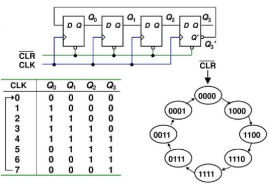
**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Registration: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1. (5 pts.)** In this problem, design a 4-bit Johnson Counter (the circuit and state diagram are given in **Figure 1**). Use as building block the D flip-flop shown below.

| **module DFF (D, clock, reset, Q, Qn);**  **input D; //Data input**  **input clock; //Clock input**  **input reset; //Asynchronous active-low reset**  **output Q, Qn; //Outputs Q and Q'**  **reg Q;**  **always @(posedge clock or negedge reset)**  **if (reset==1'b0)**  **Q <= 0;**  **else**  **Q <= D;**  **assign Qn = ~Q;**  **endmodule** |
| --- |

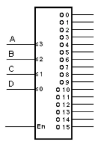


**Figure 1.** A 4-bit Johnson Counter

The suggested skeleton file of the Johnson Counter has been written below. The module has 2 inputs – CLK and CLR (which is active low). The output is Q which is 4-bit in size.

| **module JS\_CounterAB (CLK, CLR, Q);**  **input CLK, CLR;**  **output [0:3] Q;**  **//Write your code here**  **endmodule** |
| --- |

**2. (5 pts.)** In this problem, design a 4-to-16 Decoder with Enable input (the block diagram is given below).



The suggested skeleton file has been written below. The module has 5 inputs – A, B, C, D and En which is active high. The output is O which is 16-bit in size.

| **module Dec\_4x16 (A, B, C, D, En, O);**  **input A, B, C, D, EN; //A is MSB and D is LSB**  **output [15:0] O;**  **//Write your code here**  **endmodule** |
| --- |

**3. (5 pts.)** In this problem, combine the Johnson Counter (from **Problem 1**) with the Decoder (from **Problem 2**) to create a circuit such that the output of the Johnson Counter controls the data lines of the Decoder. Assume that the Decoder is enabled (i.e. the En input is tied to GND or 0).

The top-level design has the following port definitions:

CLK 1-bit clock

RESET 1-bit reset line

OUT\_DATA 16-bit data output

The suggested skeleton file has been written below.

| **module Top (CLK, RESET, OUT\_DATA);**  **input CLK, RESET;**  **output [15:0] OUT\_DATA;**  **//Write your code here**  **endmodule** |
| --- |