Name:	Entry Number:

Department of Computer Science and Engineering, IIT Delhi
Digital Logic and System Design (COL 215)
I Semester 2024-25, Major Exam, Maximum Marks: 80, 22 Nov 2024, 8:00 AM to 10:00 AM
Please write your answers ONLY IN THE SPACE BELOW THE QUESTIONS.
Use reverse side of paper for rough work.

1. **[5 Marks]** Given a set of implicants $P_1 \ldots P_n$ represented on a Karnaugh Map, how can we tell whether a given implicant P_i is an essential prime implicant?

   P_i is an essential prime implicant if it has at least one cell (minterm) that is not included in any other prime implicant in the K-map.

2. **[5 Marks]** Can a *WHILE*-loop in a VHDL model be translated into combinational hardware? Always? Sometimes? Never? Justify your answer.

   The WHILE-loop can be translated to combinational hardware SOMETIMES. We need to successfully unroll the loop. This unrolling is not always successful. However, sometimes, the information about loop condition, variable updates, and initial value, is enough for us to unroll the loop. Other conditions (such as no synchronisation/WAIT statements in the loop) also need to be met.

3. **[10 Marks]** Suppose we need to test a digital circuit/chip with *N* flip flops, *P* NAND gates, *M* input pins, *M* output pins, where we wish to initialize the flip flop values to a fixed pattern, run the chip for *L* cycles, and observe the flip flop values after the *L* cycles.
   What structural changes should we make to the circuit, assuming we want the testing procedure to complete as soon as possible?
   What is the minimum number of clock cycles needed to complete the testing procedure? Explain. [*Assume that there is only one clock in the chip. Assume we have only combinational logic and flip flops. Make any other reasonable assumptions and state them.*]

   Connect the N flip flops as M shift registers of length N/M each.
   Scan in M bits at a time over N/M cycles.
   Run for L cycles.
   Scan out M bits at a time over N/M cycles.

   Total time = 2N/M + L cycles.

   Combinational logic in the circuit (NAND gates) are not relevant.

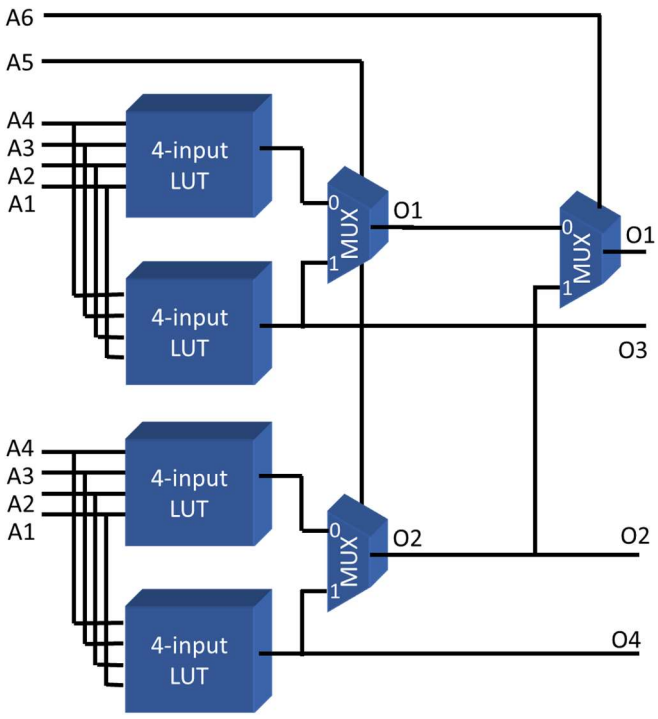
4. **[5 Marks]** Consider the programmable adder/subtractor circuit implemented through a ripple-carry adder structure. What is connected to the CARRY-IN bit of the first (least significant) full adder?

   The bit M that tells it to ADD or SUB, is connected to CARRY-IN.

   If M=0, then normal addition is performed.

   If M=1, then it provides the “+1” required in the conversion of the second operand B to a negative number (2's complement representation needs the “+ 1”)

5. **[10 Marks]** Suppose we have an FPGA combinational block consisting of 64 bits, stored as look-up tables. Show how we can use this to realise either ONE Boolean function of 6 input variables, TWO Boolean functions of 5 input variables each, or FOUR Boolean functions of 4 input variables each. Clearly indicate/label the input and outputs for each scenario.



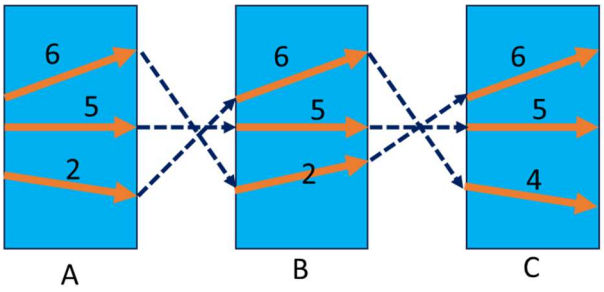
O1: 1 function of 6 inputs (A1,A2,A3,A4,A5,A6)

O1, O2: 2 functions of 5 inputs (A1,A2,A3,A4,A5)
A6 set to 0

O1,O2,O3,O4: 4 functions of 4 inputs (A1,A2,A3,A4)
A5, A6 set to 0

6. **[10 Marks]** Suppose we have 3 combinational logic blocks A, B, and C, with worst case delays D_A, D_B, and D_C, respectively. These 3 blocks are cascaded together to form a new block F. Is the worst case delay of F always given by $D_A + D_B + D_C$? Justify. Consider only gate delays and ignore wire delays.

No, the worst case delay need not be $D_A + D_B + D_C$. The worst case delay for F is $D_A + D_B + D_C$ only if the critical paths for A, B, and C are included in the critical path for F. In the figure below, $D_A = D_B = D_C = 6$, but the worst cast path delay for F = 5+5+5 = 15 (not 18).



7. **[10 Marks]** The state table of a Finite State Machine has 2^k rows. What does this tell us about the number of inputs, number of outputs, and number of states of the machine? Justify. [*Assume there are no don't care entries in the state table, and mention any other reasonable assumptions you make.*]

The number of inputs bits + state bits = maximum of k.

Assuming at least 1 bit for representing state.

Number of inputs can be anything between 0 and k-1.

Number of state bits can be anything between 1 and k. No restriction on the number of bits to use for encoding states, so the number of states can be anything from 1 to 2^k

Nothing can be said about the number of outputs.

8. **[10 Marks]** The DRAM READ operation can be thought of as consisting of three phases after the address is ready: (1) copying the addressed row from the cell array to the row buffer with delay A; (2) transferring the addressed data from the row buffer to the output pin with delay B; and (3) copying the row buffer data back to the cell array with delay C (since the original data is lost from the cell array in the process of copying it in the first phase). The total delay for a single READ operation is A+B+C in the worst case. How would we perform the DRAM WRITE operation, and what is the worst delay of a single DRAM WRITE operation? Justify. Assume that the delay for writing one data item from the input pin to the row buffer is D.

In a WRITE operation, we need to first copy the addressed row to the row buffer (Delay: A)
[*since the rest of the row needs to be unchanged*.]
Next, write one data item in the row buffer (Delay: D)
Next, copy the row buffer back to the cell array (Delay: C)

Total delay = A + C + D

9. **[5 Marks]** In the encoder implementation, where we had 8 input bits D_0-D_7, and 3 output bits, x, y, and z, we included only 8 rows, and derived the Boolean function for x as $D_4 + D_5 + D_6 + D_7$, etc. What is the justification for omitting the other rows? (Normally, we expect 2^8 rows for an 8-input Boolean function).

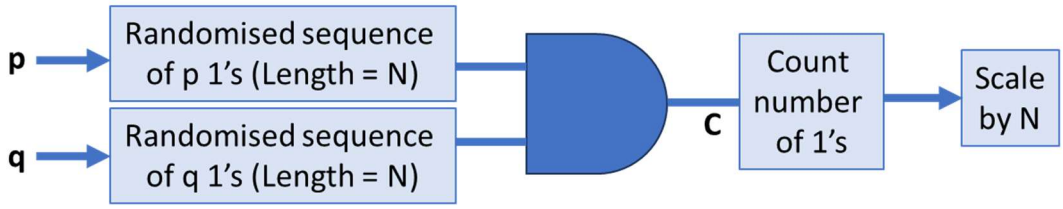
$D_0D_1D_2D_3D_4D_5D_6D_7$	x y z
1 0 0 0 0 0 0 0	0 0 0
0 1 0 0 0 0 0 0	0 0 1
0 0 1 0 0 0 0 0	0 1 0
0 0 0 1 0 0 0 0	0 1 1
0 0 0 0 1 0 0 0	1 0 0
0 0 0 0 0 1 0 0	1 0 1
0 0 0 0 0 0 1 0	1 1 0
0 0 0 0 0 0 0 1	1 1 1

Outputs are *don't care* for the other rows. This is part of the specification. For the encoder, we are allowed to have exactly one of the inputs as 1, and the remaining need to be 0.

10. **[10 Marks]** Consider an AND gate with inputs A and B, and output C. If the probability of A=1 is *p*, and probability of B=1 is *q*, then the probability of C=1 is *pq*. Use this idea to propose a probabilistic multiplier circuit with just *one* AND gate to implement the main multiplier function. The multiplier should take integers *p* and *q* as inputs and produce *pq* as its result. It is fine if you have flip-flops/registers, if there are some other combinational logic blocks (that aren't involved in the main multiplication function), and if the results have errors. Give the main ideas without going into detailed implementation. What does the accuracy of your multiplier depend on?

Convert p and q into a randomized sequence of N bits with p/q 1's and the rest 0's. Probabilities of A and B being 1 are p/N and q/N respectively. Probability of C=1 is p/N x q/N = pq/N^2. Number of 1's in the output sequence at C is pq/N^2 x N = pq/N. We can count the number of 1's at the output to obtain pq/N. From this, pq can be obtained by scaling with N (usually, this would be a shift operation, not requiring a separate multiplier).

Accuracy of the result increases with increasing N. This design would require a counter at the output, and a circuit for generating a randomized sequence at the inputs (omitted here).



Example: Length N = 10