
  always @ (posedge C)
  begin
    Y = X;
    X = B;
  end
endmodule
```

**2-bit shift register**

```
module two (X, Y, B, C);
  output Y;
  input X, B, C;
  reg Y;
  always @ (posedge X)
    if (~B && C)
      Y <= 0;
    else if (B && ~C)
      Y <= 1;
    else if (B && C)
      Y <= ~Y;
endmodule
```

**JK Flip-flop**

```
module three (Y, B, C);
  output Y;
  input B, C;
  reg Y;
  always @ (posedge C)
    Y <= ~Y;
endmodule
```

**T flip-flop with T=1**
**(An always toggling FF)**

```
module four (Y, B, C);
  output Y;
  input B, C;
  reg Y;
  always @ (B, C)
    if (C)
      Y <= B;
endmodule
```

**Clocked D-latch**
**or, D-latch with control/gate**

## Part E: Verilog (10 marks)

> **Note: The 20% rule is in effect on this part.** If you don't know how to do this question, write "I don't know" and you will get 2 marks automatically.

**1.** In the space below, complete the Verilog module for a Rate Divider. Assuming that the input clock frequenc

y is 10 MHz, produce an output signal whose frequency is 1 Hz. **(4 marks)**

```
// 1 mark penalty for each mistake

module rate_divider (output reg clk_out, input clk_in, reset);

        reg [22:0] count;

        always @ (posedge clk_in)
                if(reset)
                 begin
                    clk_out <= 0;
                    count <= 0;
                 end
                 else
                     if (count != 5000000)  //23'd5000000
                          count <= count + 1;
                     else
                      begin
                          clk_out <= ~clk_out;
                          count <= 0;  // 23'd0
                      end
endmodule
```

**2.** In the space below, complete the Verilog module for a simple ALU with three 2-bit inputs A, B, and f, and a 2-bit output Z. The ALU should work like this:

If f is 0, Z should be A+B;
if f is 1, Z should shift-left the A input by B bits;
otherwise, Z should be equal to A.

For full marks, use a `case` statement to implement this module. **(6 marks)**

```
// 1 mark penalty for each mistake

module ALU (output reg [1:0] Z, input [1:0] A, B, f);

        always @ (f, A, B) // always @(*)
                case(f)
                        0 :       Z = A+B;
                        1:        Z = A<<B;
                        default:  Z = A;
                endcase

endmodule
```

**Bonus Question:** ***What are the names of two of the TAs in your lab room? (1 mark)***

_____

***The rest of this page is left blank intentionally for answer overflows.***

Please enter your first and last name in the space below. Do NOT write your student number here.