Name:
Entry No:

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*.]

Yes.
Closure is satisfied (1 is in {0,1})
Multiplicative identity is not relevant (since it applies only when an operand is '1')
Commutativity is satisfied (both operands are equal)
Distributivity is ignored (the postulates involve **OR**)
Inverse can be satisfied (if we define 1 as the inverse of 0, and 1 **AND** 0 = 0)

   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*.]

The following 5 values are derived from the Identity (and commutativity) rules: **1 AND x = x, x AND 1 = x**

	0	1	U
0		0	
1	0	1	U
U		U	

Assuming complement is unique, we can define U' = U itself. Since **U AND U'** needs to be 0 (by definition of complement), we have: **U AND U = 0**

0 AND U can be anything, but it must be equal to **U AND 0** (by commutativity)

0 AND 0 can be anything.

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.

   If data input is I, reset signal is R, and R=1 means we should reset Q=0:
   D input of flip-flop can be I AND 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.

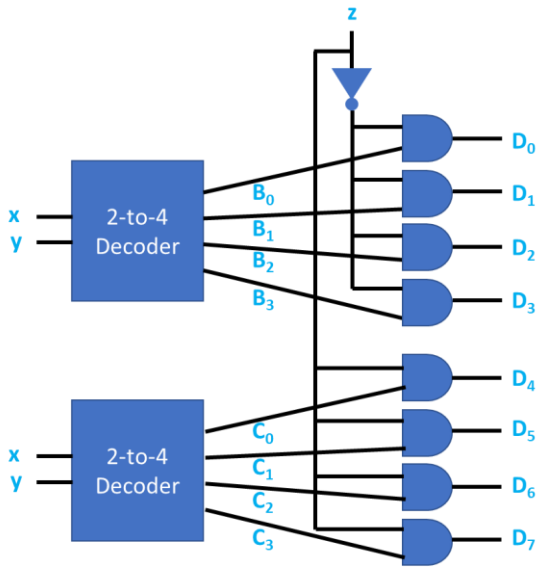
   a. **[5 Marks]** If the **C** outputs do not point to a data bit, is it still right to flip the bit? Why?

      Yes. If C doesn't point to data bit, it points to parity bit. Parity bit is incorrect and data bits are correct, so flipping the parity bit makes everything correct.

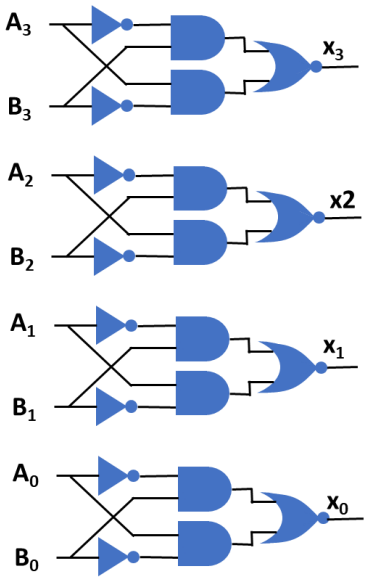
   b. **[5 Marks]** What relationship must be satisfied between **n** and **k**, for the error correction to work properly?

      $2^k \geq n + k + 1$
      The k bits should be able to represent position zero, and any position from 1 to n+k

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.)



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?



So that the AB', A'B values can be made available for the > and < operations. If XOR gates were used, these wouldn't be available.

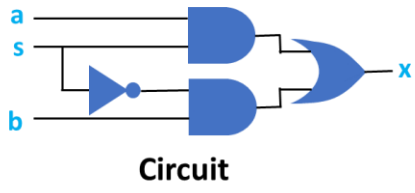
6. **[10 Marks]** Let $A_0,\ldots,A_{n-1}$ refer to a sequence of **n DRAM addresses** from which we wish to access data using the READ operation. Assume that the data accesses can be performed in any order. To optimise performance, what order of addresses should we choose? Justify.

We can sort the sequence in increasing or decreasing order and use this address sequence for the DRAM accesses. This will ensure that all accesses to the same row occur before we proceed to the next row.

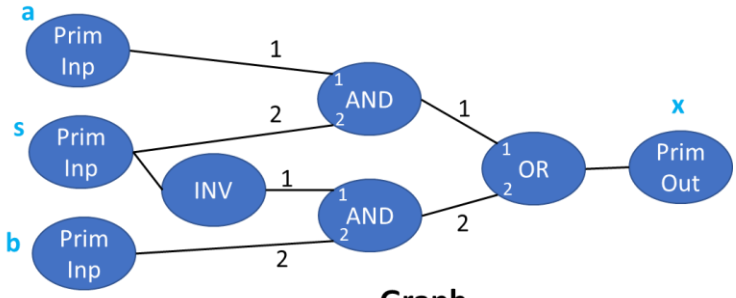
7. **[10 Marks]** Describe a data structure to represent a combinational gate-level netlist/circuit. You need to ensure that all relevant information is captured in this data structure. Typical operations on this data structure could be: (i) evaluate the outputs for a given set of inputs, or (ii) find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

We can use a **GRAPH** data structure in which the nodes represent gates and edges represent connections from the output of a source gate to input of a target gate. However, we need to keep track of a few additional points:

a. The edge should also contain, in addition to the target gate, the index of the target gate's input it is connected to (since the target gate may have multiple inputs).
b. Primary inputs should be nodes in the graph.
c. Primary outputs should be nodes in the graph.
d. The functionality of the node should be captured as a node attribute. It could be an enumeration value (one out of {AND, INV, OR, etc.}, or Truth table, or Boolean expression represented in some way.)



Circuit



Graph

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.

Some of the statements may be dependent on others. We start executing statements that are not dependent on any others. Mark them DONE. Proceed to other statements that depend on only DONE statements, etc. If there are cyclic dependencies, the order among statements in the cycle does not matter. This ensures that DONE statements are executed only once.

E.g.:
S1: a <= b
S2: c <= a + d
S2 is dependent on **S1**, so we should execute **S1** first, then **S2** (otherwise, we may have to execute **S2** twice, because **a** may change when **S1** executes).

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 we start with symbolic states, then we can use ANY state assignment/encoding. If we start with 00, 01, etc., then we are fixing the encoding at the specification itself. Some encodings may lead to better circuits than others.

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

No.

The number of flip-flops may not change (e.g., when the #states reduces from 6 to 5, we still need 3 bits to encode the states).

The combinational part of the circuit has an unclear dependence on the number of states (the resulting circuit may be smaller or larger).

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.*]