**Design Challenges**

***Challenge #1:***

Design of an LFSR

A linear feedback shift register (LFSR) is a shift register whose input bit is the output of a linear function of two or more of its previous states (taps). An LFSR of length m consists of m stages numbered 0 , 1 , … , m − 1 , each capable of storing one bit, and a clock controlling data exchange. It generally has a seed value that can be loaded into it in the beginning and it circles through a series of states.

An example is given below:



***Challenge #2:***

Design an ALU

The specifications are given below:

| 0000 | ALU\_Out = A + B;

----------------------------------------------------------------------

| 0001 | ALU\_Out = A - B;

----------------------------------------------------------------------

| 0010 | ALU\_Out = A \* B;

----------------------------------------------------------------------

| 0011 | ALU\_Out = A / B;

----------------------------------------------------------------------

| 0100 | ALU\_Out = A << 1;

----------------------------------------------------------------------

| 0101 | ALU\_Out = A >> 1;

----------------------------------------------------------------------

| 0110 | ALU\_Out = A rotated left by 1;

----------------------------------------------------------------------

| 0111 | ALU\_Out = A rotated right by 1;

----------------------------------------------------------------------

| 1000 | ALU\_Out = A and B;

----------------------------------------------------------------------

| 1001 | ALU\_Out = A or B;

----------------------------------------------------------------------

| 1010 | ALU\_Out = A xor B;

----------------------------------------------------------------------

| 1011 | ALU\_Out = A nor B;

----------------------------------------------------------------------

| 1100 | ALU\_Out = A nand B;

----------------------------------------------------------------------

| 1101 | ALU\_Out = A xnor B;

----------------------------------------------------------------------

| 1110 | ALU\_Out = 1 if A>B else 0;

----------------------------------------------------------------------

| 1111 | ALU\_Out = 1 if A=B else 0;

A sample of verilog module configuration is given below:

**module** alu(

**input** [**7**:**0**] **A**,**B**, // ALU 8-bit Inputs

**input** [**3**:**0**] ALU\_Sel, // ALU Selection

**output** [**7**:**0**] ALU\_Out, // ALU 8-bit Output

**output** CarryOut // Carry Out Flag

);

***Challenge #3:***

Design a state machine to detect a given input 3-bit sequence in a 32-bit sequence

32 bit

seq\_detector

seq\_found

3 bit

In the given 32 bit sequence should be traversed serially and when ever the 3-bit sequence is detected the seq\_found should be asserted HIGH. When writing a test bench for the above design, a counter logic should be implemented to count the number of seq\_found signals generated. Also, a logic can be implemented in the test bench to automatically verify if the seq\_detector is working as expected.

***Challenge #4:***

Design a clock with HH:MM:SS format

The clock should count from 00:00:00 to 23:59:59

***Challenge #5:***

Design the following micro-architecture

Control Logic

Shifter

Counter

The following are the requirements:

1. Both shifter and counter have the parallel load feature and they are 8 bit
2. When the counter reaches a given count value one bit in the shifter should be shifted out serially
3. When all the bits are shifted out, then the entire process can begin again
4. Data can be loaded to shifter externally, however, once the shifting starts, the next data can be input to the shifter only after all the bits are shifted out.

Design the control logic as needed to meet all the requirements above.

Also, come up with the input and output signals as per the requirement for the top level module.