CS226 Mid-Semester Examination (Spring 2018)

Max marks: 50 **Time: 120 mins**

- ***The exam is open book and notes brought to the exam hall.***
- *Be brief, complete and stick to what has been asked.*
- *Unless asked for explicitly, you may cite results/proofs covered in class without reproducing them.*
- *If you need to make any assumptions, state them clearly.*
- ***Please start writing your answer to each sub-question on a fresh page. DO NOT write answers to multiple sub-questions on the same page.***
- ***The use of internet enabled devices is strictly prohibited. You will be debarred from taking the examination if you are found accessing the internet during the examination. All IIT Bombay rules apply in this respect.***
- ***Please do not engage in unfair or dishonest practices during the examination. Anybody found indulging in such practices will be referred to the D-ADAC.***

1. *[10 marks]* Represent the Boolean function $f(x_1, x_2, x_3, x_4, x_5) = \mathsf{ite}(x_1, \overline{x_2} + (x_3 \cdot x_4), \overline{x_5} + (\overline{x_3} + \overline{x_4}))$ as an ROBDD with complement edges using the variable order $x_1 < x_2 < x_5 < x_3 < x_4$ (x_1 is the top-most variable). Use solid edges for 1-edges and dotted edges for 0-edges. Your final solution must not have any 1-leaf or any solid edge (1-edge) with a bubble on it. However, the root of the ROBDD (with complement edges) may have a bubble on its output, if necessary. Solutions that violate the above constraints will not be awarded any marks.

2. *[15 marks]* We wish to implement the following numerical algorithm where X and Y are signed n-bit integers and $Y \neq 0$. In the algorithm, some statements are specified as executable in parallel. Specifically, parallelly executable statements $s_1, s_2, \ldots s_n$ are represented as `PAR(s1, s2, ... sn);`. All statements in a `PAR(...)` construct must be executed to completion before the statement following the `PAR(...)` construct can be executed.

```
L0: done = 0;
   // read X and Y, initialize T and D, all in parallel
L1: PAR(read(X),  read(Y), T = 1, D = 0);

L2: while (D < Y) {
     // assign to T and D in parallel
L3:  PAR(T = T * Y, D = D + X);

     // sequential statements
L4:  D = D + 1;
L5:  T = T * D;
L6: }

l7: output(T);
L8: done = 1;
```

You are required to design a controller to implement the above algorithm using the datapath shown in Fig. 1. This datapath has five registers, `T, Y, D, X` and `Res` and two hardwired constant values **0** and **1**. Assume that the adder and multiplier implement signed addition and multiplication, respectively, and the comparator implements signed comparison. Assume further that the bit-widths are wide enough so that there are no overflows in any arithmetic operation. Note that the multiplexor control signals MT and MD are two bits wide, while all other multiplexor control signals are one bit wide.

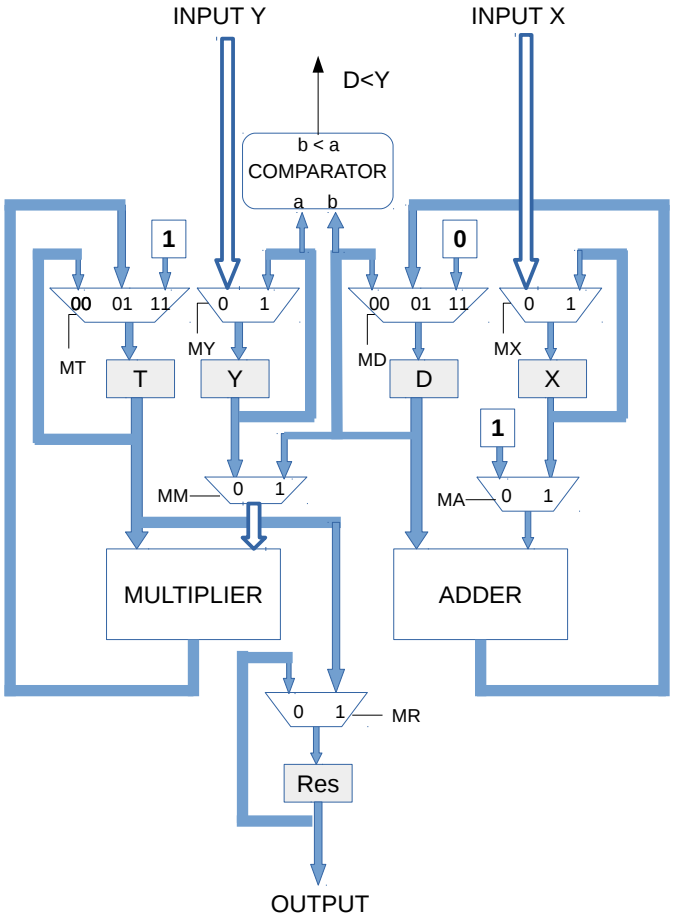

Figure 1: Datapath for Question 1

Your implementation ***must*** execute all statements in a `PAR(...)` construct in parallel. Execution of `output(T);` should result in loading the value of register `T` in the result register `Res` in Fig. 1. The controller must also reset a `done` output to 0 when the computation starts, and set it to 1 after the final result is available in register `Res`. The value in register `Res` when `done` is 0 is inconsequential. You may assume that all registers in the datapath, as well as all flip-flops in your controller, are positive edge-triggered, and are fed by the same clock.

You ***MAY NOT*** assume that a `reset` input is available in the datapath to reset all registers to have the value 0. However, you may assume that the controller, when switched on, has its flip-flops set to values that correspond to the starting state of the controller.

You must provide your solution in the form of a table as shown below. In each row of the table, you must clearly indicate comments that explain what actions are effected in the datapath by the signals that are assigned in that row.

CurrentState	$D < Y$	NextState	MT	MY	MD	MX	MM	MA	MR	done	Comment
...	...	...	...	...	...	...	...	...	...	...	...
⋮											

3. A 3-input programmable *Look-Up Table* (or *LUT*, for short) is a circuit element that has three inputs and one-output, and that can be programmed to implement any arbitrary Boolean function of 3 inputs.

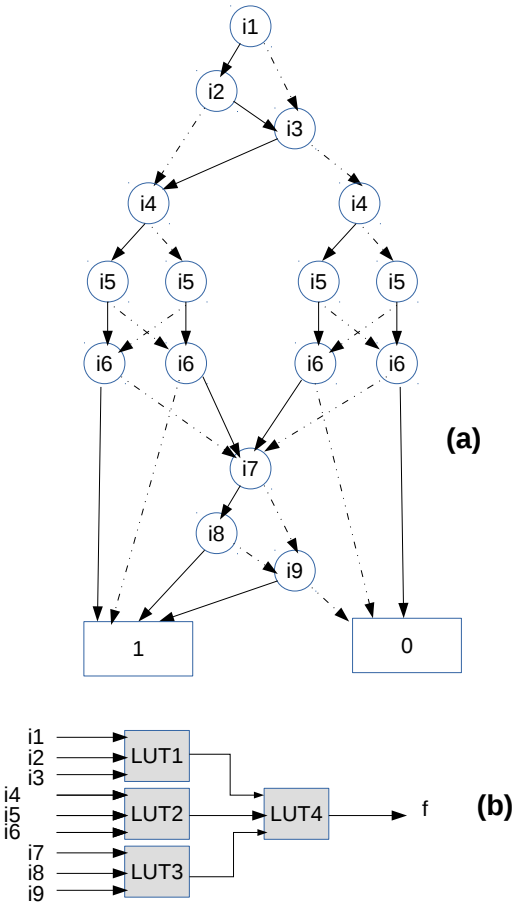

Figure 2: ROBDD and LUTs for Question 2

(a) *[10 marks]* You are given a Boolean function f of 9 inputs specifed by the ROBDD shown in Fig. 2a. Here, dotted edges represent 0-edges and solid edges represent 1-edges. You are required to implement this using four 3-input LUTs connected as shown in Fig. 2b. Indicate what Boolean functions must each of LUT1, LUT2, LUT3 and LUT4 implement. Assume that the 4 LUTs can be programmed independently, i.e. the Boolean function implemented by one LUT need not have any relation to the Boolean function implemented by another LUT. Answers without justification will fetch no marks.

(b) *[5 marks]* Argue whether it is possible to implement every 9-input Boolean function by choosing appropriate 3-input Boolean functions to be implemented by each of the four LUTs shown in Fig. 2b. You are not allowed to change the interconnection between the four LUTs. However, you are free to program each LUT to implement any arbitrary 3-input Boolean function. As before, assume that the four LUTs can be programmed independently. You must give complete justification for your answer. Answers without justification will fetch no marks.

4. *[10 marks]* We wish to implement a circuit with two inputs a and b and one output c that has the following behaviour:

   - If a and b are both 0, the output c is 0.
   - If a and b are both 1, the output c is 1.
   - If the values of a and b differ, the output c doesn't change, i.e. it retains its previous value.

   Give an implementation of the above circuit using AND, OR and NOT gates. Assume that changes in a and b happen very slowly compared to the delays of gates in the circuit. Also, assume that either both a and b change simultaneously (i.e. at the same time) or the time separation between a change in a and a change in b is very large compared to the delays of gates in the circuit.