Digital Design (CS F215) **14-10-2024**

Lab 9: Registers

Lecturer: Harikrishnan N B *Student:*

READ THE FOLLOWING CAREFULLY:

Honour Code for Students: I shall be honest in my efforts and will make my parents proud. Write the oath and sign it on the assignment.
For the lab, you need a dedicated fresh notebook. Keep this notebook dedicated for solving the questions pertaining to lab.

There will be design questions in the lab, the students are supposed to theoretically show the correctness of their design function. Following that, the students are encouraged to use iverilog software to implement the same.

The lab evaluation consists of two parts.

1) Step 1 (Theory): The students are supposed to solve the design question in their dedicated lab notebook. They are supposed to get it checked and validated by TA's/ instructors. (**Deadline for theory evaluation: 10:15 AM**)

2) Step 2 (Lab): Once Step 1 is completed, they can use iverilog to implement their design. Once you have completed the design, inform the TA's/ instructors and get it verified. (**Deadline for lab file submission in local quanta: 10:45 AM**)

3) Step 3: Upload your solution to quanta.bits-goa.ac.in (local quanta). Create a zip file CampusID_Lab9.zip which contains all the files.

`Note: The clock period is 10 seconds (with 50 % duty cycle). Initialize the clock to 0 for all the testbenches.`

All questions must be solved using the modeling mentioned in the question. Only solutions implemented using the mentioned modeling will be accepted for evaluation.

---

1. **Theory(4 Mark):** Draw a custom 4-bit Right-Shift Register using 4 positive-edge D Flip-Flops where at every clock cycle, A[3] is updated as follows: $newA[3] = A[0] \oplus A[2]$, where **A[i] represents the ith D-Flip-Flop from the right**. The other details of the custom Shift Register are the same as a standard Shift Register. First, the XOR is calculated using initial value of A[0] and A[2], then the right-shift occurs and then A[3] is updated to the calculated XOR value . The reset is active-high. This is known as an LFSR or Linear-Feedback Shift Register.

2. **Lab(4 Marks):** Complete the module `lfsr.v` in Verilog using a behavioural model to implement the LFSR. The module should take a posedge clock input (`clk`), an active-high set input (`set`) that is used to initialize the LFSR Flip-Flops states to **1001**. The output (`A[3:0]`) represents the state of the Flip-Flops in the register. The state of the register should change only at the rising edge of the clock cycle. Test the output by running the `tb_lfsr.v`, which has already been implemented for you. **Do not make any changes to the template code provided to you.**

3. **Theory(14 marks):** LFSRs are used to generate keys in cryptography. The encryption of the input is done bit-by-bit using the XOR operation. The equation is as follows: $Oi = Ki \oplus Ii$, where O is the output, I is the input and K is the key. Oi denotes the ith bit of the Output and likewise for K and I . This operation works from left hand side to right. For example, if the input is **10010** and key is **01010**, the output will be **11000**.

   Key Generation: The key is generated bit-by-bit every clock cycle. At every clock cycle, the rightmost flip-flop of the LFSR is used as the key-bit. This key-bit is then used to encrypt the ith bit of the input from the left. After this, the shifting of the LFSR states occur. **Please refer to table 9.1 to understand the key generation of the LFSR that you made in Q1 and Q2.**

   Note: In Q1 and Q2, you used A[2] XOR with A[0] to get the new A[3], however for a general LFSR, we can XOR any one of the flip-flops(A[1], A[2], A[3]) with A[0] to get the new A[3].

   Question: You are a cybersecurity expert at Rubrik. A spy was able to obtain the input and output data from rival company Arista. You know the rival company uses a 4-bit LFSR to encrypt their data. You are tasked with finding the workings of the encryption. You know that the **input is 10100111** and the **output is 00010101.** Your job is to:

   (a) Find the key used to encrypt the input.

   (b) Find what is the initial state of the LFSR required to generate this key.

   (c) Find the flip-flop in the LFSR used to XOR A[0] to get the new A[3]. (Refer to note above).

4. **Lab(8 marks):** Implement the above encrypter in `enc.v` in Verilog using behavioural modelling. The module should take a posedge clock input (`clk`), an active-high set input (`set`) that is used to initialize the LFSR Flip-Flops states. It also takes an input (`in`) which is the input bit we have to encrypt. The output (`A[3:0]`) represents the new state of the Flip-Flops after the encryption and shift. The output (`out`) represents the encrypted output bit. The state of the register should change only at the rising edge of the clock cycle. Output bit is also generated only at the rising edge of the clock.

   Testbench details: We have provided a new input stream to you along with the time in Table 9.2 Use this input to test your encrypter in `tb_enc.v` **Do not make any changes to the template code provided to you.**

Table 9.1: Key Generation

State of LFSR	Key Bit Generated
Initial state: 1001	1
1100	0
1110	0
1111	1
0111	1
0011	1
1001	1

Table 9.2: Testbench Input

Time	**Inputs**
0	set = 1, in = x
10	set = 0, in = 1
20	in = 0
30	in = 1
40	in = 1
50	in = 0