CS226 End-Semester Examination (Spring 2018)

Max marks: 100 **Time: 180 mins**

Roll No.: _____ Name: _____ _____ _____

- ***Please write your name and roll number in the space provided at the top.***
- ***Please write your answer to each sub-question only in the space provided in the question paper. Answers written elsewhere will not be graded.***
- ***You must return the question paper along with your answer at the end of the exam.***
- ***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.*
- ***The use of electronic 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. Consider the Boolean function $F(x_1, \ldots x_4) = (x_1 \oplus x_2) \cdot ((x_3 + x_4) \oplus x_1) \cdot (x_1 \oplus \overline{x_4})$.

   (a) *[10 marks]* Construct an ROBDD for F using the variable order $x_1 < x_2 < x_3 < x_4$.

(b) *[10 marks]* Construct an AIG for F using (exactly) the following sequence of operations. You must show all AIG nodes obtained using the given sequence of operations. You must use only strash-ing (structural hashing) and no fraig-ing. You must also not use any re-write rules other than $AND(0, g)$ rewrites to 0, and $AND(1, h)$ rewrites to h for any AIG node g or h.

1 $t_1 = XOR(x_1, x_2)$
2 $t_2 = OR(x_3, x_4)$
3 $t_3 = XOR(t_2, x_1)$
4 $t_4 = XOR(x_1, \overline{x_4})$
5 $t_5 = AND(t_1, t_3)$
6 $F = AND(t_5, t_4)$

(c) *[10 marks]* We wish to design a circuit that implements F using only XOR and NOT gates. Do you think this is possible? If so, give the resulting circuit. Otherwise, explain why this is not possible. If you are answering in the affirmative, you will not get marks without a correct circuit. If you are answering in the negative, you will not get marks without justification.

2. A *synchronous token-ring arbiter* (henceforth called STA) is a synchronous sequential circuit with $2n$ *request* inputs, $R_0, \ldots R_{2n-1}$, and $2n$ *grant* outputs $G_0, \ldots G_{2n-1}$. The behaviour of the STA is as follows:

   - At the rising edge of the clock, the STA samples all its request inputs.
   - If none of the R_i's are sampled 1, all G_j are set to 0 in the same cycle.
   - If exactly one R_i is sampled 1, only the corresponding G_i is set to 1 and all other G_j's (for $j \neq i$) are set to 0 in the same cycle.
   - If more than one R_i are sampled 1, *exactly one* of the corresponding G_j's is set to 1, while all other G_k's (for $k \neq j$) are set to 0 in the same cycle.

     In order to determine which G_j should be set to 1 in a fair manner, a token is kept circulating (within the STA circuit) among the R_i inputs. When the circuit starts operation, R_0 has the token. At each rising edge of the clock, the token moves from R_i to $R_{(i+1) \pmod{2n}}$.

     If R_j is sampled 1 by the rising edge of the clock and if the token was with R_l immediately prior to this rising edge, then G_j is set to 1 if R_j the only request sampled 1 within the set $\{R_l, R_{(l+1) \pmod{2n}}, R_{(l+2) \pmod{2n}} \ldots R_j\}$. In other words, if we were to hypothetically move the token forward from R_l (i.e. from R_l to $R_{(l+1) \pmod{2n}}$ to $R_{(l+2) \pmod{2n}}$ and so on), then R_j would be the first request encountered whose value was sampled to be 1.

     Note that using the above mechanism, G_j may be set to 1 even if R_j didn't have the token in the current cycle.

   Unfortunately, building an STA for large values of n can become unwieldy. Hence a designer wants to come up with a synchronous sequential circuit called *modular token-ring arbiter* (henceforth called MTA), instances of which can be connected together to build an STA. Specifically, an MTA is a synchronous sequential circuit with two request inputs: r_1 and r_2, and two additional inputs: t_{in} (for "token in") and c_{this} (for "can grant request from this MTA"). It has two grant outputs: g_1 and g_2, and two additional outputs: t_{out} (for "token out") and c_{next} (for "can grant request from next MTA"). T_{in} and T_{out} are used to manage the circulation of the token among the different MTA's. Specifically, T_{in} being 1 means that the token is being given to the current MTA (by another MTA), and T_{out} being 1 means that the current MTA is passing the token to the next MTA. Similarly, if c_{this} is 1, it means that the current MTA can grant one of its incoming requests even if it doesn't have a token. Finally, if c_{next} is 1, the current MTA is informing the next MTA that it can grant one of its incoming requests, even if it doesn't have a token.

   The designer wishes to build an $2n$-input, $2n$-output SSA by connecting multiple MTA units as shown in Fig. 1 (for $n = 3$). Note that each MTA is a separate synchronous sequential circuit, and they communicate through each other only through $t_{in}, c_{this}, t_{out}$ and c_{next}. Assume that all MTAs are fed the same clock, and there is zero delay in the clock distribution network across MTAs and within each MTA. Therefore, the clock signal reaches all flip-flops at all MTAs at exactly the same time. In answering the following questions, remember that our goal is to be able to build a $2n$-input, $2n$-ouptut STA by interconnecting n MTAs as shown in Fig. 1.

   (a) *[20 marks]* Fill in the following *(partial)* state transition table for the synchronous sequential circuit MTA. Note that the table uses only 3 states, and ignores several inputs combinations (marked as -) depending on the state. Similarly, the outputs are don't cares (marked as -) for several state/input combinations. State "NoTok" signifies that the token is neither with r_1 nor with r_2 in the MTA. Similarly, state "r_iHasTok" signifies that the token is with r_i in the MTA. Assume also that when the interconnectd MTA units in Fig. 1 are started, the topmost MTA (corresponding to inputs R_0 and R_1 of the STA) starts in state"r_1HasTok" and all other MTA's start in state "NoTok".

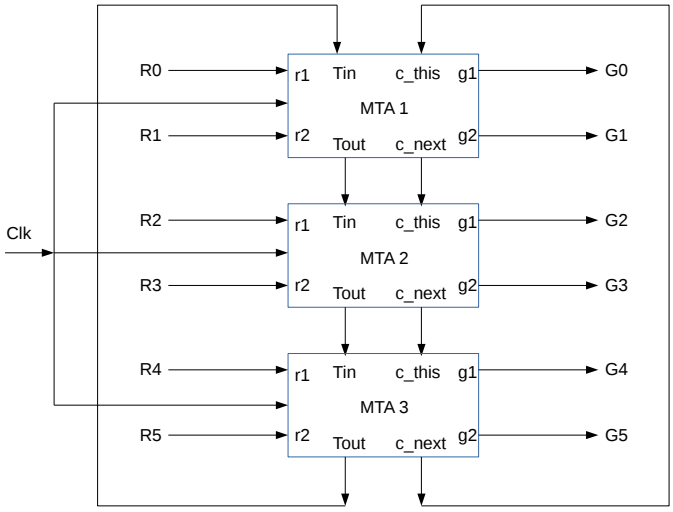

Figure 1: Arbiter Circuit

CurrentState	r_1	r_2	T_{in}	c_{this}	g_1	g_2	T_{out}	c_{next}	NextState
NoTok	-	-	0	0					
NoTok	-	-	1	0					
NoTok	0	0	0	1					
NoTok	0	1	0	1					
NoTok	1	0	0	1					
NoTok	1	1	0	1					
NoTok	0	0	1	1					
NoTok	0	1	1	1					
NoTok	1	0	1	1					
NoTok	1	1	1	1					
r_1HasTok	0	0	0	0					
r_1HasTok	0	1	0	0					
r_1HasTok	1	0	0	0					
r_1HasTok	1	1	0	0					
r_2HasTok	0	0	0	0					
r_2HasTok	0	1	0	0					
r_2HasTok	1	0	0	0					
r_2HasTok	1	1	0	0					

(b) *[10 marks]* The above table only showed some of the state and input combinations. Do you think there are state and input combinations in the complete table, for which *all outputs (other than NextSt)* can be don't cares, and yet the combination of MTAs will function as desired? If you think no such state/input combinations are possible, give reasons. Otherwise, list all such state and input combinations in the following format.

CurrentState	r_1	r_2	T_{in}	c_{this}	g_1	g_2	T_{out}	c_{next}	NextState

(c) *[15 marks]* Suppose the setup and hold times of flip-flops used in the circuit in Fig. 1 are 2ns each. Assume that the clock-to-q propagation delay of each flip-flop is 0 ns. Assume further that the maximum and minimum delays through the combinational logic in each MTA that implements the state transition table, part of which was shown in the previous sub-question, are 20ns and 100ns respectively. All wires may be assumed to have zero delay. Recall that all clock distribution networks are assumed to have 0 delay.

If the designer wants to build an STA (as in Fig. 1) that can arbitrate between 20 inputs, what is the maximum clock frequency that can be used to run the individual MTA units.

Given justification for your answer.

3. Consider two Boolean functions $F(x_1, x_2, x_3, x_4)$ and $G(x_3, x_4, x_5, x_6)$ and let $H(x_1, \ldots x_6) = FopG$, where *op* is a binary Boolean operator. Let $ODC_{F,x_3}(x_1, x_2, x_4)$ denote the ODC of x_3 in F, and similarly for $ODC_{G,x_3}(x_4, x_5, x_6)$.

   (a) *[5 marks]* Show that every satisfying assignment of $ODC_{F,x_3}(x_1, x_2, x_4) \cdot ODC_{G,x_3}(x_4, x_5, x_6)$ is also an ODC of H regardless of what F, G and *op* are.

(b) *[10 marks]* Give a K-map for F in which the absolute value of the difference in the number of squares (cells) filled with 1's and 0's is minimized (as small as possible), and $ODC_{F,x_3} = (x_1.x_2) \oplus x_4$ and $ODC_{F,x_4} = x_1 + x_2 + x_3$.

(c) *[10 marks]* Does there exist a binary Boolean operator *op* such that $ODC_{H,x_3}(x_1, x_2, x_4, x_5, x_6) = ODC_{F,x_3}(x_1, x_2, x_4) \cdot ODC_{G,x_3}(x_4, x_5, x_6)$ regardless of what F and G are? Give justification for your answer. We are not interested in Boolean operators that always evaluate to 1 or always evaluate to 0.