CS226 Quiz 1 (Spring 2017)

Max marks: 40 **Time: 90 mins**

- *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.***
- ***IIT Bombay prohibits the use of communication devices and internet enabled devices during examinations. You will be debarred from taking the examination if you are found accessing the internet during the examination.***
- ***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. *[5 + 5 marks]* Consider the K-maps shown below. A "-" in a K-map square means that the value in the square is a don't-care, i.e. you are free to choose 0 or 1.

(a)

X,Y→ / Z,W↓	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	0	1	0	0
10	1	0	1	1

(b)

X,Y→ / Z,W↓	00	01	11	10
00	1	0	0	0
01	0	-	-	0
11	0	0	-	1
10	0	-	1	1

   (a) You are required to implement the function represented by K-map (a) as a sum-of-products using as few gates as possible, where the only gates available to you are 2-input AND gates, 2-input OR gates and NOT gates. Each gate counts as 1 regardless of the type of gate.

   (b) You are required to implement the function represented by K-map (b) using only 2-to-1 multiplexors, using as few multiplexors as possible. A 2-to-1 multiplexor has 3 Boolean inputs, say i_0, i_1 and c, and one Boolean output, say f_{mux}. It implements the function $f_{\text{mux}}(c, i_0, i_1) = c.i_1 + \overline{c}.i_0$.

2. *[10 marks]* In this question, we want to use multiplexors, as defined above, in a different way to design digital circuits. Specifically, we want to implement the Boolean function $g(x_1, x_2, x_3, x_4)$ shown in the K-map below as $f_{\text{mux}}\left(g_0(x_1, x_2, x_3), g_1(x_2, x_3, x_4), g_2(x_1, x_3, x_4)\right)$, where $f_{\text{mux}}(c, i_0, i_1) = c.i_1 + \overline{c}.i_0$. You are required to give K-maps of $g_0(x_1, x_2, x_3)$, $g_1(x_2, x_3, x_4)$ and $g_2(x_1, x_3, x_4)$ that achieves the above purpose. Please note that there is no unique solution to this question.

x1,x2→ / x3,x4↓	00	01	11	10
00	1	1	1	1
01	0	1	1	0
11	1	1	0	0
10	0	1	1	1

3. *[10 marks]* We want to implement the following algorithm, written in a C-like language **(not VHDL)** using the datapath shown below (Fig. 1). Note that this is the same datapath we used to implement integer division using `FindSmallestIndex` in class. Specifically, the clock inputs of registers are not controlled by the controller. Instead, the controller only controls what inputs appear at the data inputs of various registers. Also, `aLTb` evaluates to 1 if $a < b$ and to 0 otherwise. You need not worry about what the algorithm below is implementing – simply consider it as some data processing algorithm.

```
T = 0; Q = 0; R = 0; A = 0; B = 0;
read A and B;
if (A < B) { R = A; Q = T; output Q and R;}
else {  while (A >= B) {
            temp = FindSmallestIndex(A, B);
            A = A - B * 2^temp;
            temp = FindSmallestIndex(A, B);
            T = T + 2^temp;
            T = T + 2^temp;
 }
}
```

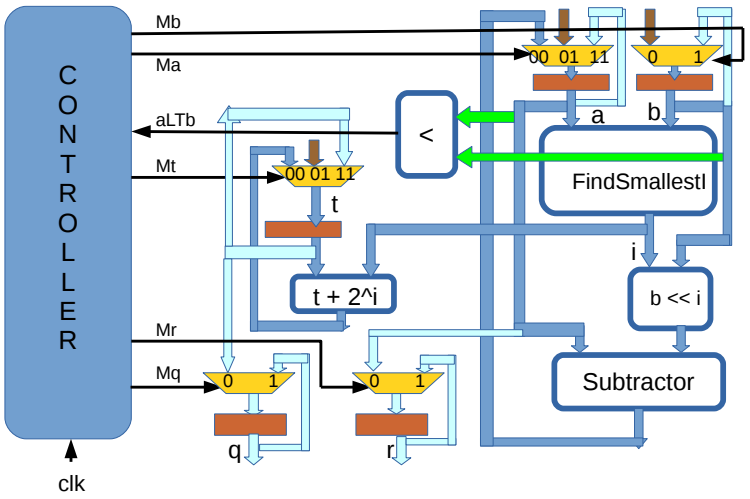

Figure 1: Datapath studied in class

You are free to add or combine intermediate steps of computation in the algorithm above, if it helps you solve the problem. However, all such steps must be clearly indicated.

For the controller, you must give the state transition table in the format given below. You may assume that the values of A and B do not change until the entire computation is over.

You MUST indicate through brief comments what each row of the controller table achieves, e.g. resets registers, or updates T with such and such expression, etc.

CurrState	$aLTB$	NextState	M_a	M_b	M_t	M_q	M_r	Reset	Comment
...	...	...	...	...	...	...	...	...	...
			⋮						

4. *[5 + 5 marks]* Give ROBDDs for the following Boolean functions using the variable order $a < b < c < d < e$, i.e. a appears higher up (closer to the root) in the ROBDD than b and so on,

   (a) $f = (a + b).(c + d).(a + e)$

   (b) $g = (\overline{a} \oplus b).(\overline{c} \oplus d).e$