Name:

CA Test   **Variant 5**

---

1. Consider a Linear Feedback Shift Register (**LFSR**) on 7 bits, constructed according to the following output sequence: $q[0] \leftarrow q[6]$; $q[1] \leftarrow q[0]$; $q[2] \leftarrow q[1]$; $q[3] \leftarrow q[2]$; $q[4] \leftarrow q[3]$; $q[5] \leftarrow q[4] \oplus q[6]$; $q[6] \leftarrow q[5]$.

   a) Design, on a sheet of paper, the complete architecture of the 7-bit **LFSR** unit.
   b) Implement, using Verilog language, an **LFSR** module, using instances of a D-type flip-flop unit.
   c) Determine, by means of simulation, the output periodicity of the **LFSR** architecture.

2. Consider a combinational device having an input *i* and an output *o* on 4 bits, that encodes a 4-bit number in the **BCD-8421** format:

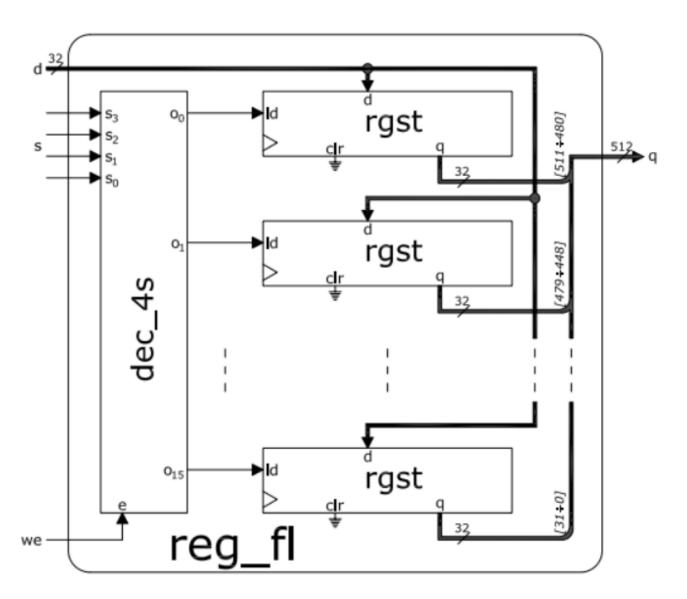| Inputs				Outputs			
I_3	I_2	I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	1
0	0	1	1	1	1	1	0
1	1	0	0	0	1	1	0
1	0	0	1	1	1	0	0
1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	0

Note: The unlisted inputs are expected to have an output decimal value of 3.

   a) Minimize the output functions on a sheet of paper using the Karnaugh maps method.
   b) Write a module that implements the resulting Boolean function after minimization. The module will be given a suggestive name (e.g. **BCD_encoder**).
   c) Build a testbench for exhaustive evaluation of the **BCD_encoder** module implemented in sub point b).

Name:

3. Construct a **16 x 38** Register File (**RF**) for assembling 38-bit data packets into a 608-bit block. The unit has a 38-bit input, **data**, a 4-bit selection line, **wr_data**, a 1 -bit enable line, **wr_en**, as well as **clock** and **reset** lines that are connected to each of the registers in the stack. For this:

    a) Design, on a sheet of paper, a 4-to-16 lines decoder **(dec_4x16)** architecture, and implement its module using Verilog language.

    b) Design, on a sheet of paper, a 38-bit register (**rgst**) architecture, and implement its module using Verilog language.

    c) Design, on a sheet of paper, the entire Register File by linking the above constructed designs together. Implement the **reg_fl** module, using **a generate block.**

    d) Implement, using Verilog language, a testbench for evaluating the functionality of the **reg_fl** module.

The architecture of a **16x32** Register File architecture is given below as support:

Score: **1.** a) 1p; b) 1p; c) 1p | **2.** a) 1p; b) 1p; c) 1p | **3.** a) 0,75p; b) 0,75p; c) 75p; d) 0,75p | *1 bonus point* | **Total**: 10p