2201-COL215 Major exam

Aaveg Jain

TOTAL POINTS

99.5 / 120

QUESTION 1

Boolean postulates 15 pts

1.1 Part a **5 / 6**

✓ **+ 1 pts** *Closure is satisfied (1 is in {0,1})*

+ 1 pts Commutativity is satisfied (both operands are equal)

✓ **+ 1 pts** *Inverse can be satisfied (if we define 1 as the inverse of 0, and 1 AND 0 = 0)*

✓ **+ 1 pts** *Multiplicative identity is not relevant (since it applies only when an operand is '1')*

+ 0 pts Distributivity is ignored. As it was mentioned in paper to ignore hence no credit

✓ **+ 2 pts** *Correct answer: Yes*

+ 0 pts Incorrect

please write neatly

1.2 Part b **7.5 / 9**

✓ **+ 3 pts** *Assignment satisfies identity rules*

- 1.5 pts Justification not given for identity rules

✓ **+ 3 pts** *Assignment satisfies commutativity rules*

- 1.5 pts Justification not given for commutativity rules

+ 3 pts U' defined correctly

✓ **+ 1.5 pts** *U' not explicitly written but inverse postulate considered*

+ 0 pts Incorrect

QUESTION 2

2 D flipflop synchronous reset **10 / 10**

✓ **+ 5 pts** *Identify the correct signals (input and reset complement)*

✓ **+ 5 pts** *Used AND gate to feed the input of D flip flop*

+ 7 pts Used another flip flop for reset (only circuit), this will cost more area and timing delay will be more before reset occurs

+ 0 pts incorrect/not attempted

+ 0 pts No justification

QUESTION 3

Hamming code 10 pts

3.1 Flip the bit **5 / 5**

✓ **+ 2 pts** *Correct Answer*

✓ **+ 3 pts** *Explanation*

+ 0 pts Incorrect

3.2 Parity and data bits **5 / 5**

✓ **+ 5 pts** *Correct*

+ 0 pts Incorrect

✓ **+ 0 pts** *Justification*

QUESTION 4

4 3x8 decoder **10 / 10**

+ 0 pts Not Answered.

✓ **+ 5 pts** *connecting two decoders.*

✓ **+ 5 pts** *Correct final output.*

QUESTION 5

5 Magnitude comparator 10 / 10

✓ **+ 10 pts** *Correct*

+ 0 pts Incorrect/not attempted

+ 5 pts Partial (didn't explain which term generation is not possible via xor)

QUESTION 6

6 DRAM address sequence 0 / 10

✓ **+ 0 pts** *InCorrect*

+ 5 pts Mentioned Increasing or Decreasing order

+ 5 pts Justification is given

QUESTION 7

7 Representation of netlist 5 / 10

✓ **+ 2 pts** *Appropriate data structure identified (as graph)*

✓ **+ 2 pts** *Representation of gates (as node in graph)*

✓ **+ 2 pts** *Attribute to represent functionality of gate (enum {AND,OR, INV etc}, truth table or boolean expression)*

✓ **+ 2 pts** *Connections between gates (as edge in graph)*

✓ **+ 2 pts** *primary inputs and output representation (as node in graph)*

+ 0 pts Incorrect/not attempted

- 5 *Point adjustment*

input as leave will not work for the netlist. As in tree there can be only one path from parent node to leaf but in netlist multiple paths are possible

QUESTION 8

8 VHDL concurrent statements 9 / 10

✓ **+ 4 pts** *Identification of dependency*

✓ **+ 3 pts** *Correct Order of Execution*

+ 1 pts Manage Execution Status

✓ **+ 2 pts** *Example Given*

+ 0 pts Incorrect/Not Answered

QUESTION 9

9 FSM symbolic states 10 / 10

✓ **+ 7 pts** *Symbolic states can be used to represent any state/assignment encoding. Some encodings may lead to better circuit than others.*

✓ **+ 3 pts** *Starting with 00/01 fixes the encoding at the specification itself. (Limitation)*

+ 5 pts Partially Correct / Correct Idea

+ 0 pts Incorrect/Not Attempted

QUESTION 10

10 State reduction 9 / 10

✓ **+ 4 pts** *State reduction reduces the size of the netlist. - no*

✓ **+ 4 pts** *Correct explanation*

✓ **+ 1 pts** *Correct sequential logic example*

+ 1 pts Correct combination logic example

+ 0 pts Incorrect/Unattempted

QUESTION 11

11 Overlapping wires 14 / 15

✓ **+ 12 pts** *Correctly drawn circuit without directly using XOR gate*

✓ **+ 3 pts** *Correctly marked signals in circuit or correctly explained the changes in signals*

+ 0 pts Incorrect/ used non-allowed

gate/overlapping wires/ Wire crossings

- 1 *Point adjustment*

Not explained how NAND forms XOR.

Name: Aaveg Jain
Entry No: 2021CS1007

Department of Computer Science and Engineering, IIT Delhi
Digital Logic and System Design (COL 215)
I Semester 2022-23
Major Exam
Maximum Marks: 120
17 November 2022, 11:00 AM to 1:00 PM

Answer ONLY in the spaces indicated below the questions. Answers written elsewhere will be ignored.

1. **[6+9 = 15 Marks]**

   a. **[6 Marks]** Suppose we propose an algebra in which **0 AND 0** is defined to be **1**. Does this proposal satisfy the postulates of Boolean algebra we studied in class? Justify. [*Ignore the postulates that involve **OR**, as we haven't defined the **OR** operation.*]

(algebra has 2 values only.)

Yes; the postulates involving AND only are and comm. of AND

$x \cdot 1 = x$; $x \cdot \bar{x} = 0$ if $x = 0$, and we have 0 AND 0 = 1 from second postulate, observe $\bar{x}$ cannot be 0 as 0 AND 0 = 1 ≠ 0 thus set $\bar{x} = 1$ and 0 and $\bar{0} = 0$ and $1 = 0$; this does not violate the first postulate as as $x \cdot 1 = x$ and $1 = 0$ thus, it satisfies the postulates. (comm. is also satisfied)

   b. **[9 Marks]** Suppose we define an algebra with **THREE** values **{0,1,U}**, (where **U** could represent, for example, the *Uninitialised* state of a signal...although that is not important for this question). Fill in the following table to define the **AND** operation with values that are consistent with the postulates of Boolean algebra. Justify your answers. [*Fill in the value of **X AND Y** in the row corresponding to **X** and column corresponding to **Y**. Ignore the postulates that involve **OR**, as we haven't defined the **OR** operation.*]

	0	1	U
0	0	0	0
1	0	1	U
U	0	U	0

the relevant postulates are

① $x \cdot 1 = x$ ② $x \cdot \bar{x} = 0$, ③ commutativity of AND

define complements as follows:

$\bar{U} = U$, $\bar{0} = 1$, $\bar{1} = 0$

for row of 1, we have $U \cdot 1 = U$, $0 \cdot 1 = 0$, $1 \cdot 1 = 1$ (from first post.)

also, from second postulate

$U \cdot \bar{U} = U \cdot U = 0$, $1 \cdot \bar{1} = 1 \cdot 0 = 0$, $0 \cdot \bar{0} = 0 \cdot 1 = 0$

Let $0 \cdot 0 = 0$ and $U \cdot 0 = 0$ to obtain the given table. also, it is observed that commutativity is satisfied in all cases.

R	D	D'
0	1	1
0	0	0
1	0	0
1	1	0

$D = R' \cdot Q$

$R' \cdot Q = D$

R	D	D'
0		0
1		D

$D' = R \cdot D$

$= R' \cdot D$

$k \geq 2^k - 1$

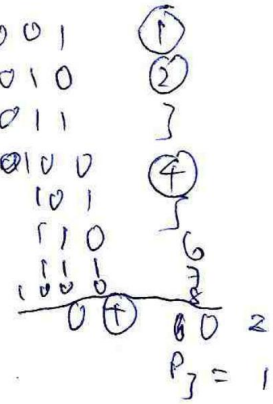
$n + k \geq 2^k - 1$

$n > 4 \quad n = 5$

$q > n \quad k = 3 \quad n = 5$

1	2	3	4	5	6	7	8
P_1	P_2	0	P_3	1	0	1	0
$=0$	$=0$		$=0$				

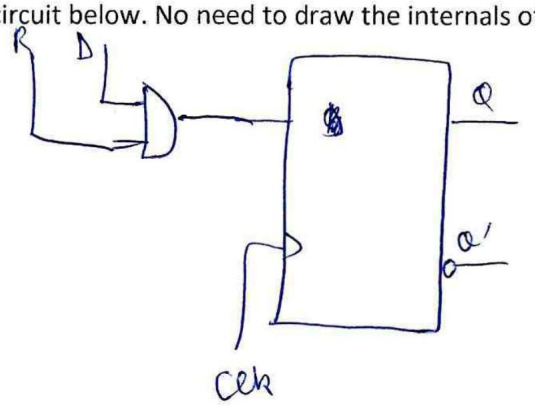
0 1 0 0 1 0 1 0



1 1 1

0 ⊕ 1
⊕ 1
3 6 7

2. **[10 Marks]** We have studied the design of a D flip-flop with an asynchronous reset, where the reset signal works independently of the clock. Design a variation of the D flip-flop with a **synchronous reset**. In this design, the reset signal does not act independently of the clock; instead, the reset signal influences the Q value only at the rising edge of the clock. Draw the circuit below. No need to draw the internals of the D flip-flop.



R is the reset signal and is active low

(we introduce a new input signal R)

3. **[5+5 = 10 Marks]** In the error correction mechanism using *Hamming Code*, where **k** parity bits are added to an **n**-bit data word, the error is corrected by flipping the bit corresponding to the **C** outputs.

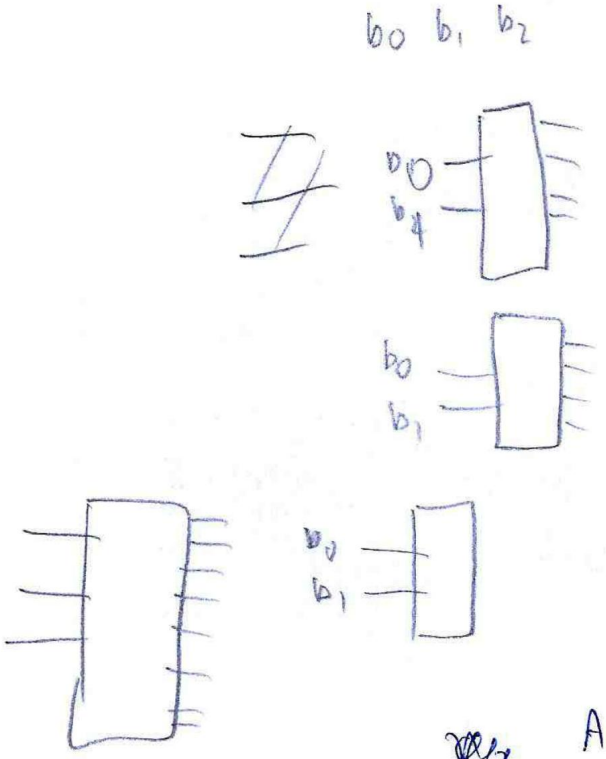
a. **[5 Marks]** If the **C** outputs do not point to a data bit, is it still right to flip the bit? Why?

Yes, if it doesn't point to a data bit, then also we should flip the corr. parity bit. If C points to a parity bit, it means that during read the parity bit got flipped. When we find C, for a pure position p we take XOR of all data bits with 1 at position p along with corr. parity bit also. Thus, if a parity bit flips, then the corr. parity bit position is reflected in C.

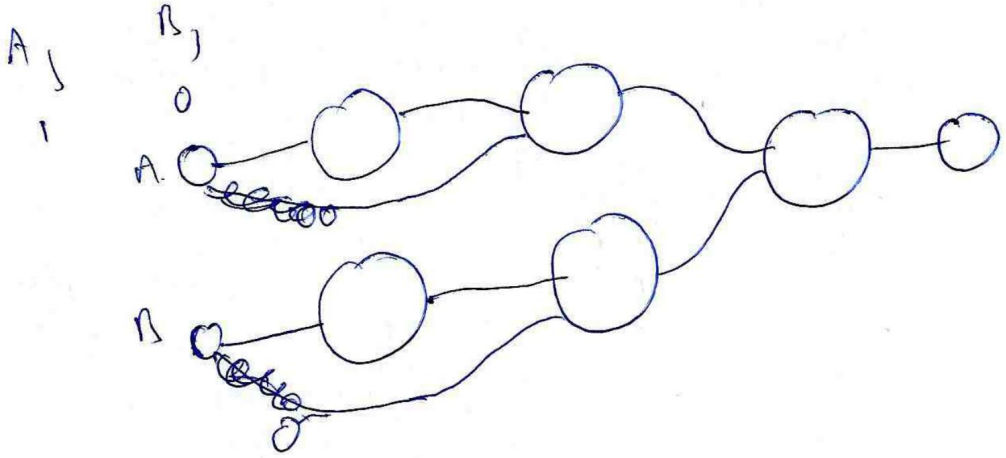
b. **[5 Marks]** What relationship must be satisfied between **n** and **k**, for the error correction to work properly?

C is of k bits (k parity bits)
thus range of values it can represent is 0 to $2^k - 1$.
assuming 0 is reserved for denoting no error, we have $2^k - 1$ possible values; this should be more than $n+k$ so that all positions can be acc. for, thus $2^k - 1 \geq n + k$

i.e. $2^k - k - 1 \geq n$

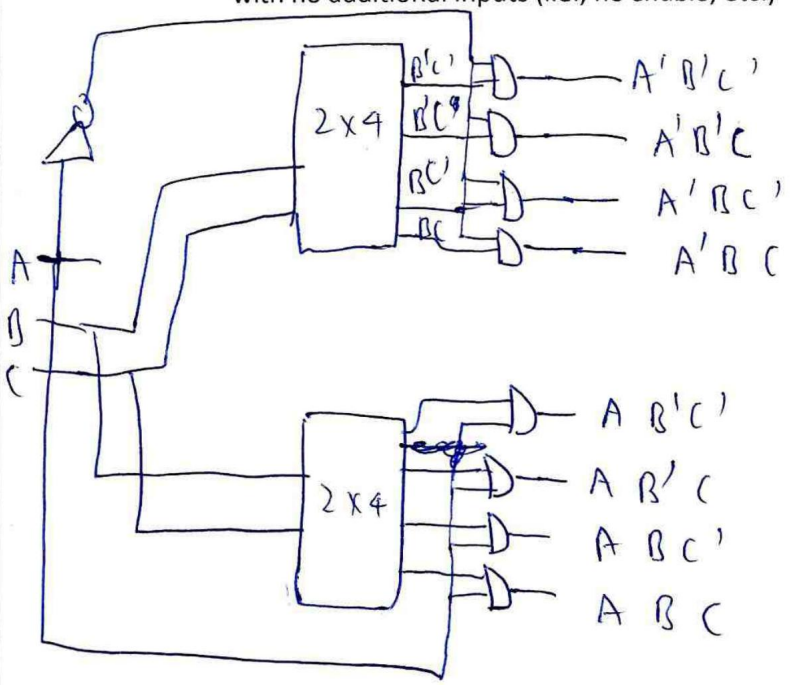


$A_3 B_3' + x_1 A_2 B_2' + x_1 x_2 A_1 B_1' + x_1 x_2 x_3 A_0 B_0'$

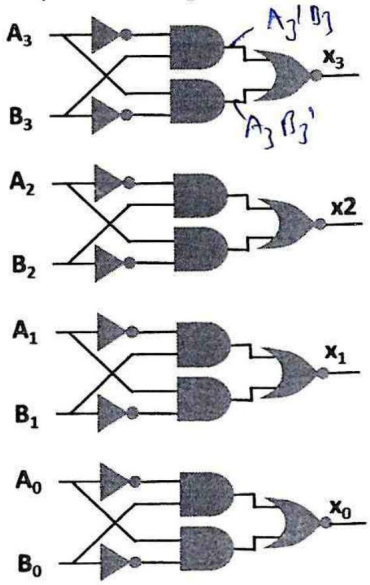


4. **[10 Marks]** We wish to design a **3x8 decoder** using **2x4 decoders** as components. Show how to design this using the 2x4 decoders and any additional gates you need. Assume that the 2x4 decoders have 2 input bits and 4 output bits, and implement the usual decoder functionality, with no additional inputs (i.e., no *enable*, etc.)

inp. — A, B, C



5. **[10 Marks]** Consider the *magnitude comparator* design discussed in class, shown below. Equality between bits can be checked directly with XOR gates. Why do we choose to build the comparator using the AND, INVERTER, etc., instead of just XOR gates?



we can take output corr. to A > B. (C)

$\Rightarrow C = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$

(ass. A_3, B_3 are the MSB)

now, in C we need the signals A_i, B_i' for all i, similiarly for B > A we would need A_i', B_i. ~~Thus~~ In ~~the~~ using AND, INVERTER etc., we need not find these signals again as they are already present as shown. & Had we directly used XOR gates, using AND, NOT etc. we would have to find these signals seperately, which would waste a hardware.

00
01
11
10

0 0 1 0
0 1 0 1
1 0 1 0
0 0 0 0
0 1 0 0

0 0 1 0
0 0 0 0
0 1 0 1
0 1 0 0
1 0 1 0

6. **[10 Marks]** Let $A_0,\ldots,A_{n-1}$ refer to a sequence of **n DRAM addresses** from which we wish to access data using the READ operation. Assume that the data accesses can be performed in any order. To optimise performance, what order of addresses should we choose? Justify.

Let row decoder size be r, col. decoder size be c

We should reshuffle the addresses s.t all the addresses with same row address (first r bits) come together and are then followed by the next set of addresses with same row address which is bigger than the row adder of this set.

eg. for r=c=2, if 0010, 0101, 1010, 0000, 0100, is the seq. we choose 0010, 0000, 0101, 0100, 1010. This is done as the row decoder output is stored in a row buffer, thus if the next adder's row adder is same it is not decoded again. In our choice, all adder with same row adder come together, thus the row adder needs to be decoded the min. no. of times, which is 1.

7. **[10 Marks]** Describe a data structure to represent a combinational gate-level netlist/circuit. You need to ensure that all relevant information is captured in this data structure. Typical operations on this data structure could be: (i) evaluate the outputs for a given set of inputs, or (ii) find the delay through the circuit. Justify your choices. Assume that all gates have one output bit, and one or more input bits.

Data str. – tree. The vertices of the tree will represent a logic gate with the no. of children = no. of inputs each node, which stores the signals it takes as input. The last gate in the netlist (which generates the output) is the root, and each gate is a vertice with children pointing to the gates from which it receives its input. The leaf nodes will be the first logic gates they encounter in the netlist, and they will contain a list of inputs they require.

While evaluating outp, we recursively traverse the tree top to bottom; when we reach a leaf we evaluate outp. using input list and return; in a non-leaf node we take the return value of the children, evaluate the outp. and return. Thus the root ret. the final answer.

for delay, we store the delay in each node of that logic gate. Then we recursively find max. delay among children, add delay of node, and return.

A <= B , C

P D <= A , B C

E <= A, B, C, D

8. **[10 Marks]** Given a set of concurrent VHDL signal assignment statements that are triggered (scheduled for execution) at the same time, in what order should we execute them in the simulator? Give the strategy and justify. Let the triggering signals be $x_1, x_2, \ldots$

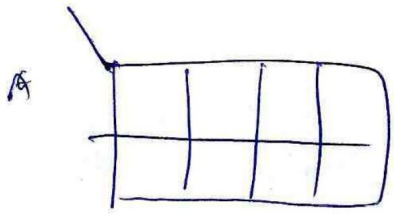
First we execute those stat. whose RHS depends only on $x_1, x_2, \ldots$ i.e they are of form $y_1 <= f(x_1, x_2, \ldots)$. etc. Then we execute those st. whose RHS depends on the prev. signals LHS & input signals only i.e $y_2 <= f(y_1, x_1, \ldots)$. The strategy maintain a list, whose initial el. are triggering signals. Then search for stat. whose RHS depends on signals in this list. & for each such st., evaluate RHS and add LHS to the list, & and store value of LHS. then repeat the same process with this new list, till all stat. have been evaluated. [When we evaluate a stat, delete it from the list of stat. to be searched]

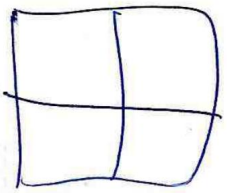
this is correct, as to evaluate a expr. of signals, we need to know value of each signal beforehand. So only those stat. can be eval. whose RHS depends only on triggering signals, then those whose RHS depends on the triggering and the prev. st. LHS, and so on. (since now we know value of prev. stat. LHS)

9. **[10 Marks]** In our Finite State Machine discussion, we argued that FSM states can be **symbolic** (S0, S1, etc., instead of 00, 01, etc.) What is the advantage of starting with symbolic states in the FSM? If the states are symbolic, it becomes easier to perform state reduct^n. & Also, it may happen that after state reduct^n, the no. of flip flops we require decreases. If we had initially chosen n flip flops and assigned states, and no. of states reduces after state red. s.t we now require lesser no. of & flip flops, then we will have to reassign states to acc. for that. This problem is not faced for symbolic states, where we assign states after state red^n and hence find no OFF at the end.

Also, after red^n, we can also see which state encoding gives us simpler logic circuits; this is not possible if we assign states in the beginning.

A	B	C	D
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1



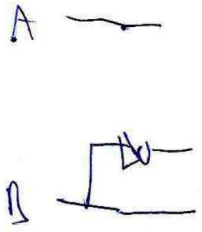


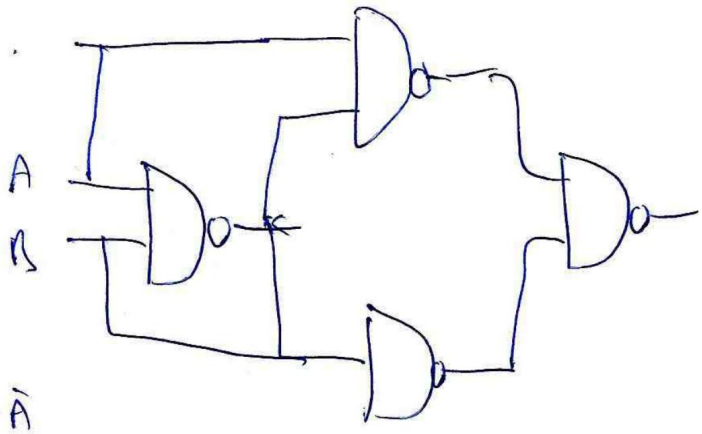
~~A⊕B~~ $A \oplus B$

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

C	D
0	0
1	0
0	1
1	1

$C = A \oplus A \oplus B$





~~A⊕B~~ $\overline{(\bar{A} + \bar{B}) + \bar{A}}$

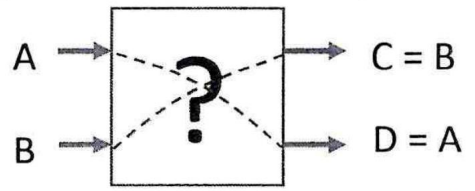
$= A \cdot B + \bar{A} =$ ~~A⊕B~~ $\underline{\bar{A} \oplus \bar{A} + B}$

$= (A \bar{B})$

10. [10 Marks] Is the *State Reduction* step guaranteed to reduce the size of the final generated netlist? Why?

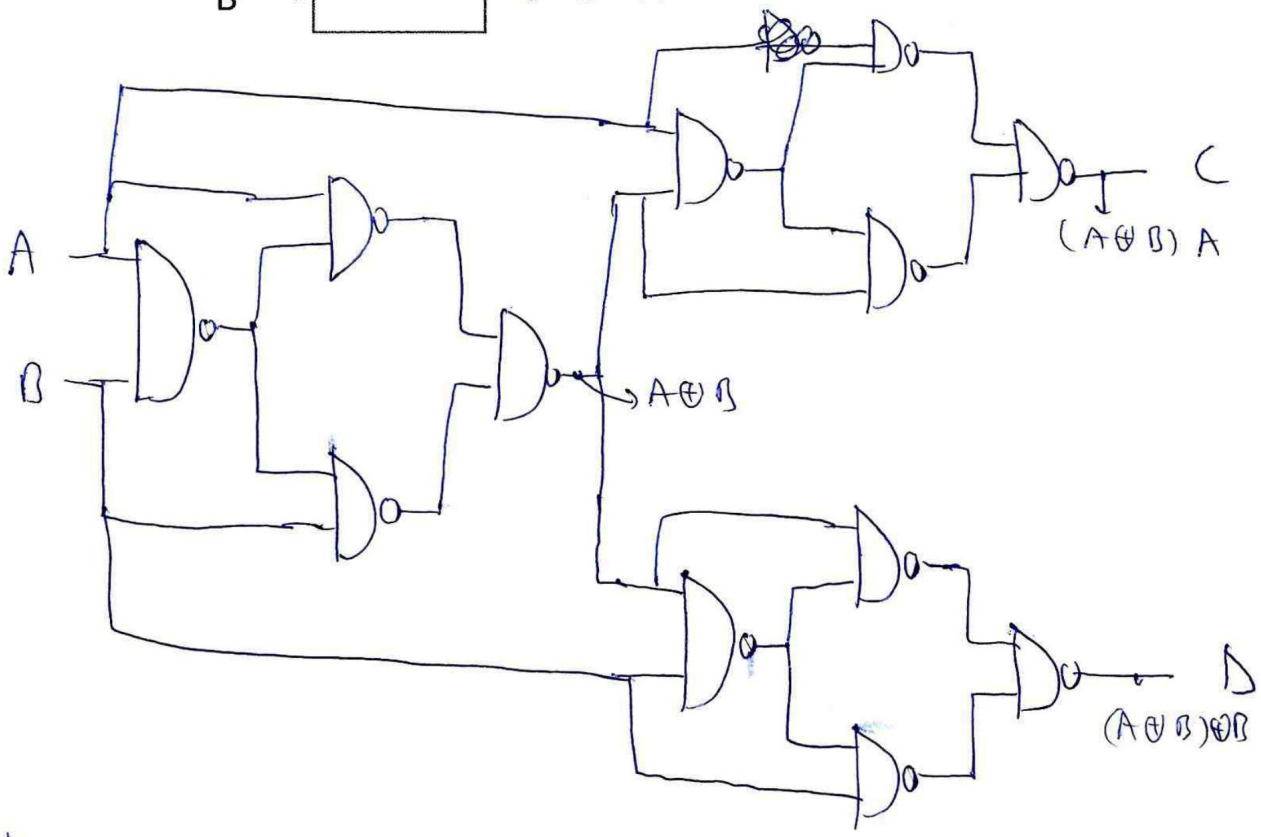
No, this step is not guaranteed to reduce the size of the final generated netlist. There are two factors - reduction in no. of flipflops, and Boolean function complexity of the output and input equation of flipflop. It is not necessary that state reduction leads to reduction in no. of FF. Even if it does, the complexity of the Boolean function of output and input of FF can easily increase. If the increase in no. of logic gates for Boolean function of output and input equation overcompensates the reduction in no. of FF, the netlist size can increase. Also, it is possible that no. of FF doesn't reduce, but complexity of Boolean function increases; thus netlist size can increase also in this step.

11. [15 Marks] Draw a combinational circuit to complete the following picture, where the order of input signals A and B, is reversed in the outputs, the condition being that **no two wires should cross each other**. You are allowed to use only the following gates: INVERTER, NAND, NOR, AND, OR. Assume that no wire crossings occur within the gates. [*Hint: Consider an application of the XOR function. Remember that using XOR gate directly is not allowed.*]



$C = A \oplus (A \oplus B)$

$D = B \oplus A \oplus B$



from TT, $C = (A \oplus B) \oplus A$

$D = (A \oplus B) \oplus B$